SECURWARE 2013

The Seventh International Conference on Emerging Security Information, Systems and Technologies

ISBN: 978-1-61208-298-1

August 25-31, 2013

Barcelona, Spain

SECURWARE 2013 Editors

Hans-Joachim Hof, Munich University of Applied Sciences, Germany

Carla Westphall, Federal University of Santa Catarina, Brazil

SECURWARE 2013

Foreword

The Seventh International Conference on Emerging Security Information, Systems and Technologies [SECURWARE 2013], held between August 25-31, 2013 in Barcelona, Spain, continued a series of events covering related topics on theory and practice on security, cryptography, secure protocols, trust, privacy, confidentiality, vulnerability, intrusion detection and other areas related to low enforcement, security data mining, malware models, etc.

Security, defined for ensuring protected communication among terminals and user applications across public and private networks, is the core for guaranteeing confidentiality, privacy, and data protection. Security affects business and individuals, raises the business risk, and requires a corporate and individual culture. In the open business space offered by Internet, it is a need to improve defenses against hackers, disgruntled employees, and commercial rivals. There is a required balance between the effort and resources spent on security versus security achievements. Some vulnerability can be addressed using the rule of 80:20, meaning 80% of the vulnerabilities can be addressed for 20% of the costs. Other technical aspects are related to the communication speed versus complex and time consuming cryptography/security mechanisms and protocols.

Digital Ecosystem is defined as an open decentralized information infrastructure where different networked agents, such as enterprises (especially SMEs), intermediate actors, public bodies and end users, cooperate and compete enabling the creation of new complex structures. In digital ecosystems, the actors, their products and services can be seen as different organisms and species that are able to evolve and adapt dynamically to changing market conditions.

Digital Ecosystems lie at the intersection between different disciplines and fields: industry, business, social sciences, biology, and cutting edge ICT and its application driven research. They are supported by several underlying technologies such as semantic web and ontology-based knowledge sharing, self-organizing intelligent agents, peer-to-peer overlay networks, web services-based information platforms, and recommender systems.

We take here the opportunity to warmly thank all the members of the SECURWARE 2013 Technical Program Committee, as well as the numerous reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to SECURWARE 2013. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the SECURWARE 2013 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that SECURWARE 2013 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in emerging security information, systems and technologies.

We are convinced that the participants found the event useful and communications very open. We hope Barcelona provided a pleasant environment during the conference and everyone saved some time for exploring this beautiful city.

SECURWARE 2013 Chairs:

SECURWARE Advisory Chairs
Juha Rőning, University of Oulu, Finland
Catherine Meadows, Naval Research Laboratory - Washington DC, USA
Petre Dini, Concordia University, Canada / China Space Agency Center - Beijing, China
Reijo Savola, VTT Technical Research Centre of Finland, Finland
Masaru Takesue, Hosei University, Japan
Mariusz Jakubowski, Microsoft Research, USA
Emmanoil Serelis, University of Piraeus, Greece
William Dougherty, Secern Consulting - Charlotte, USA

SECURWARE 2013 Industry Liaison Chair
Rainer Falk, Siemens AG - München, Germany

SECURWARE 2013 Research/Industry Chair
Mariusz Jakubowski, Microsoft Research, USA

SECURWARE 2013

Committee

SECURWARE Advisory Chairs

Juha Rőning, University of Oulu, Finland
Catherine Meadows, Naval Research Laboratory - Washington DC, USA
Petre Dini, Concordia University, Canada / China Space Agency Center - Beijing, China
Reijo Savola, VTT Technical Research Centre of Finland, Finland
Masaru Takesue, Hosei University, Japan
Mariusz Jakubowski, Microsoft Research, USA
Emmanoil Serelis, University of Piraeus, Greece
William Dougherty, Secern Consulting - Charlotte, USA

SECURWARE 2013 Industry Liaison Chair

Rainer Falk, Siemens AG - München, Germany

SECURWARE 2013 Research/Industry Chair

Mariusz Jakubowski, Microsoft Research, USA

SECURWARE 2013 Technical Program Committee

Habtamu Abie, Norwegian Computing Center - Oslo, Norway
Maurizio Aiello, National Research Council of Italy - IEIIT, Italy
Hamada Alshaer, Khalifa University of Science, Technology & Research (KUSTAR), UAE
Claudio Agostino Ardagna, Università degli Studi di Milano, Italy
David Argles, Haven Consulting, UK
George Athanasiou, KTH Royal Institute of Technology, Sweden
Feng Bao, Institute for Infocomm Research, Singapore
Ilija Basicevic, University of Novi Sad, Serbia
Lejla Batina, Radboud University Nijmegen, The Netherlands
Georg T. Becker, University of Massachusetts Amherst, USA
Francisco Jose Bellido Outeiriño, University of Cordoba, Spain
Jorge Bernal Bernabé, University of Murcia, Spain
Catalin V. Birjoveanu, "Al.I.Cuza" University of Iasi, Romania
Lorenzo Blasi, Hewlett-Packard, Italy
Carlo Blundo, Università di Salern, Italy
Wolfgang Boehmer, Technische Universitaet Darmstadt, Germany
Ravishankar Borgaonkar, Technical University Berlin and Deutsche Telekom Laboratories, Germany
Jérémy Briffaut, ENSI - Bourges, France
Julien Bringer, SAFRAN Morpho, France
Martin Brunner, Fraunhofer Research Institution AISEC - Munich, Germany

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Identifying Suitable Attributes for Security and Dependability Metrication

Erland Jonsson, Laleh Pirzadeh
Department of Computer Science and Engineering
Chalmers University of Technology
Göteborg, Sweden
{erland.jonsson, laleh.pirzadeh}@chalmers.se

***Abstract*— In this paper, we suggest a framework for security and dependability metrics that is based on a number of non-functional system attributes. The attributes are the traditional security attributes (the "CIA") and a set of dependability attributes. Based on a system model, we group those attributes into protective attributes and behavioural attributes and propose that metrication should be done in accordance. We also discuss the dependence between these two sets of attributes and how it affects the corresponding metrics. The metrics themselves are only defined to a limited degree. The concepts of security and dependability largely reflect the same basic system meta-property and are partly overlapping. We claim that the suggested approach will facilitate making quantitative assessment of the integrated concept of security and dependability as reflected by those attributes.**

Keywords - security and dependability metrics; security and dependability modelling; protective metrics; behavioural metrics

I. INTRODUCTION

There exists a large number of suggestions for how to measure (or metricate) security, with different goals and objectives. The application areas range from business management and organizational systems to large software systems. The approaches may be theoretical, technical, administrative or practical. In many cases, the goal is to find a single overall metric of security. Given that security is a complex and multi-faceted property, we believe that there are fundamental problems to find such an overall metric. In this paper, we suggest a restricted view on security [29] as being only the integrity attribute of the dependability-security concept. Thus, we start out from a conceptual system model that integrates security and dependability. Other approaches have been suggested, e.g., by emphasizing the uncertainty dimension [11] or using ontologies [25]. Further, an excellent overview and classification is given in [33]. Our model is an input-output model in the sense that it describes a system's interaction with its environment via the system boundaries [15, 38]. The model identifies the main attributes of security and dependability. It clarifies the relation between malicious environmental influence on the input side and the service output to the users of the system. Based on the model we regroup the traditional security and dependability attributes into protective attributes and behavioural attributes. We argue that metrics for dependability and security attributes can be defined in accordance. Thus, protective attributes can be metricated by protective metrics and the behavioural attributes by behavioural metrics as originally proposed in a short paper [31]. Here, we extend and detail this original proposal. Also, we apply a metrication process perspective and discuss the system-related dependencies between different types of metrics. This approach is different from existing approaches to clearly relate the metrics to system input and output attributes and to address the impact of latency aspects.

In the following, Section II gives a brief summary of traditional security and dependability attributes. Section III describes the security model. The three defence lines in the model are described in Section IV as well as the causal relationship between the impairments in the system model. In Section V, security metrication according to the model is suggested. Section VI discusses the dependence between protective and behavioural metrics and Section VII briefly describes some benefits with our approach. Finally, we conclude the paper in Section VIII.

II. TRADITIONAL DEFINITIONS OF SECURITY AND DEPENDABILITY

In this section, we briefly summarize the traditional security and dependability terminology. Security is normally decomposed into three different aspects: *confidentiality*, *integrity* and *availability* [8], loosely called "the CIA". Confidentiality is the ability of the computing system to prevent disclosure of information to unauthorized parties. Integrity is the ability of the computer system to prevent unauthorized withholding, modification or deletion. Availability is the ability of the system to in fact deliver its service. More formally, it can be described as the probability that the system will be available, or ready for use, at a certain instant in time. Sometimes other characteristics are also suggested as security aspects, e.g., authentication and non-repudation, e.g., see [6, 9].

Dependability, on the other hand, is decomposed into the attributes: *availability*, *reliability*, *safety*, *integrity* and *maintainability* [2]. Here, reliability is a characteristic that reflects the probability that the system will deliver its service under specified conditions for a stated period of

 ISBN: 978-1-61208-298-1

time. Safety denotes the system's ability to fail in such a way that catastrophic consequences are avoided. Thus, safety is reliability with respect to catastrophic failures. (Please note that there exist several other definitions of safety, e.g., in the software development area [2, 39].) Availability and integrity are defined as above. Finally, the maintainability attribute denotes the system's ability to undergo modifications and repairs.

It must be noted that the original dependability fault assumption was that of non-malicious, "stochastic" or "random" faults, such as those resulting from a component failure, rather than deliberate, malicious security faults (attacks). Such arbitrary faults might be internal faults, occurring (seemingly) spontaneously within the system, as well as external faults. Nowadays, both non-malicious and malicious faults are considered in existing models. However, because of the difficulty of making a formal or mathematical treatment of deliberate, malicious faults, most research so far has been done on dependability with a random fault assumption.

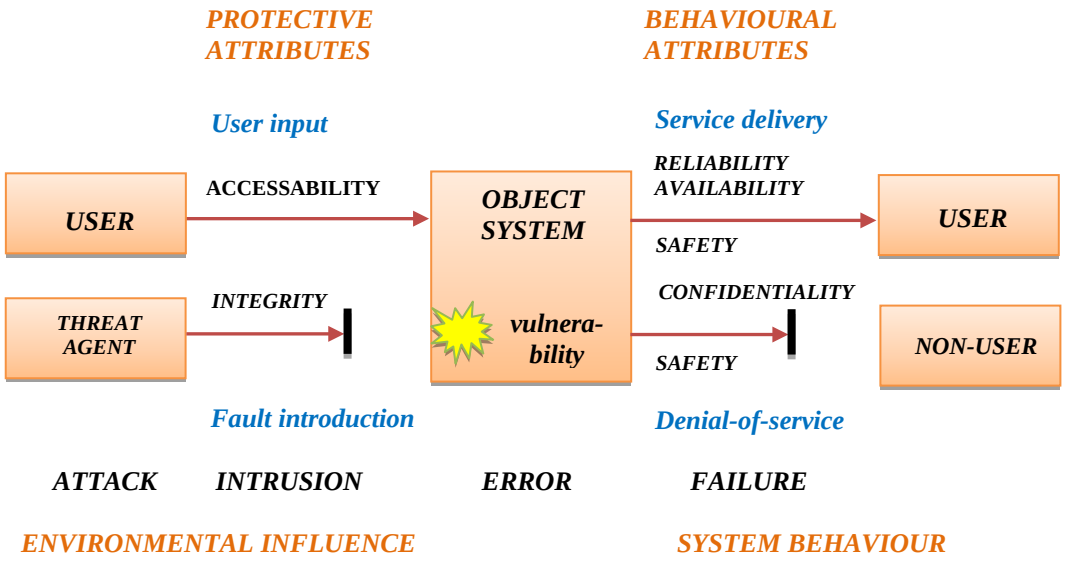


Fig. 1. An integrated model of security and dependability

III. A CONCEPTUAL SYSTEM MODEL

A. *Interaction between the system and the environment*

This section gives a brief description of the system model for security and dependability attributes originally proposed by Jonsson [15]. Once again, for simplicity, we use the term *security* to denote the combined concept of security and dependability.

Our approach is that the security of a system should be understood in relation to its environment, in terms of system input and output. First, we define the system that we are considering, the ***object system***. It is important to clarify the boundaries of the object system, since the subsequent discussion of the security model is based upon a well-defined system. The object system may be arbitrarily complex: a single computer, a computer network or possibly a whole organisation, including people. Note that by studying a larger system more of the potential problems are "embedded" into the system as internal or insider problems. These problems are not directly addressed in the paper. The object system interacts with the environment in two basically different ways. The object system either receives an *input* from the environment, or delivers an *output* to the environment; see Figure 1. The input to the system is denoted ***environmental influence***. The environmental influence may be of many different kinds. The type of interaction we are interested in here is that which involves fault introduction. Malicious, external faults, i.e., attacks, are particularly interesting. Such faults originate from a ***threat*** (or ***threat agent***) in the environment. The threat may be a human being, a natural phenomenon or another computer system, among other things. The threat agent launches an ***attack*** towards the system. The attack will be successful if it can exploit a ***vulnerability*** in the system so that an ***intrusion*** results. The result of the intrusion can be regarded as an ***error*** (or erroneous state) in the system. Note that a vulnerability is a passive feature of the system as opposed to an error. The error may (or may not) propagate and lead to a system ***failure***. This depends on the implementation of the system, how it is operated, what defensive mechanisms are active etc. Thus, there is a causal relationship between those ***impairments***: fault/attack, error/intrusion and failure. Further details on impairments and their interaction can be found in [2, 14].

 ISBN: 978-1-61208-298-1

B. Defining the system attributes

We will now discuss the relation between these impairments and security aspects. Since faults are detrimental to the system, we seek to design the system such that the introduction of faults is prevented. (This is marked as a bold "stop-bar" in Figure 1.) We denote this ability ***integrity***. It is thus a ***protective attribute*** of security. The conceptual output from the object system is the ***system behaviour***. The system behaviour includes the notion of the degree of service delivery to the *authorized user* of the system, in the following denoted USER, and to the *non-authorized user*, denoted NON-USER.

Thus, the required system behaviour is different for USERs and NON-USERs. The desired ***service delivery*** to the USER is described by the ***availability*** and ***reliability*** attribute. The other desired quality is that the system shall have an ability to deny service, denoted ***denial-of-service***, to the NON-USER. (Marked by a "stop-bar".) Note the duality of these concepts. The normal and preferred situation with respect to the USER, i.e., that the service is indeed delivered, implies a failure with respect to the NON-USER and vice versa. If the service denied relates to information it is described by the behavioural attribute ***confidentiality***. In case it relates to other services, we use the word ***exclusivity*** [18]. Thus, exclusivity is the ability of the system to deny unauthorized use of system service.

Finally, the ***safety*** attribute introduces another aspect of system behaviour. It models the severity of a failure in the sense that it maps failures into catastrophic and non-catastrophic failures. All failures that are regarded as catastrophic, whether they represent a failure of service delivery or a failure of denial-of-service, are represented by the safety attribute. Thus, safety failures represent subsets of reliability/availability failures or confidentiality/exclusivity failures. An example of a "catastrophic failure" is a failure in the drive-by-wire system of a car that would lead to an accident, with possible casualties. Another example is the unauthorized disclosure of secret, military information that would have disastrous consequences in case of war.

The ***maintainability*** attribute has no place in our model, as it does not describe an operational system-environmental interaction. Maintainability rather represents the efficiency of the implementation of a security mechanism that is aimed at making the system security design better (more secure, reliable, safe, etc).

C. A limitation: The binary assumption for impairments

It must be noted that throughout this paper we have implicitly applied a binary model of our impairments. For example, we have assumed that the system is functioning or non-functioning, i.e., that there is a failure or there is no failure. Is is obvious that in many cases this is an over-simplification. In reality, the system will not fail completely, but only to a certain degree. It may continue to work, but with degraded service delivery or degraded performance. This aspect is encompassed by the attribute *performability*. See [34] and references therein.

There have been a few studies on behavioural metrics considering the degradation approach. In [12], a practical dependability metric for degradable computer systems with non-exponential degradation was proposed. The dependability attributes covered by this approach were: reliability, safety and performability. Markov modelling with phase-type assumption to enhance assessment of systems with non-exponential and time-dependent degradation was used. These types of studies have a good potential of being applied in behavioural security metrication.

We have also used the binary assumption on the input side in that we say that there is an intrusion or there is no intrusion. This assumption is also a significant simplification. We all know that intrusions are in many cases something that happens gradually, maybe starting with a session of port-scanning and continuing with increasing degrees of penetration. Thus, it may not be evident exactly when it happens. Further, an intrusion is not always a single event, but the combined effect of two or several events that cooperate.

IV. IMPLICATIONS OF THE MODEL

A. The causality perspective

A benefit of the model is that it clearly exhibits the causal chain of impairments, from attack to system failure. The attack is launched by a threat agent. If successful, there is an intrusion, which produces an unwanted system state, i.e., an error. There are three different outcomes of the system error. First, it may be immediately removed by some recovery mechanism. Second, it may be latent in the system for some time, before it propagates to the output. The latency time may be short. It may also last for very long time periods, e.g., many years, whether for operational reasons or because this was the intention of the attacker [1]. Third, the error may propagate through the system without any noticeable delay and directly cause a system failure.

The above reasoning shows how an attack may cause an error that propagates to cause a failure. On the other hand, it also shows that a successful attack may cause an error but that this error will not lead to a failure, i.e., it will not affect the system service. Therefore, insufficient integrity could lead to a behavioural failure, whether reflected in reduced reliability, availability, safety or confidentiality. Thus, the service delivered may be impaired by attacks on the system, but the relation between the attacks and the service is complicated and dependent on system internal factors among other things.

In another situation, the service delivered by the system may fail as a result of some (apparently) random error within the system, e.g., a component failure.

In summary, a system failure may be caused by an attack, but may also be due to some random event. Or,

 ISBN: 978-1-61208-298-1

taking the opposite view, a successful attack may or may not lead to a failure. If it leads to a failure, there may be considerable delay between the attack and the resulting failure.

B. Three basic methods to avoid failures ("defence lines")

Considering the above causal relation between impairments, we can see that there are three basic ways to break the causal chain of unwanted events and to counter the propagation of impairments; see Figure 1. The basic causal chain is attack - error - failure. We observe that the attack, i.e., external fault, originates from the environmental threat. The error is the result of insufficient protection against the attack. Finally, the failure occurs since the error was permitted to propagate to the system output. The obvious conclusion is that defence methods could be applied accordingly. We name them ***threat reduction***, ***boundary protection*** and ***recovery***. Threat reduction methods focus on the threat. These methods aim to reduce or eliminate the threat, i.e., make it less probable that an attack is launched towards the system. An example of threat reduction would be legal measures. If the threat agent is a human attacker the prospect of facing a jail sentence would most probably decrease her motivation to launch attacks as compared to if the act was legal.

Boundary protection is the set of methods that protect the system from malicious external influence. An example would be authentication, which aims at refusing access for unauthorized entities.

Recovery methods aim at eliminating errors inside system boundaries before they produce a failure. For internal faults this is the only available defence methodology. An anti-virus tool is an obvious example of a recovery mechanism. A virus that has entered the system represents an error. It is well known that many viruses will not become visible to the USER until at some later occasion. If they can be found and deleted before they have caused a failure, a successful recovery has taken place.

In order to counter an attack, i.e., to avoid a system failure, only one of these methods needs to be effective. On the other hand careful security work requires that all three types of methods are used and are continuously active.

V. SECURITY METRICS BASED ON THE SYSTEM MODEL

A. Previous Research on Security Metrication

There have been several previous attempts to present various frameworks and directions in the security metrication research field. The first comprehensive attempt towards structuring the security measurement and metrication research was carried out at the WIISSR workshop [27]. A generic concept for Information-Security *, denoted (IS)* was defined in the workshop to avoid confusion in terminology. IS* was intended to cover all different terms in the area, e.g., metric, measure, score, rating, rank, or assessment result. A significant outcome of the workshop was its proposal for the three main tracks for security metrication, i.e., *Technical, Organizational and Operational* metrics. Following this proposal, other researchers tried to add more categories with respect to various metrication applications, objectives and goals. Vaughn et al. [17] proposed two main categories for Information Assurance measurement: *Technical Target of Assessment* and *Organizational security*. From the Organizational perspective, NIST 800-26 [21] and Savola [3] proposed three main tracks for security metrication: *Technical, Operational* and *Management*. NIST 800-55 [19, 20] offered another categorization for metrication suggesting Implementation, Effectiveness and Efficiency as well as Business Impact as the main metrication categories. Other well-known security metrics approaches have been suggested by Savola [4], Pironti [7], CISWG [10] and NISTIR 7564 [22], ISO/IEC 27004 [26] and Payne [30], each of them for different systems and applications. There have not been many attempts to model-based security metrication. However, Savola [5] proposed a Security Metrics Objective Segment model, which is a taxonomy model including five levels for the main security metrics objective segments.

B. Different Approaches to the Security Metrication Process

The process to find a metric for a concept such as security involves several steps. First, you must define the concept that you intend to metricate, i.e., you make a model of it. Second, you must decide which logical attributes of the model that could serve as carriers for the metric that you are interested of. Third, you must select a suitable method for assessing the "magnitude" of these attributes. Such a method could very often be based on some tangible feature of the system, such as a protection mechanism or a vulnerability. Finally, you must find a way to carry out the metrication in a practical way. Practical ways may involve data gathering, electrical measurement or inquiries.

The discussion in this paper mainly covers the two first steps above. However, for the integrity attribute we also have suggestion for the following steps.

C. Metrication of Security and Dependability Based on Protective and Behavioural Attributes

In our approach, metrication is based on the attributes defined in the system model presented in section III. The attributes suitable for metrication are those defined according to the suggested two types of system-environment interaction, i.e., input from the environment and output to the environment. Thus, it should be possible to define ***protective metrics*** and ***behavioural metrics***, related to the system input and output respectively.

 ISBN: 978-1-61208-298-1

D. Protective Security Metrics

1) Protective security is integrity

Protective metrics should assess the extent to which the system is able to protect itself against unwanted external influence, e.g., external attacks. Normally, we assume that there is some kind of malicious intent involved in this influence, but you could also think of situations when the unwanted input is the result of e.g., a mistake made by an "ordinary" user. According to our system model, it is the ***integrity*** attribute, that embodies ***(protective) security*** and in our opinion it is the integrity attribute that captures the essence of security.

2) Protective metrics based on protection mechanisms

There may be several ways to measure the protective ability. One way could be based upon the strength of the (protective) security mechanisms of the system, under the assumption that the stronger the mechanisms are, the better the system is protected. In this situation, the measure would be based on the combined strength of all involved security mechanisms. For example, the ISO 27004 standard assesses the effectiveness of the implemented information security controls [26]. The problem with this approach is that the security (i.e., integrity) will not necessarily be higher if stronger mechanisms are involved. This is due to the fact that the protective strength rather lies in the fact that there are no weak mechanisms. Or in other words, there should be no vulnerabilities or "holes" in the system. However, it is a non-trivial task to find a method for such a combination of the effect of a number of protective mechanisms.

A similar approach is to base the metric on the three fundamental defence methods ("defence lines") described in IV.B: threat reduction, boundary protection and recovery. We realize that there are available mechanisms for the defence against intrusions for each of these methods and in this case the metric would assess the combined strength of the corresponding mechanisms.

3) Using attacker effort as a protective security metric

Another way could be to base the metric upon the *effort* that has to be expended by an attacker in order to make a breach into the system (i.e., compromise integrity). This idea was first proposed by Littlewood et al. [35] and their work has been extended in [16, 23, 36]. The idea is that an effort-based metric should be representative of all environment factors having effect on the attacker's ability to make a successful intrusion. The main contributing factors of effort are the *time* it takes to carry out the attack and the *skill level* of the attacker. However, many other parameters have to be considered: population of attackers, attack space size, reward effect on attackers' behaviour, system feedback to the attacker, attackers' willingness, etc.

4) How to find an effort metric in practice

In the above section, we discussed which environmental parameters that an effort metric should reflect and in particular the attacker behaviour. However, it is probably infeasible to really measure all those parameters in practice. Instead we have to rely upon representative samples. An attempt to make a real measurement by performing supervised attack experiments was reported in [16, 28]. This work showed that it is in principle possible to find a metric for effort. In this simplified case, the metric was Mean Time To Intrusion (MTTI), or Mean Time To Compromise, i.e., the average time used by an attacker to make an intrusion. It was also shown that, given certain pre-conditions the MTTI metric could be combined with a MTTF metric derived from random errors, such as component errors. However, the practical metric from such a single experiment has limited applicability and does only reflect the security of the used system at the time of measurement. It remains to be demonstrated how to make measurements that are generally applicable and could serve to make predictions of the security of other similar systems.

E. Behavioural Security Metrics

As suggested by the model, the behavioural security attributes (or more accurately: security and dependability related attributes) are: reliability, availability, safety, confidentiality and exclusivity. There are already a large number of metrics suggested for reliability, availability and safety and they could readily be incorporated into the framework. Confidentiality and exclusivity metrics are less well investigated. Below we shortly describe existing or proposed metrics for behavioural security attributes.

Reliability is the expected time duration the system is operating before it fails in delivering its service. The common metric for this is Mean-Time-to-Failure (MTTF).

Availability measures to which degree, often expressed in percent, the system is capable of delivering its service taken into account the alternation of service delivery and non-delivery [22].

Safety evaluates the absence of catastrophic consequences on the USERs and the environment in case of a failure [22]. A common metric for safety is Mean Time to Catastrophic Failure (MTTCF) and it is defined in analogy with Mean Time To Failure.

Confidentiality quantifies the ability of the system to keep sensitive information confidential with respect to NON-USERs.

One of the approaches to confidentiality metrication is to derive behavioural measures from traditional reliability methods, such as Markov modelling. Jonsson et al. [13] proposed performance measures on user-specified service levels. They discussed that certain levels could be related to confidentiality degradation or confidentiality failures. Hence, Mean Time To Degradation was suggested both as a reliability metric (w.r.t the USERs) and a confidentiality metric (w.r.t. NON-USERs). We proposed a vectorized measure reflecting the status of the service levels defined for the system. Other approaches to confidentiality metrication are found in [38, 24].

 ISBN: 978-1-61208-298-1

The concept of ***exclusivity*** is not widely used and we know of no suggestions for how to measure it. However, it seems plausible that an approach similar to that of confidentiality could be adopted.

VI. THE RELATION BETWEEN PROTECTIVE AND BEHAVIOURAL METRICS

A. *Implication of the chain of impairments on behavioural metrics*

In "The causality perspective", Section (IV.A), we identified a causal chain of impairments from the attack phase to the system failure. In this section, we discuss the effect of the chain of impairments in security metrication.

We realize that the behavioural attributes of the system are dependent upon the environmental threats, protection mechanisms and the internal recovery mechanisms. The stronger a threat reduction mechanism is, the less becomes the threat towards the system and consequently the number and/or strength of potential attacks. Further, the better a boundary protection mechanism is, and the higher the integrity is, the lesser would the number of errors in the system be. Finally, the better a recovery mechanism is, the less probable is a system failure. As a conclusion, the behavioural attributes (and metrics), depend on the strength of the three defence lines in the system in such a way that a better defence will lead to increased reliability. Thus, the better the defence mechanisms are, the higher becomes the reliability of the system. In conclusion, higher integrity will lead to higher reliability and the integrity metric will potentially affect the reliability metric as well as metrics for all other behavioural attributes.

B. *Implications of latency on behavioural metrics*

In the preceding section, we noted that there is a coupling between protective and behavioural attributes (mechanisms, metrics). In this section, we will deal with the latency aspect. Error latency is the delay between the introduction of an error into the system, as a consequence of an intrusion, and the resulting failure. The latency is mainly a function of system operation and/or of recovery mechanisms. The latency may be short or long. In the case of infinite latency there will be no failure and the system behaviour will never be affected. Now, by applying the same reasoning as in the previous section we realize that latency will also affect behavioural attributes and metrics. The longer the error latency, the better is the system behaviour, i.e., the better the reliability, etc. The conclusion of this is that integrity "failures" are related to behavioural failures, but that there is no deterministic correspondence.

VII. DISCUSSION

It is well known that security is a multi-faceted and complex concept. Further, there are several definitions of security, in the sense of which attributes should be included, on top of the traditional "CIA" ones. Some of these attributes may also be in contradiction to each other, for example integrity vs availability. Despite these facts many (if not most) authors suggest metrics for security without making a proper definition of it. We believe that the advantage of our approach is that it suggests a *model* of the integrated security-dependability meta-concept, in which it is split into a number of attributes. Our message is that metrication must focus on these attributes and that metrication of the meta-concept is not feasible or even possible. Thus, we have defined these attributes and the relation between them. There are several advantages with this approach: 1) It clarifies the relation between security "failures" and system (behvioural) failures. A security failure does not necessarily affect the service delivered. If it does indeed lead to a failure, this may take considerable time. 2) It becomes clear how preventive and protective actions, as well as recovery, may have beneficial consequences on the behavioural attributes. 3) The distinction between safety and security (integrity), which is sometimes an issue of controversy, becomes well defined. And in all of the three cases mentioned above there is an implication for the related metrics. For example, it clarifies why increased security leads to better safety. This is typically applicable for the "connected car", i.e., for virtually all modern cars. Another example, which shows that the model is very general can be taken from social sciences: by addressing problems with young people in metropolitan problem areas, we can mitigate criminality and its consequences many years later.

VIII. CONCLUSION

We have described an approach for the meta-concept of security and dependability. The approach is based on a system model that re-groups its attributes into protective ("input") and behavioural ("output") ones. We have outlined how metrics could be defined in accordance: protective metrics and behavioural metrics. There are already some metrics for behavioural attributes, but less so for the protective attribute, integrity. We have argued that the integrity attribute captures the essence of security and could indeed serve as a definition of security, in a restricted sense. We have outlined two methods for metricating security and shown how behavioural metrics depend on security metrics.

REFERENCES

[1] E. N. Adams, "Optimizing preventive service of software products", IBM Journal of Research and Development, vol. 28, no. 1, pp. 2-14, 1984.

[2] A. Avizienis, J-C. Laprie, B. Randell and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing", IEEE Transactions on Dependable and Secure Computing, Vol.1, No.1, Jan-Mar 2004, pp. 11-33.

[3] R. Savola, "Towards a taxonomy for information security metrics", In Proceedings of the ACM workshop on Quality of protection (QoP '07), pp. 28-30.

 ISBN: 978-1-61208-298-1

[4] R. Savola, “A novel security metrics taxonomy for R&D organisations”, ISSA 2008, July 2008, Johannesburg, South Africa, pp. 1-12.

[5] R. Savola, “A security metrics taxonomization model for software-intensive systems”, Journal of Information Processing Systems, vol. 5, No. 4, 2009, pp. 197-206.

[6] Common Criteria. ISO/IEC 15408-1, Information Technology - Security Techniques - Evaluation Criteria for IT Security, Part 1: Introduction and General Model, 1999.

[7] J. P. Pironti, “Information security governance: Motivations, benefits and outcomes”, Information Systems Control Journal, vol. 4, 2006, pp. 45-48.

[8] Information Technology Security Evaluation Criteria (ITSEC): Provisional Harmonized Criteria, December 1993. ISBN 92-826-7024-4.

[9] C. Irvine, T. Levin, “Toward a taxonomy and costing method for security services”, Computer Security Applications Conference, 1999. (ACSAC ’99) Proceedings. 15th Annual, 6-10 Dec. 1999, pp.183 – 188.

[10] CISWG, “Report of the best practices and metrics teams” (Revised), Government Reform Committee, United States House of Representatives, 2005.

[11] Y. Asnar and P. Giorgini, “Uncertainty dimensions of risks in secure and dependable domain”, Technical Report # DISI-08-058, Univ. of Trento, 2008.

[12] E. Jonsson, M. Andersson, S. Asmussen, “A practical dependability measure for degradable computer systems with non-degradation”, Proc. IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes, SAFEPROCESS’94, Espoo, Finland, 1994, pp. 231-237.

[13] E. Jonsson, M. Andersson, S. Asmussen, “An attempt to quantitative modeling of behavioural security”, Proc. 11th International Information Security Conference, Cape Town, Sout Africa, May 1995 (IFIP/SEC’95).

[14] E. Jonsson, L. Strömberg, S. Lindskog, “On the functional relation between security and dependbility impairments”, ACM New Security Paradigms Workshop, Caledon Hills, Canada, 23- 25, September 1999 (NSPW 1999), pp. 104-111.

[15] E. Jonsson, "Towards an integrated conceptual model of security and dependability," The First International Conference on Availability, Reliability and Security, (ARES 2006), 20-22 April 2006, pp. 8-16.

[16] E. Jonsson, T. Olovsson, "A quantitative model of the security intrusion process based on attacker behavior", Software Engineering, IEEE Transactions on , vol.23, no.4, pp.235-245, Apr 1997.

[17] R. Vaughn, R. Henning,A. Siraj., “Information assurance measures and metrics - state of practice and proposed taxonomy,” Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS 2003), Vol. 9. IEEE Computer Society, Washington, DC, USA, pp. 10-16.

[18] C. Meadows, “An outline of a taxonomy of computer security research and development”, New Security Paradigms Workshop, (NSPW 1993). ACM O-89791-635-2, pp. 33-35.

[19] NIST 800-55 Rev1, E. Chew, M. Swanson, K. Stine, N. Bartol, A. Brown, W. Robinson, “Performance measurement guide for information security”, National Institute of Standards and Technology Special Publication #800-55-rev1, 2008.

[20] NIST 800-55, M. Swanson, B. Nadya, J. Sabato, J. Hash, L. Graffo, “Security Metrics Guide for Information Technology Systems”, National Institute of Standards and Technology Special Publication #800-55, 2003.

[21] NIST 800-26, Swanson M., “Security Metrics Guide for Information Technology Systems”, National Institute of Standards and Technology Special Publication #800-26, November 2001.

[22] NISTIR 7564, W. Jansen, “Directions in security metrics research”, National Institute of Standards and Technology, April 2009.

[23] S. Brocklehurst, B. Littlewood, T. Olovsson and E. Jonsson, “On measurement of operational security”, Aerospace and Electronic Systems Magazine, IEEE, Vol. 9, Issue 10, Oct. 1994, pp. 7-16.

[24] L. Blasi, R. Savola, H. Abie and D. Rotondi, “Applicability of security metrics for adaptive security management in a universal banking hub system”, ECSA companion volume, August 23–26, 2010, Copenhagen, Denmark, pp. 197-204.

[25] S. Fenz and A. Ekelhart, “Formalizing information security knowledge”, ASIACCS’09, March 10-12, 2009, Sydney, NSW, Australia, pp. 183 – 194.

[26] ISO/IEC 27004:2009 Information technology — Security techniques — Information security management – Measurement.

[27] Workshop on Information-Security-System Rating and Ranking (ISSRR) held in Williamsburg, VA, May 21-23, 2001.

[28] U. Gustafson and E. Jonsson, “Security Evaluation of a PC Network based on Intrusion Experiments”, Proc. 14th International l Congress on Computer and Communications Security, SECURICOM '96, Paris, France, pp. 187-203.

[29] A. Hecker, “On system security metrics and the definition approaches”, The Second International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2008, 25-31 Aug. 2008, pp. 412-419.

[30] S. C. Payne, “A guide to security metrics”, SANS Security Essentials.

[31] E. Jonsson and L. Pirzadeh, “A framework for security metrics based on operational system attributes”, International workshop on Security Measurements and Metrics (MetriSec 2011), Bannf, Alberta, Canada, 2011-09-21, pp. 58-65

[32] D.M. Nicol, W.H. Sanders and K.S. Trivedi, “Model-based evaluation: from dependability to security”, Dependable and Secure Computing, IEEE Transactions on , vol.1, no.1, Jan.-March 2004, pp. 48- 65.

[33] K.S. Trivedi, D. S. Kim, A. Roy and D. Meedhi, “Dependability and security models”, Proceedings of 7th International Workshop on the Design of Reliable Communication Networks (DRCN 2009), Washington, DC, October 2009, pp. 11-20.

[34] J. F. Meyer, “Performability: A retrospective and some pointers to the future”, Performance Evaluation, Vol. 14, No. 3-4, Elsevier, 1992, p. 139-156.

[35] B. Littlewood et al, “Towards operational measures of computer security”, Journal of Computer Security, Vol.2, 1993, pp.211-229.

[36] T. Olovsson, E. Jonsson, S. Brocklehurst, and B. Littlewood, “Towards operational measures of computer security: Experimentation and modelling”, Predictably Dependable Computing Systems, B. Randell et al., eds., ISBN 3-540-59334-9, Springer-Verlag, 1995, pp. 555-572.

[37] E. Jonsson, “An integrated framework for security and dependability”. Proceedings of the New Security Paradigms Workshop 1998, Charlottesville, VA, USA, September 22-25, 1998, pp. 22-29.

[38] R. Savola and H. Abie, ”Development of measurable security for a distributed messaging system”, International Journal on Advances in Security, vol. 2, no. 4, 2009, pp. 358-380.

[39] D. G. Firesmith, “Common Concepts Underlying Safety, Security, and Survivability Engineering”, CMU/SEI-2003-TN-033, Carnegie-Mellon University, Pittsburg, USA.

Risk-based Dynamic Access Control for a Highly Scalable Cloud Federation

Daniel Ricardo dos Santos, Carla Merkle Westphall, Carlos Becker Westphall
Networks and Management Laboratory
Federal University of Santa Catarina
Florianópolis, Brazil
{danielrs, carlamw, westphal}@inf.ufsc.br

Abstract— **Cloud Computing is already a successful paradigm for distributed computing and is still growing in popularity. However, many problems still linger in the application of this model and some new ideas are emerging to help leverage its features even further. One of these ideas is the cloud federation, which is a way of aggregating different clouds to enable the sharing of resources and increase scalability and availability. One of the great challenges in the deployment of cloud federations is Identity and Access Management. This issue is usually solved by the creation of identity federations, but this approach is not optimal. In this paper, we propose an access control system for a highly scalable cloud federation. The presented system is dynamic and risk-based, allowing the use of cloud federations without the need of identity federations. We also present results of a prototype implementation and show that it is scalable and flexible enough to meet the requirements of this highly dynamic and heterogeneous environment.**

Keywords- cloud computing; access control; risk; cloud federation

I. INTRODUCTION

Cloud computing is a model for enabling on-demand network access to a shared pool of computing resources [1]. It is widely adopted and provides advantages for customers and service providers.

As cloud computing grows in popularity, new ideas and models are developed to exploit even further its full capacity, increasing efficiency and scalability. One of these ideas is the deployment of cloud federations [2, 3]. A cloud federation is an association among different Cloud Service Providers (CSPs) with the goal of sharing data and resources [4].

However, to make such a scenario feasible it is necessary to develop authentication and authorization models for largely distributed, dynamic and heterogeneous environments.

This problem is usually treated by the deployment of identity federations. An identity federation is a model of identity management where identity providers and service providers share users' identities inside a circle of trust [32].

This solution, nevertheless, is not optimal, since identity federations present problems such as the necessity of attribute and trust agreements, interoperability issues and, in practice, show limited scalability [5]. This paper shows that it is possible to provide authorization in cloud federations without the need for an identity federation.

The difference between cloud federations and identity federations is that cloud federations are built to share resources and identity federations are built to share users and identity information.

In this paper, we propose to use a risk-based dynamic access control to enable authorization in a cloud federation without the necessity, but allowing the possibility, of using identity federations.

The rest of the paper is organized as follows: Section II presents the related work; Section III discusses the concept of cloud federations; Section IV analyses dynamic access control; Section V presents our proposal; Section VI shows some results and Section VII is the conclusion.

II. RELATED WORK

There are two main kinds of work which are related to this paper: those which study cloud federations and authorization in these scenarios and those which propose dynamic access control models.

CLEVER Clouds [6, 7, 8] is a "horizontal federation" model, built on top of a component called Cross Cloud Federation Manager (CCFM), responsible for the discovery of clouds in the federation, finding the best match for resource requests and handling authentication. Based on this architecture, there is the proposal of using federated identity management with a third party identity provider to handle authentication and authorization [9].

The Contrail project [10] is a framework for the construction of cloud federations. It is built upon a set of core components: the Virtual Execution Platform (VEP), the XtreemFS and the Cloud Federation. Contrail is a big project funded by the European Union and is under active development. It also uses federated identity management and provides support for eXtensible Access Control Markup Language (XACML) authorization and the Usage Control (UCON) access control model.

A basic blueprint for the Intercloud is presented in [11] and [12]. In those papers the aggregate of clouds is envisioned based on an architecture comprised of an Intercloud Root, responsible for naming and trust; Intercloud Gateways, responsible for enabling communication between protocols and standards; and finally the clouds. These papers propose that trust be managed by the Intercloud Root, in a configuration that is similar to an identity federation.

Some challenges for access control in highly distributed environments are presented in [13], which compares the Attribute-based Access Control (ABAC), UCON and Risk-adaptive Access Control (RAdAC) models.

 ISBN: 978-1-61208-298-1

The idea of using risk-based access control in cloud computing is presented in [14], where the authors claim this model is adequate to solve the problems presented by multi-tenancy and also that a dynamic environment requires a dynamic access control model. The paper presents a scenario where RAdAC is used to enforce access control among the tenants of a cloud, considering the risks of illegal access to tenants' data by other tenants or by administrators. The paper, however, shows only an overview of the proposal and lacks validation.

Arias et al. [15] proposed a set of metrics, organized in a taxonomy, to be used in the establishment of identity federations in the cloud and to handle access requests. The authors claim that the federated identity management model is hindered by the underlying trust models that must be pre-established, and that the use of risk metrics can mitigate this problem.

The main difference between our approach and the related work is the use of a risk-based access control model to enable the deployment of cloud federations without the need for identity federations. This proposal is detailed in Section V, and a deeper comparison to the related work can be found in the conclusions.

III. CLOUD FEDERATIONS

The cloud computing paradigm has reached a relative success due to its well-known advantages in scalability and cost reduction, but to enable its full potential we must step forward towards cloud federations [16].

As seen in Section II, there are several proposals of architectures for cloud federations in the literature, but they all share a common goal of aggregating different clouds through standard protocols, enabling their interaction and the sharing of resources available in each one. Cloud federation comprises services from different providers aggregated in a single pool supporting resource migration, resource redundancy and combination of complementary resources or services [4].

The main benefits of this new approach are an increase in scalability, availability and interoperability. It also helps in reducing costs of single providers, since the workload may be shared among the members of the federation.

Thinking even further, there are already proposals for an Intercloud, which is a global aggregate of clouds, such as the Internet is a global aggregate of networks [11, 17].

The establishment of cloud federations presents challenges such as the definition of standard protocols and the migration of virtual resources among diverse providers, but the focus of this work is in the security aspects of the federations, especially Identity and Access Management.

Cloud security is a challenge, since providing availability, integrity and confidentiality for a huge number of users and resources in an Internet-accessible environment is not easy. Cloud federations tend to increase concerns because of the increase in the number of users and resources, the use of different protocols and the exchange of sensitive data among providers. Issues such as governance, auditing and risk management are being actively researched for clouds [18]; these studies must be extended to understand the influences of a cloud federation.

Compliance to regulations and the definition and fulfillment of Security Service Level Agreements (SecSLAs) are also indispensable.

One of the most important issues in the establishment and running of a cloud federation is Identity and Access Management (IAM) [28].

When in a single cloud, it is possible to use traditional IAM procedures and authorization models to handle access control because all of the users and resources are within the same security domain. When resources and subjects are scaled to a federation of clouds, nevertheless, there is the concern with the fact that subjects may come from a different security domain than the resource to which they are requesting access.

To implement authorization using models such as Role-Based Access Control (RBAC) or Attribute-Based Access Control (ABAC), the cloud must use information provided by a system about a user. This information may be, for instance, the user's identity or attributes of this identity, such as name, organizational role and date of birth.

For a cloud to trust the identity or attribute information of a user that comes from another cloud, both clouds must share some agreement of trust. That is why this process is commonly mediated by an identity federation. With Federated Identity Management (FIM), every participant of the federation is expected to agree that the information received by another participant is correct, in what is called a Circle of Trust (CoT).

A problem with this approach is the fact that this agreement requires previous negotiation, which may be an extensive process and hinder dynamic collaboration. Dynamic collaboration is achieved when entities which have a need to collaborate can instantly form a federation, without the need for a previous trust agreement.

Another problem faced by identity federations is the extensive number of protocols and standards, which actually reduces interoperability. Federations tend to get bigger and bigger and users may participate in different federations. All of those facts combined lead to, in practice, a limited scalability of identity federations, reducing their effectiveness in real world.

But even with the formation of identity federations and the possibility of using ABAC, there are challenges to be considered. The static policies which are predefined to be used in traditional access control models cannot comprehend every possible access situation, because in the cloud this is an ever changing process, with users and resources being deployed and deleted all the time. Static models, thus, lack the flexibility necessary to support exceptional situations, which are common in military and medical applications, among others [19] and important for collaboration and information sharing [20]. Examples of these exceptional access requests are given in the next section and are abundant in the literature.

 ISBN: 978-1-61208-298-1

IV. DYNAMIC ACCESS CONTROL

Identity and access management encompasses several processes related to the identification, authentication, authorization and accountability of users in computer systems [21]. Authorization or access control is the process through which a system guarantees that access requests are validated using well-established rules. These rules are known as policies and the way through which the policies are enforced together with the mechanisms used in this enforcement is known as an access control model.

Classical access control models are known to present problems in highly distributed and dynamic environments [13], especially scalability and flexibility limitations and the use of static policies [14]. Role-based models, for instance, lack granularity of control, because roles share their permissions with every user they are attributed to.

To enable more flexible access control decisions, which reflect current needs for information sharing and allow for a secure handling of exceptional requests, dynamic access control models were developed [22, 26, 29, 33].

In contrast with classical models, dynamic access control has the characteristic of using more than predefined policies to compute access decisions. These models are based on dynamic characteristics, which are assessed in "real time" as the subject requests access to a resource. Characteristics such as trust, context, history and risk are often used to reach decisions, and exactly which characteristics to use and how to measure them is discussed in several works [23, 24, 25, 26].

Risk-based access control models are often used as a "break-the-glass" mechanism, allowing for exceptional access requests to be handled by the system more effectively than simply granting full access [13, 30].

Exceptional requests and special access are sometimes necessary in medical and military applications, among others. A well-known example is in a healthcare facility where only doctors have access to patients' histories, but in the case of an emergency, a nurse may need to access this information to save a patient's life. If this kind of situation was not predicted in any policy, either the nurse won't be able to perform his/her duty or the nurse may be given a doctor's access, which may grant a broader access than the necessary in this case, allowing misuse. In either case, it represents a greater risk to the system than if a dynamic access control system were used and the access control needs were evaluated per request.

Granting special access in exceptional cases usually involves some form of monitoring by the system. It may be in the form of: obligations, which are post-conditions that a user must fulfill in order to keep his or her access right [13]; a reputation system, which logs users' actions and assigns rewards and penalties to them [26]; or a market system, in which users have a limited amount of points that may be used to "buy" exceptional accesses [27].

Supporting this kind of access control involves an effective logging system for posterior audit and incident response.

There are several different approaches to risk-based access control, but they all share some common features. Fig. 1 presents an overview of a risk-based access control model.

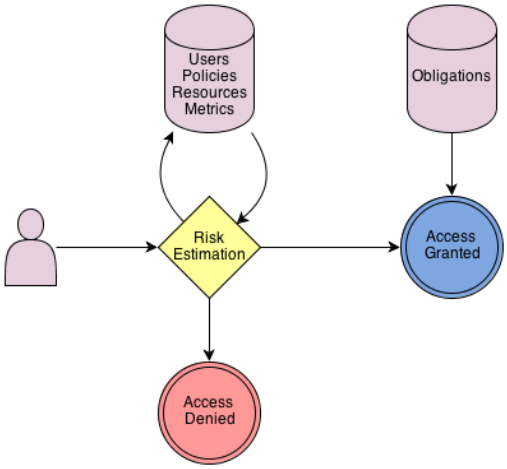


Figure 1. Risk-based access control overview

The figure is based on common points found in diverse models, and the main elements present are the subject, the resource and the risk estimation engine.

The subject tries to access a resource by issuing an access request, which is then processed by a risk estimation engine that uses all the information it deems necessary to come to a decision. Usually there is a risk threshold defined by the system administrators, and if the risk is lower than this threshold, access is granted. Other variations measure risk versus benefit of an access, and decide based on which one is greater [31].

V. PROPOSAL

In this paper, we propose that it is possible to provide a way to establish cloud federations without the need for identity federations, by using risk-based access control and relying on the authentication provided by each cloud. This can increase the scalability of this model and handle exceptional requests.

A. *Cloud Federations*

Fig. 2 presents an overview of the cloud federation architecture that we are considering. This architecture is based on the common points found in the main federation projects currently being developed, some of which were described in Section II.

The main application scenarios for such federations are medical, military and scientific collaborations, which require large storage and processing capabilities, as well as efficient information sharing. In this architecture we have the following components:

CloudProvider: this is the Cloud Service Provider (CSP) itself, who provides the infrastructure over which the virtual resources are allocated (they are represented by the clouds in the figure);

CloudManager: responsible for attaching a CloudProvider to the federation. It is composed of several services that deal with users, resources, policies, service-

 ISBN: 978-1-61208-298-1

level agreements, security and the CloudProvider. It is modular so that it can be attached to different cloud management software just by changing one of its services.

FederationManager: responsible for coordinating the federation. It acts as a naming service and is also responsible for message passing.

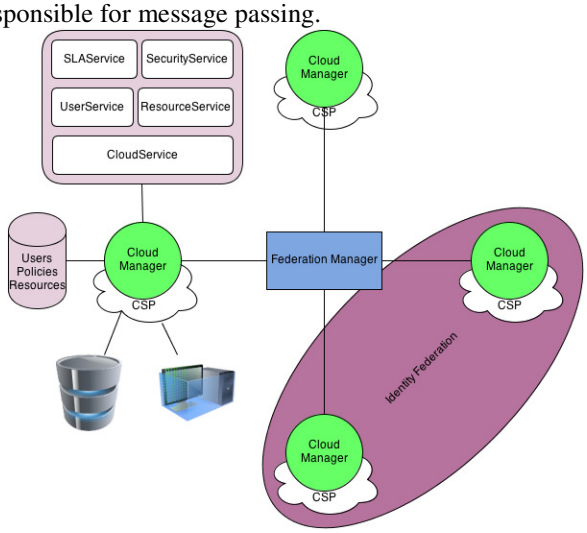


Figure 2. Overview of the federation

B. *Access Control*

As shown in Fig. 2, some of the participating clouds may form identity federations among themselves.

Under the point of view of a user there are two types of clouds in this architecture: a home cloud (the user's original CSP) and foreign clouds (the other clouds in the federation). Users can deploy and access resources in both types of cloud, but access control behaves differently for each case.

When users deploy a resource in their home cloud they may choose if it will be available for users of foreign clouds. In any case the user must upload an XACML policy file together with the resource, which will be used for ABAC.

Users may also deploy resources in a foreign cloud and it will automatically be available to every user of the federation. Finally, users may access resources in their home cloud or shared resources in foreign clouds.

When a user tries to access a resource in their home cloud, this request is handled by a classical ABAC model. Based on user attributes and XACML policies the system grants or denies the requested access.

When a user tries to access a resource in a foreign cloud, the system first verifies if both clouds are in an identity federation, in which case the access will also be handled by ABAC, but if there is not an identity federation between them, the "break-the-glass" mechanism is activated and the risk-based access control Policy Decision Point (PDP) is called.

The PDP is located in the cloud handling the access request (foreign to the requester) and the metrics and parameters of risk estimation are defined by the administrators of this cloud and the users who own the resources.

These metrics are informed in an eXtensible Markup Language (XML) file, containing definitions of risk metrics and how to measure and aggregate them, as well as a threshold level for granting access to the resource and possible obligations that users will have to follow. This file is known as a risk policy.

Each cloud provider must provide a set of basic metrics with their quantification rules. Those will be used to create a baseline risk policy for the provider. This guarantees that a cloud provider is able to maintain their minimal security requirements.

Each resource has its own risk policy, which must respect what is defined in the baseline policy, but may be extended to become more or less restrictive as the user desires. The XML file of the policy must be uploaded by users when they choose to deploy a shared resource. The system does not generate risk policies on the fly and all the risk policies must follow a predefined XML schema, so that different clouds can communicate.

If a user chooses to define a risk metric that is not available in the server, he/she must provide a way for the CSP to quantify this risk. This is done by defining a Web Service that will be called by the PDP upon the evaluation of the access request. The PDP will forward the access request to the Web Service, which will have to parse it, process it and return a numeric value representing the associated risk for the metric being evaluated.

To handle the access request for a given resource all of the metrics are valued, based on the rules defined by the CSP and the Web Services defined by the user. The chosen aggregation engine is used to reach a final risk value. This value is then compared to the defined threshold and, if lower, the subject is given special access.

Before granting access, however, the policy is analyzed in search of obligations that were defined by the user. Those obligations are stored in a system monitor, which will watch and log every user action once the access is granted.

Fig. 3 shows an example of a risk policy file. In this example a metric for transport layer encryption will be quantified, along with other metrics. They will be aggregated based on a maximum value rule. If the final value is lower than 10, access will be granted.

```
<risk-ac>
    <resource id="1"/>
    <user id="2"/>
    <metric-set name="transport layer">
        <metric>
            <name>Transport Layer Encryption</name>
            <description>Quantifies the strength of the encryption scheme
            used in the access request</description>
            <quantification>https://example.com/quantify-tl-encryption
            </quantification>
        </metric>
    </metric-set>
    <aggregation-engine>maximum_value<aggregation-engine>
    <risk-threshold>10</risk-threshold>
</risk-ac>
```

Figure 3. XML risk policy example

Fig. 4 presents a step-by-step flow of the handling of an access request in a foreign cloud. In this figure, step 1 is the issuing of an access request from a user to a foreign shared

resource (since resources that are not shared are not visible to foreign users). The Policy Enforcement Point (PEP) receives this request and forwards it to a PDP (step 2). The PDP verifies if the user's home cloud and the foreign cloud participate in the same identity federation. If they do, then the PDP requests the XACML policies applicable to the resource (step 3a), the Policy Access Point (PAP) responds to this request (step 4a), and the PDP retrieves the necessary attributes from the Policy Information Point (PIP) in steps 5a and 6a.

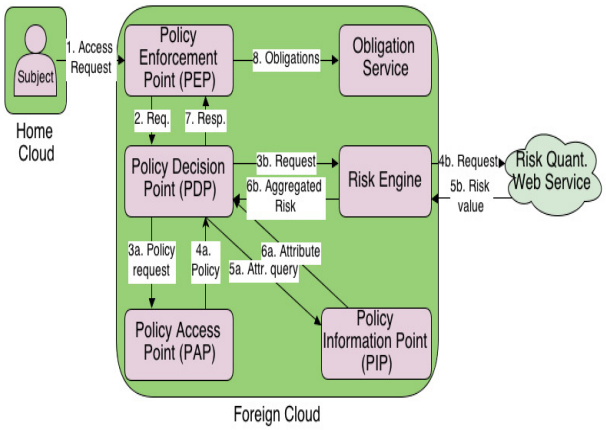


Figure 4. Access control step-by-step

Steps 3a to 6a represent a classical XACML access control decision. However, if the user's home cloud and the foreign cloud are not participants of the same identity federation, the PDP will forward the access request to a risk engine (step 3b). This risk engine will then parse the XML risk definition file associated with the resource and quantify the metrics defined. If the quantification rules are local, the predefined functions are called, if any of the rules are defined in a web service, then it is invoked, having the access request as a parameter (step 4b). The risk quantification web service performs its role and returns a risk value. After all of the metrics are valued, the risk engine applies an aggregation rule, which is always local. The aggregated risk is then returned to the PDP, which uses this value to decide upon the granting of the access request, once again based on what is defined in the XML file. After reaching a decision, the PDP returns it to the PEP, which applies the necessary obligations.

The dynamic nature of access control is present in the system because the access decision may vary according to contextual information evaluated by the metrics.

VI. RESULTS

To validate our proposal and measure some performance characteristics we implemented the key parts of the federation system and the whole access control system.

The implementation used the Python programming language, the zeromq library to handle message passing, MySQL for persistence, the ndg-xacml library for XACML evaluations and the web.py framework for the web services.

The infrastructure over which the federation was deployed was composed of two OpenNebula clouds running on a laptop with a 2,53GHz Core i5 processor and 4GB of RAM. All of the experiments were repeated 50 times to obtain the averages and all of the times shown here refer only to the execution of the access control decision function, ignoring message passing between the clouds. Table I shows four different cases of access request. Case A represents 10 requests handled by local XACML only; case B represents a risk decision that involves 10 risk quantification rules performed locally; case C uses 5 local rules and 5 external (web service) rules; and case D represents a risk policy with 10 external risk quantification rules.

TABLE I. COMPARISON OF DIFFERENT ACCESS REQUEST CASES

	min. (ms)	max. (ms)	average (ms)
A	1.057	9.372	1.46
B	1.824	15.564	4.574
C	1556.182	2813.56	1726.71
D	3247.563	10350.5	4220.6

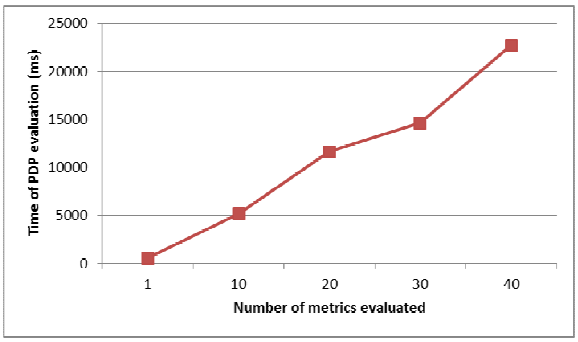


Figure 5. Performance with a varying number of external metrics

It is possible to see that the use of local risk quantification rules has no significant impact on performance, while the use of web services does affect performance, as expected, because of the HTTP invocations that must be performed for each metric.

Fig. 5 shows the growth in time spent reaching an access decision as we increase the number of metrics which call web services in a risk policy file.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a risk-based dynamic access control system to enable cloud federations without the need, but allowing the possibility, of identity federations. By eliminating the need for identity federations our proposal eases the use of cloud federations, since it doesn't depend on the establishment of agreements and circles of trust, also enhancing scalability, by avoiding the formation of "identity islands" [5].

The main contributions of this paper are the definition of a risk-based access control system for cloud federations and the proposed use of risk policies in the form of XML files to allow the use of different risk metrics and quantification methods that are not necessarily predefined.

The proposal is flexible enough to handle the needs of a cloud federation and the performance evaluations indicate

 ISBN: 978-1-61208-298-1

that it is scalable and that the risk estimation process is not a big hindrance in the process, especially if the quantification is performed locally.

In comparison to the related work we first have to clarify that we have not implemented a whole cloud federation system such as [6, 7, 8, 10], since it is a huge task and not our focus. We have, however, described and implemented a simple federation model that is sufficient for our access control research and we can highlight that our proposal is the only that uses risk-based access control. Also, we still allow the use of identity federations, but offer a choice of establishing the cloud federation without the need for Federated Identity Management.

Compared to the works that deal with risk-based access control in cloud [14, 15], our approach has the advantage of allowing the resource owner to choose different risk quantification and aggregation engines through a risk policy definition file, also the cloud that hosts the resource can define a baseline risk policy, to ensure its minimum security requirements are met.

As future work we foresee the possibility of enlarging the federation used in our experiments and deploying it to real use. Also, we want to explore further the use of the risk policies with different risk metrics and quantification methods.

REFERENCES

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing", 2011. Available at: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf [Retrieved : May, 2013]

[2] E. Carlini, M. Coppola, P. Dazzi, L. Ricci, and G. Righetti, "Cloud Federations in Contrail", Proc. Euro-Par 2011: Parallel Processing Workshops, 2012, pp. 159-168

[3] B. Rochwerger et al., "The reservoir model and architecture for open federated cloud computing", IBM J. Res. Dev., vol. 53, no. 4, July 2009, pp. 535-545

[4] T. Kurze, M. Klems, D. Bermbach, A. Lenk, S. Tai, and M. Kunze, "Cloud Federation", The Second International Conference on Cloud Computing, GRIDs, and Virtualization, September 2011, pp. 32-38

[5] K. Lampropoulos and S. Denazis, "Identity management directions in future internet", IEEE Communications Magazine, vol. 49, no. 12, December 2012, pp. 74-83

[6] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to Enhance Cloud Architectures to Enable Cross-Federation", Proc. 3rd IEEE International Conference on Cloud Computing, July 2010, pp.337-345

[7] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "Security and Cloud Computing: InterCloud Identity Management Infrastructure", 19th IEEE WETICE, June 2010, pp. 263-265

[8] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "Three-Phase Cross-Cloud Federation Model: The Cloud SSO Authentication", 2nd AFIN, July 2010, pp. 94-101

[9] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "Federation Establishment Between CLEVER Clouds Through a SAML SSO Authentication Profile", International Journal on Advances in Internet Technology, vol. 4, no. 12, 2011, pp.14-27

[10] M. Coppola et al., "The Contrail approach to cloud federations", Proc. ISGC'12 , 2012

[11] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, "Blueprint for the Intercloud - Protocols and Formats for Cloud Computing Interoperability", Proc. 4th ICIW, May 2009, pp. 328 - 336

[12] D. Bernstein and D. Vij, "Intercloud Directory and Exchange Protocol Detail Using XMPP and RDF", Proc. 6th SERVICES, July 2010, pp. 431-438

[13] V. Suhendra, "A Survey on Access Control Deployment", Proc. FGIT-SecTech, 2011, pp. 11-20

[14] D. Fall, G. Blanc, T. Okuda, Y. Kadobayashi, and S. Yamaguchi, "Toward Quantified Risk-Adaptive Access Control for Multi-tenant Cloud Computing", Proc. 6th JWIS, October 2011

[15] P. Arias-Cabarcos, F. Almenárez-Mendoza, A. Marín-López, D. Díaz-Sánchez, and R. Sánchez-Guerrero, "A Metric-Based Approach to Assess Risk for "On Cloud"' Federated Identity Management", Journal of Network and Systems Management, vol. 20, no. 4, 2012, pp. 513-533

[16] P. Harsh, Y. Jegou, R. Cascella, and C. Morin, "Contrail virtual execution platform challenges in being part of a cloud federation", Proc. 4th ServiceWave, 2011, pp.50-61

[17] R. Buyya, R. Ranjan, and R. Calheiros, "InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services", Algorithms and Architectures for Parallel Processing, vol. 6081, Springer, 2010, pp.13-31

[18] D. Catteddu and G. Hogben, "Cloud Computing: benefits, risks and recommendations for information security", Technical Report. European Network and Information Security Agency, 2009

[19] B. Farroha and D. Farroha, "Challenges of operationalizing dynamic system access control: Transitioning from ABAC to RAdAC", Proc. SysCon, March 2012, pp. 1-7

[20] F. Salim, J. F. Reid, and E. Dawson, "Towards authorisation models for secure information sharing : a survey and research agenda." The ISC International Journal of Information Security, vol. 2, 2010

[21] B. McQuaide, "Identity and Access Management", Information Systems Control Journal, vol. 4, ISACA, 2003

[22] JASON Program Office, "Horizontal Integration: Broader Access Models for Realizing Information Dominance", Technical Report. MITRE Corporation, December 2004

[23] D. W. Britton and I. A. Brown, "A Security Risk Measurement for the RAdAC Model.", Naval Postgradute School Thesis, 2007

[24] Q. Wang and H. Jin, "Quantified risk-adaptive access control for patient privacy protection in health information systems", Proc. 6th ASIACCS, 2011, pp. 406-410

[25] Y. Li, H. Sun, Z. Chen, J. Ren, and H. Luo, "Using Trust and Risk in Access Control for Grid Environment", Proc. SECTECH, 2008, pp. 13-16

[26] R. A. Shaikh, K. Adi, and L. Logrippo, "Dynamic risk-based decision methods for access control systems", Computers & Security, Elsevier, vol. 31, no. 4, 2012, pp. 447-464

[27] I. Molloy, P. Cheng, and P. Rohatgi, "Trading in risk: using markets to improve access control", Proc. NSPW, 2008, pp. 107-125

[28] D. N. Sriram, "Federated Identity Management in Intercloud", Der Technischen Universität München Thesis, January 2013

[29] R. Lepro, "Cardea: Dynamic Access Control in Distributed Systems", Technical Report, NASA, November 2003

[30] A. Ferreira et al., "How to Securely Break into RBAC: The BTG-RBAC Model", Proc. ACSAC '09, 2009, pp. 23-31

[31] L. Zhang, A. Brodsky, and S. Jajodia, "Toward information sharing: benefit and risk access control (BARAC)", Proc. 7th IEEE POLICY, 2006, pp. 9-53

[32] H. Lee, I. Jeun, and H. Jung, "Criteria for evaluating the privacy protection level of identity management services", Proc. SECURWARE, 2009, p. 155–160

[33] J. Tigli, S. Lavirotte, G. Rey, V. Hourdin, and M. Riveill, "Context-aware Authorization in Highly Dynamic Environments", International Journal of Computer Science Issues, vol. 4, no. 1, 2009, pp. 24-35

 ISBN: 978-1-61208-298-1

Towards a Policy-Framework for the Deployment and Management of Cloud Services

Tim Waizenegger, Matthias Wieland
Institute of Parallel and Distributed Systems
University of Stuttgart
Stuttgart, Germany
{waizentm, wieland}@ipvs.uni-stuttgart.de

Tobias Binz, Uwe Breitenbücher, Frank Leymann
Institute of Architecture of Application Systems
University of Stuttgart
Stuttgart, Germany
{binz, breitenbuecher, leymann}@iaas.uni-stuttgart.de

***Abstract*—As the adoption of Cloud Computing is growing, the automated deployment of cloud-based systems is becoming more and more important. New standards, such as TOSCA (OASIS), allow the modeling of interoperable Cloud services. It is now possible to build reusable and portable cloud services that can be (semi-) automatically deployed by different cloud-deployment-engines at various Cloud environments. However, there is still an acceptance problem among potential users, especially in the enterprise segment, that stems from security issues like data security. To improve security in automatic Cloud management engines, this paper proposes a framework for processing non-functional requirements of Cloud services.**

Keywords-Cloud Computing; Security; Policy-Framework; TOSCA; Cloud Service; Cloud Management

I. INTRODUCTION

According to the definition of NIST [3], Cloud Computing is a model for enabling ubiquitous, convenient, and on-demand network access to a shared pool of configurable computing resources. An important aspect is the fast deployment of Cloud Computing resources with minimal management effort. To achieve this goal, the new Cloud standard TOSCA [2] was developed allowing the portable modeling and automatic deployment and management of Cloud services. This enables the effortless migration of Cloud services across different Cloud environments. The TOSCA specification provides a domain-specific language to describe Cloud services based on different components and their relationships using a so-called Service Template (ST), which describes the topology of services. The orchestration via so-called management plans enables automated deployment and management of Cloud services. This allows the deployment of a TOSCA-based service in any Cloud environment that supports the execution of these models. In addition, TOSCA allows defining non-functional requirements of Cloud services based on so-called policy types that we use to define security requirements.

We work on implementing OpenTOSCA - an open source TOSCA container - that enables the automatic deployment of TOSCA based application models. Figure 1 shows an example for such an application. The diagram uses the visual notation Vino4TOSCA [1]. It defines a Cloud service running on two virtual machines with a separate stack for the database and web server. Both machines run the Ubuntu Linux operating system. The diagram shows the web server, PHP module and web application installed on one machine, and the database on the other.

To secure the application, we define different policies for

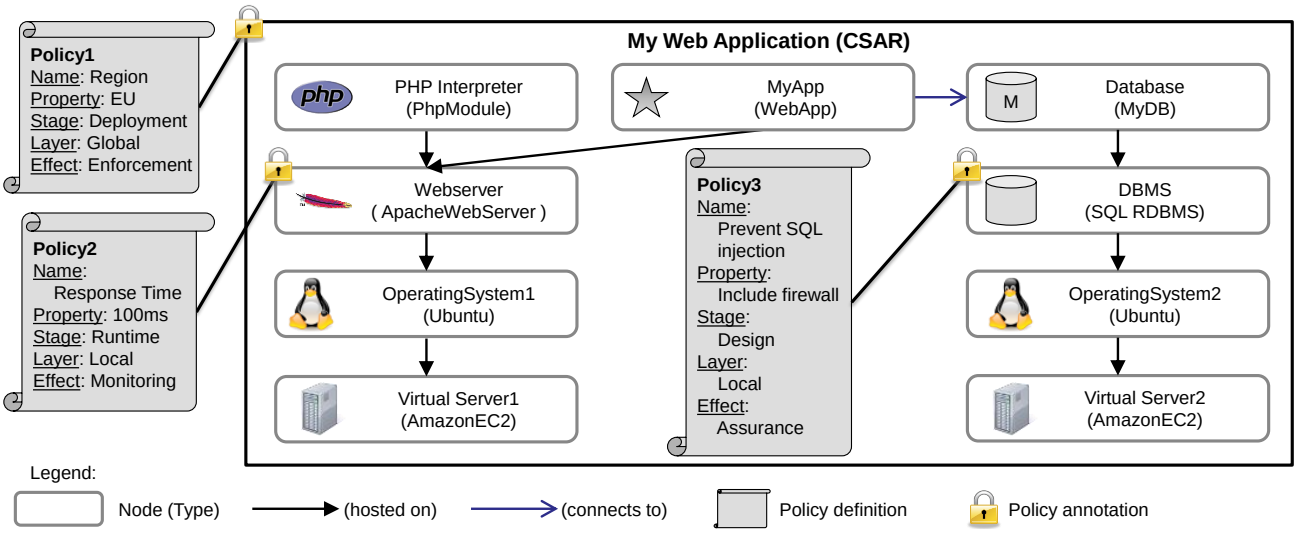


Figure 1. Example Scenario for a Cloud Service Policy in TOSCA.

 ISBN: 978-1-61208-298-1

the service model. *Policy1* requires that all components have to be deployed in a specific geographic region, due to data privacy reasons, as the data managed in the system must not leave Europe. *Policy2* specifies the maximum response time for HTTP requests on the web server in order to stay within customer requirements. When the defined response time is exceeded, the policy will deploy another instance of the web server to provide more capacity. *Policy3* is a security policy for the database system. When enabled, it adds a special application-firewall to the service topology, which provides protection against SQL-injection attacks.

OpenTOSCA does not yet support Policies; so, our next step is to make OpenTOSCA policy-aware. For that purpose, this paper presents the architecture of a Policy-Framework for security related issues in Cloud service deployment and management. The focus of the proposed Policy-Framework is the specification of non-functional requirements and their automatic processing in a Cloud service deployment engine (TOSCA container). OpenTOSCA can already deploy this application but without fulfilment of the policies. In the remainder of the paper we show how we plan to extend the system to realize the implementation of the policies.

This work takes a different approach than other publications that focus on specifying frameworks for implementing different methods to provide security features, e.g., authentication across different providers or trust management [4]. The goal of this paper is to introduce different aspects of policies and to evaluate how to use policies for secure Cloud service deployment and management with TOSCA. It is not meant to be a complete list covering all aspects of policies. We are working towards building a generic framework that supports the aspects mentioned in this paper as well as the ones emerging in future research. The definition of interoperable standard policy types should be addressed in future publications in the context of the TOSCA specification.

The remainder of this paper is structured as follows: Section II explains the ecosystem where policies are used. Section III presents a taxonomy for describing policies. Section IV describes an architecture implementing the different technical aspects of policies for Cloud service deployment. In Section V, we summarize our findings.

II. POLICY ECOSYSTEM

The source of policies, especially in the context of Cloud security, is usually a Service Level Agreement (SLA) between the Cloud provider and the customer. This SLA comprises the business perspective of the policy, whereas the specific implementation is considered the technical perspective.

The business perspective determines the business aspects of the policy such as price, penalties, and legal obligations as well as the subject.

The technical perspective is derived from the requirements given by the business perspective and describes the specific way in which the policy is implemented.

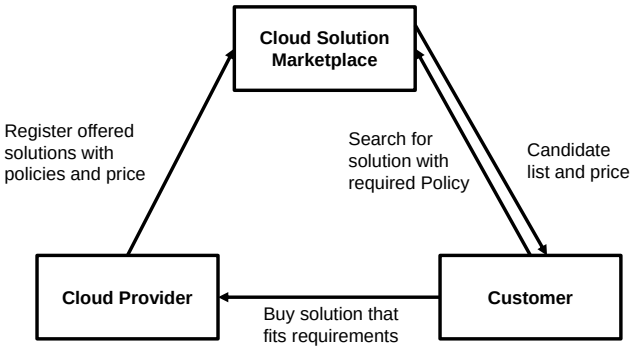


Figure 2. Actors in Ecosystem and Their Perspective on Policies.

Figure 2 shows the actors in this ecosystem. The customer requests the policy that he requires for his service. Then either a marketplace already provides the requested solution with the needed policy or the customer orders such a solution from a service provider. The customer can then buy the solution and contract a Cloud provider to deploy it while complying with the negotiated SLA and, therefore, the resulting policies.

III. TAXONOMY AND ASPECTS OF POLICIES

A policy in its most basic form is a single property that is attached to a service, a component of a service, or a relationship between components. That property is used as a parameter for determining the behavior of a system or process within the lifecycle of the service. A policy is therefore, defined by a) its *property* and b) the *aspects* that define the type of property and for which operation or system it is used during which stage of the lifecycle. We propose the following list of policy aspects as a generic model:

1) Stage in service lifecycle
   a) Design
   b) Provider selection
   c) Deployment
   d) Runtime
   e) Termination
2) Layer in topology
   a) Global policy (for whole Service Template)
   b) Local policy (for specific node or relationship type)
3) Policy effect:
   a) Assurance
   b) Enforcement
   c) Monitoring

A. The Signature of a Policy

The aspects introduced above identify a policy and comprise its signature much like the signature of a program function. The values for name, stage, layer, and effect are fixed since they are determined by the implementation of the specific policy. The policy property is variable and can be selected by the customer in order to fine-tune the behavior of the policy (see response-time policy example below) or if appropriate, it can be a fixed value as well. It should be noted that a property can be an atomic value or a complex

 ISBN: 978-1-61208-298-1

type with multiple values. Policy internal dependencies between property values have to be considered when defining the properties. Dependencies to properties of other policies should be handled by providing dependency-aware policy implementations.

B. Policy Aspects in Detail

For our Cloud policy taxonomy, we define three categories of aspects: *1) lifecycle stage*, *2) affected topology layer*, and *3) policy effect.* The following list gives a detailed description of the aspect categories.

1) Stage in Service Lifecycle

The first aspect defines the stage in the lifecycle of a Cloud service. Figure 3 shows the different stages that every Cloud service undergoes. This aspect describes at which stage the policy has to be fulfilled and implemented. However, a policy implemented at one stage might still affect the service behavior during the subsequent stages.

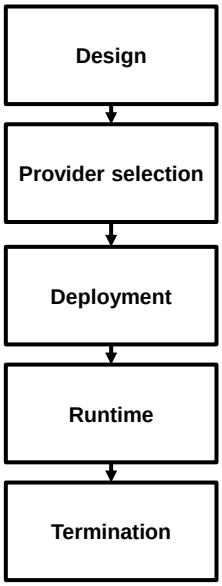


Figure 3. Lifecycle Aspects of Policies.

a) Design

In this phase, a service provider develops the solution as a Service Template (TOSCA model). Policies in this stage of the lifecycle are non-functional requirements either provided by the customer or chosen by the service provider himself. This includes the choice of software components, infrastructure requirements, and topology layout. These policies are not programmatically enforced or defined; they rather comprise a set of guidelines for service development to which the provider adheres.

b) Provider Selection

The provider selection is the first stage where policies are defined in a machine-readable form in order to be automatically evaluated. The requirements defined by these policies are used to determine, which Cloud providers are suited for service deployment. They include capabilities like redundant networking and disaster recovery, off-site backup, and the geographic location of the data center.

c) Deployment

After handling the previous aspects, the selected provider deploys the solution. Policies affecting this lifecycle stage determine the structure of the service topology and the configuration of its components. The service is set up according to the value of the policy property. Policies at this stage may determine characteristics such as the amount of memory and processing that get assigned to a certain node, the type of encryption used in transport protocols or the level of error-logging.

d) Runtime

The runtime phase starts after the service is deployed and available. It ends when service termination is requested. Policies belonging to the runtime stage determine the behavior of the service in a certain situation. This includes the response to events such as a high response time or system load, components becoming unavailable or other events such as suspicious system/user behavior. Therefore, policies at this stage often have a policy effect of the type *monitoring*.

e) Termination

This is the last stage in the service lifecycle. Policies of this type determine the actions that are taken when the service is terminated. This includes secure deletion of customer data and sending shutdown-notifications to connected clients.

2) Layer in Topology and Ecosystem

The second aspect classifies the annotation of the policy in the topology template. Policies may be annotated to a specific node or relationship with a specific type, e.g., a database or database-connection relationship. These are local policies. Global policies are annotated to the whole topology template. These affect the entire service and often recursively apply to many components of the topology. By defining boundary definitions, it is possible to select a subset of a topology and annotate this subset only. This subset is handled in a similar way as a global policy.

3) Policy Effect

This aspect defines what effect a policy has or what operations it triggers.

a) Assurance

This policy effect indicates that the property of a policy is assured by the design of the service or the Cloud provider. It is not programmatically enforced during runtime or deployment but rather the service provider has to engineer the service in a way that it guarantees compliance with the policy. Policies with this effect usually affect the lifecycle stages prior to runtime. Policies with this effect include geographical data center location and redundant networking.

b) Enforcement

Policies with this effect are actively enforced during a certain stage of the service lifecycle. They determine the service behavior according to the set property. They usually occur in the lifecycle stages including and after deployment. Policies with this effect include setting up encrypted transport channels and encrypting stored data.

 ISBN: 978-1-61208-298-1

c) Monitoring

Monitoring policies determine certain parameters of the service that trigger an operation. They usually occur in the lifecycle stages after deployment. Contrary to enforcement policies, they do not determine service behavior by default but rather react to a certain event, although these two policy effects are similar in the way that they trigger operations. Policies with this effect include automatic scaling and error notification. Table I shows three example policies and their main aspects.

TABLE I. EXAMPLE POLICIES WITH DIFFERENT ASPECTS

	Region policy	SQL-Injection firewall policy	Response time policy
Property	EU	Include firewall	100ms
Lifecycle stage	Deployment	Design	Runtime
Topology layer	Global	Local (DB node)	Local (HTTP node)
Policy effect	Enforcement	Assurance	Monitoring

Aspects of policies often depend on the specific implementation and most policies can be defined and implemented using different aspects. The Region Policy, for example, could be enforced during provider selection by selecting a provider that uses data centers in the EU. It could also be enforced during service design by implementing the service model in a way that it will only deploy on servers within the EU.

IV. ARCHITECTURE OF THE POLICY-FRAMEWORK

Figure 4 shows the architecture consisting of components for the different lifecycle stages of the Cloud service:

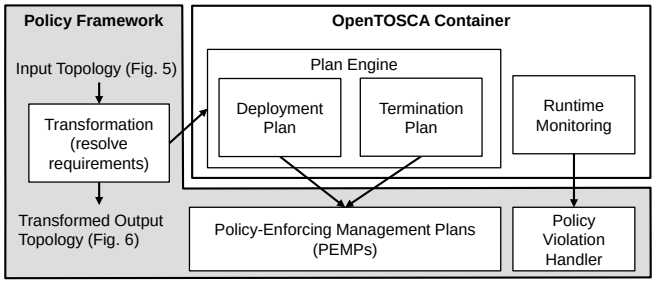


Figure 4. Architecture of the Policy-Framework.

Initially, the Cloud service is described as a solution package using TOSCA. This solution can be annotated with various policies. This TOSCA description is the input for the first stage, the *transformation*. The solution can contain *abstract components* that represent a *requirement* for a certain functionality. These abstract components are refined to *specific components* during the transformation while the annotated policies and functional requirements are considered. We therefore, differentiate between the requirement for a policy and a policy. A policy is provided as part of a system component and it can be enabled and configured according to its property. A requirement for a policy, therefore, limits the choice of components to the ones providing that policy. The transformation also adapts the deployment plan of the solution in order to cope with and install the newly added components.

Figure 5 shows an example for the transformation with an annotated requirement for a policy, here a requirement for SQL injection prevention is annotated. Figure 5 is an excerpt of the main example in Figure 1 and shows the database stack from the example service topology only. The transformation then processes this Service Template and retrieves a security-policy-pattern for the annotated requirement.

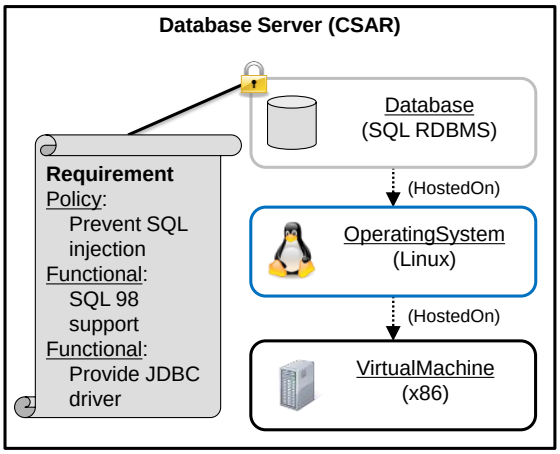


Figure 5. Transformation Example - Input Topology.

Figure 6 shows the result where the abstract database is replaced by a specific one (Oracle 11g). Furthermore, to secure this Database System a new node *database firewall* is added. To show that the new solution implements the policy requirement. Additionally, the new node it annotated by a "Prevent SQL injection" policy that is realized by the design of the system, which documents that the system is now fulfilling the policy.

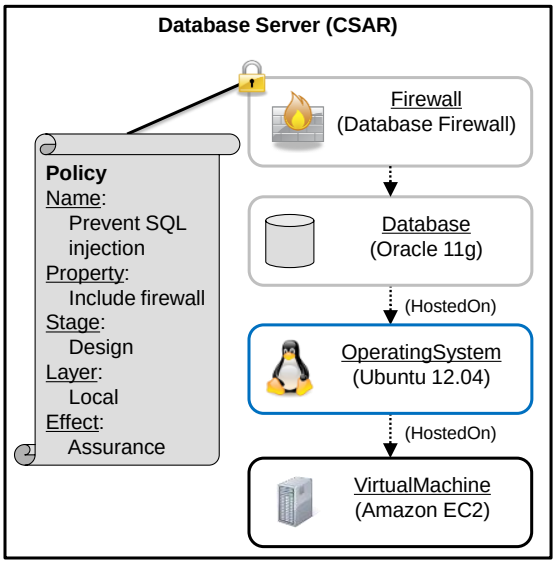


Figure 6. Transformation Example - Resulting Topology.

 ISBN: 978-1-61208-298-1

The next step in the architecture in Figure 4 is the installation of the *transformed solution* to the TOSCA container. For that purpose, the TOSCA container interprets the solution and starts the Cloud service deployment. The plan engine installs the package. Afterwards, Cloud services can be instantiated by using the *deployment plan*.

Policies that have the aspects enforcement and deployment must be enforced during deployment of the service. For that purpose, so-called *Policy-Enforcing Management Plans* (PEMPs) are used, which implement the functionality that guarantees that the defined policies are met.

The next step in the lifecycle is the runtime phase of the Cloud service. This phase starts after the deployment has successfully finished. During the runtime phase, different policies can be in effect, e.g., the *response time policy*. For this purpose, a *Runtime Monitoring* component is used, which continuously monitors the service and detects policy violations. A violation is handled based on the requirements, e.g., stop the service or scale up using a *Policy Violation Handler*.

The last step in the lifecycle is the *termination* of the service. In this step, the same mechanism used during deployment (PEMPs) is used. A typical task for termination policies is to guarantee secure deletion of all customer data.

V. CONCLUSION AND OUTLOOK

The contribution of this paper is to introduce and establish aspects of policies in the context of Cloud service definition and to present a first architecture realizing these aspects in a Policy-Framework. The proposed framework supports non-functional aspects in Cloud service models that are built using TOSCA.

Security is a major concern in using Cloud Computing for outsourcing data and services especially in the enterprise segment. It is, therefore, important to agree upon a common standard for describing, implementing, and realizing security policies in Cloud service models.

The next steps are to implement the different components of the proposed policy-framework, but even more important is the definition of a catalog of security policies and their implementation in topology components using TOSCA.

ACKNOWLEDGMENT

This work was partially funded by the BMWi project CloudCycle (01MD11023).

REFERENCES

[1] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, and D. Schumm. Vino4TOSCA: A Visual Notation for Application Topologies based on TOSCA. In *CoopIS*, pp. 416–424. Springer-Verlag, 2012.

[2] P. Lipton, S. Moser, D. Palma, and T. Spatzier. Topology and Orchestration Specification for Cloud Applications (TOSCA). http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html, March 2013.

[3] P. Mell and T. Grance. The NIST Definition of Cloud Computing. *Recommendations of the National Institute of Standards and Technology*, Special Publication 800-145:7, 2011.

[4] H. Takabi, J.B.D. Joshi, and G.-J. Ahn. SecureCloud: Towards a Comprehensive Security Framework for Cloud Computing Environments. In *Computer Software and Applications Conference Workshops (COMPSACW)*, IEEE 34th Annual, pp. 393–398, 2010.

 ISBN: 978-1-61208-298-1

Implementation of Trust Metrics in X.509 Public Key Infrastructure

Lucas Gonçalves Martins and Ricardo Felipe Custódio
Departamento de Informática e Estatística
Universidade Federal de Santa Catarina
Santa Catarina, Brazil
Email: {lucasgm,custodio}@inf.ufsc.br

***Abstract*—The *X.509* hierarchical public key infrastructure model is used to distribute trust and to decentralize the responsibility of managing digital certificates among certification authorities. However, the trust indiscriminately flows through these certification authorities, allowing any of them to issue trusted certificates. Many works propose trust quantification and calculation as a solution for this problem, but most of them apply their proposed methods in hypothetical public key infrastructure networks. In this paper, we propose a plausible implementation of quantification and calculation of trust for the *X.509* public key infrastructure, specifying *ASN.1* structures and trust management procedures to initialize and update trust values in this model's relationships.**

Keywords*—*PKI; Trust; Certificate; Metrics; Calculation; X.509; ASN.1; Hierarchy.

I. INTRODUCTION

Nowadays, the *X.509* standards are the most used Public Key Infrastructure (PKI) model. However, this model has been subject of concern about its security and usability [1], [2]. Nevertheless, the usage of digital certificates, as a method for identification and authentication in the digital environment, has been growing over the years. With this growth, there have also been an increasing of attempts to obtain fraudulent digital certificates to impersonate big companies on the Internet [3], [4], [5].

Most of the attacks are done to Certification Authorities (CA) at the end of the PKI hierarchy, which usually use semi-automatic on-line applications to manage their certificates. The attackers use a combination of social engineering along with the exploitation of these application flaws to obtain valid certificates. The only way to avoid the attacks is to guarantee the security of the technology stack used by all CA's applications, and to take the human factor off from its sensitive procedures; a virtually impossible task, considering the number of trusted CAs (in the most popular repositories) and the diversity of applications used by them. Since we cannot guarantee the security of all CA applications, we need to focus on the reason why certification authorities are chosen by hackers.

The *X.509* PKI trust model follows a hierarchical structure to distribute trust among PKI entities and decentralize the responsibility of managing digital certificates. Also, if a node of the hierarchy is compromised, only its adjacent nodes will be compromised, too. This characteristic makes the attacks on lower levels of the PKI less harmful to the whole PKI.

However, when a CA delegates a services to another CA, it also gives its trust to that CA. The trustworthiness is transitive in the *X.509* PKI [6] and all certificates in the hierarchy have the same trust value. As a result, all delegated CAs are as trustworthy as the root CA, but most of them are not as secure as it is. This allows an attacker to attack the entities at the bottom of the hierarchy to obtain a certificate as trustworthy as a certificate issued by the entity at the top of it. Burmester [2] also wrote about this issue: *"The problem with X509 is that it cannot tolerate even one penetration: each node [in the hierarchy] is a single point of failure."*

This flaw leads to another, which allows the attacker to profit from the stolen certificate. The entities in a hierarchy should be organized through a measurable organization criteria, which defines in which level of the hierarchy the entity should be placed. However, the *X.509* hierarchy does not have a well-defined organization criteria, allowing any certificate to be positioned below any CA, regardless of the importance of the certificate owner. For example, when a certificate is issued to identify a company, the company will be considered inferior to the CA, which is not necessarily true. Thereby, an attacker can attack a less important and, probably, more insecure company (the CA) to jeopardize a bigger one.

Several works propose the usage of trust quantification and calculation as a solution for these problems. Through the measurement of trust, it is possible for a certificate verifier to decide if the calculated trust level of a certificate is enough for the context in which he is using it. However, most of these works apply their method to hypothetical web PKIs. In this paper, we propose a plausible implementation of trust quantification and calculation for *X.509* PKI. We specify *ASN.1* structures to represent trust values, as well as trust management procedures for the initialization and update of these values. The model is specified to be independent of the trust calculation method. However, we use Jøsang's trust model [7] to interpret the trust relationships, and use his quantification and calculation methods as an example of how to measure trust in a PKI [8].

Through our proposal, we want to bring all the benefits of trust metrics to the *X.509* PKI. With trust metrics we can build a stronger PKI against attacks and introduce semantics in the certificate verification procedures, that may help end-users to decide if he should trust in a specific digital certificate or not. We also tried to make the trust metrics implementation the less impacting possible to the *X.509* standards, keeping it compatible to the existent PKI-enabled applications.

 ISBN: 978-1-61208-298-1

The remainder of this paper is organized as follows. In Section II, we present the related works found in our research. In Section III, we present the Jøsang definitions of trust, as well as his notations for trust networks and his method to calculate trust. Also in this section, we specify a method of interpreting a PKI as a trust network. In Section IV, we propose the procedures to initialize and update the trust values in the *X.509* PKI. In Section V, we show how our method can be used to solve the problems discussed in Section I. Finally, in Section VI, we give our final considerations and future works.

II. Related Work

In this section, we present the related works found in our research. First, we give a brief overview about the studies directly associated with trust models and the generic view of trust. Then, we discuss PKI trust models and the application of trust metrics to improve them.

A. Trust Concepts and Models

Trust modeling is a topic easily found in the literature. Several works study the semantics of trust and its transitivity, with a generic view about the subject [9]. Other works defined notations for the specification of trust networks that can be used to evaluate and measure trust [7][10][11][12][13]. Among these studies we highlight the following.

Ruohomaa [9] presented the concepts of trust and its applications in computer science. She defines trust as the extent to which one party is willing to participate in a given action with a given partner, considering the risks and incentives involved. She also discusses trust management life-cycle, which she defines in three steps: determining initial trust, observing the trustee's actual behavior and updating trust accordingly. We use this definition in our work to propose the trust management in the *X.509* PKI.

Jøsang [7][10] is an active researcher in the trust model field. He published several works about basic trust concepts, proposed notations for trust network specification and applied subjective logic with belief calculus to measure trust in trust networks. His notations and trust calculus are better explained in Section III.

Trust metrics have a visible application in PKI models, such that most of the work in this field uses PKI as a practical example for their proposals. However, they usually give a superficial view of the problem, and do not define real solutions for a hierarchical PKI. In the next sections, we present the studies of PKI trust models and the existent proposal that applies trust metrics in PKI.

B. PKI Trust Models

The *X.509* PKI is specified by several documents called *Request For Comments* (RFC), which are maintained by the *Internet Engineering Task Force* (IETF) [14]. The RFC 5280 - *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile* - specifies the data structures of digital certificates and CRL, as well as the interpretation of these structures and an algorithm for certification path validation [15]. We use these specifications to understand the semantics of trust in *X.509* and how the trust flows through the PKI entities.

There are several works in the literature that specify different trust models for PKI [16][17][18][19]. However, most of them address structural characteristics and procedures for certification path validation, with a superficial approach to the concepts and semantics of trust. As the proposal of this paper does not change the *X.509* structure, we focus on the works that apply trust metrics in PKI [20][21][8][22].

Maurer [20] proposed in his work a deterministic PKI model, based on recommendations and confidence levels. This model specifies trust relationships as predicates of authenticity, trust and recommendation. He also defined inference rules to be applied to an initial set of predicates, generating new predicates of trust and authenticity (these inferences are similar to certification path validation that uses path size limitations). Based on the deterministic model, he proposed a probabilistic model, which uses probabilistic logic to measure the confidence of his predicates. Through these values, he calculates the resultant trust of predicates generated by inferences.

Jøsang [8] uses his trust model to interpret public key infrastructures and measure their trust relationships. He also specifies an algebra to calculate trust transitivity in the PKI, which we discuss in Section V of this paper. Levien [22], by contrast, addresses the efficiency of PKI trust models that use trust metrics to resist to attacks. He proposes an attack model to be used as a framework to calculate the index of attack-resistance efficiency. He also demonstrates that the closest attacks to the certificate verifier have more chances of success.

III. Specifying Trust Networks

In this section, we present Jøsang's [7][10] notation for trust network specification, and his proposed methods to measure trust and calculate its transitivity. We use these tools to define a certificate interpretation method that allows us to build a trust network based on PKI hierarchy.

Jøsang uses Gambetta's definition of trust, which he calls *reliability trust*. He represents the reliability trust as a tuple (A, B, P, μ, τ), that can be interpreted as: A trusts B for the purpose P with the measure value μ in the specific time τ. A and B are principals, or nodes, in the trust network. P is the purpose of the trust (e.g., "being a good mechanic"). The kind of the measurement μ is arbitrary, and τ is any representation of time. In our examples, we may use a simplified notation, making μ and τ implicit, as in (A, B, P).

> **Definition 1 (Reliability Trust)** *"Trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends [23]."*

The purpose P can be categorized by prefixes in its identifier, as xyP. The category x identifies if the trust is direct (d) or indirect (i), while the category y identifies if the trust is functional (f) or for referral (r). A functional trust means that A trusts in B for the purpose P (e.g., A trusts B to be a good mechanic), while a referral trust means that A trust in who B recommends for the purpose P (e.g., A trusts B to recommend a good mechanic). The direct trust is used when

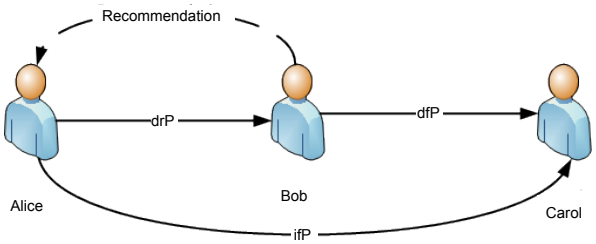


Fig. 1. Trust Network

A direct trusts B, and the indirect trust is used when A trusts B because of a recommendation of an entity X that A trusts. These modifiers generate four kinds of purpose: dfP, drP, ifP and irP.

Figure 1 shows a graph that represents a simple trust network. Considering the node Alice as A, Bob as B and Carol as C, and using ":" as a transitive connection between two consecutive trust edges of the graph, we can specify this network as follow:

$$(A, C, ifP, \mu_1, \tau_1) = (A, B, drP, \mu_2, \tau_2) : (B, C, dfP, \mu_3, \tau_3) \quad (1)$$

To calculate the measurement μ_1 based on μ_2 and μ_3, Jøsang proposed the usage of a belief metric called opinion, explained below:

> *"Subjective logic represents a specific belief calculus that uses a belief metric called opinion to express beliefs. An opinion denoted by $\omega_x^A = (b_x^A, d_x^A, u_x^A, a_x^A)$ expresses the relying party A's belief in the truth of statement x. Here b, d, and u represent belief, disbelief and uncertainty, and relative atomicity respectively where $b_x^A, d_x^A, u_x^A, a_x^A \in [0, 1]$ and the following equation holds:*
>
> $$b_x^A + d_x^A + u_x^A = 1 \quad (2)$$
>
> *The parameter a_x^A reflects the size of the state space from which the statement x is taken. In most cases the state space is binary, in which case $a_x^A = 0.5$".*

Assume that the values of μ_2 and μ_3 are $\omega_B^A = (b_B^A, d_B^A, u_B^A, a_B^A)$ and $\omega_x^B = (b_x^B, d_x^B, u_x^B, a_x^B)$, respectively. We use the discount ($\otimes$) operator, from subjective logic, to calculate the transitive opinion $\omega_x^{A:B} = (b_x^{A:B}, d_x^{A:B}, u_x^{A:B}, a_x^{A:B})$, using the definitions below:

$$\omega_B^A \otimes \omega_x^B = \omega_x^{A:B} = \begin{cases} b_x^{A:B} = b_B^A b_x^B \\ d_x^{A:B} = b_B^A d_x^B \\ u_x^{A:B} = d_B^A + u_B^A + b_B^A u_x^B \\ a_x^{A:B} = a_x^B \end{cases} \quad (3)$$

We also define the conjunction ($\wedge$) operator, from subjective logic, to calculate the trust of an entity in two different statements. Assume the following opinions $\omega_x^A = (b_x^A, d_x^A, u_x^A, a_x^A)$ and $\omega_y^A = (b_y^A, d_y^A, u_y^A, a_y^A)$. To calculate the resultant conjunction $\omega_x^A \wedge \omega_y^A$, we use the following definition:

$$\omega_x^A \wedge \omega_y^A = \omega_{x \wedge y}^A = \begin{cases} b_{x \wedge y}^A = b_x^A b_y^A \\ d_{x \wedge y}^A = d_x^A + d_y^A - d_x^A d_y^A \\ u_{x \wedge y}^A = b_x^A u_y^A + u_x^A b_y^A + u_x^A u_y^A \\ a_{x \wedge y}^A = a_x^A a_y^A \end{cases} \quad (4)$$

Now, we will demonstrate how to specify an *X.509* PKI in the Jøsang's notation. We use the PKI represented by the directed graph in Figure 2, as an example. We simplify a certificate as a tuple $C = (X, Y, k, p, s)$, where X is the certificate issuer, Y the certificate subject, k a public key, p a certificate policy, and s a signature done by X over these data. To interpret the certificate as a trust relationship, we read this tuple as follows: (a) X trusts that Y is responsible for the public key k; (b) X trusts that Y follows the certificate policy p; and (c) s proves the authenticity of X's trust in (a) and (b).

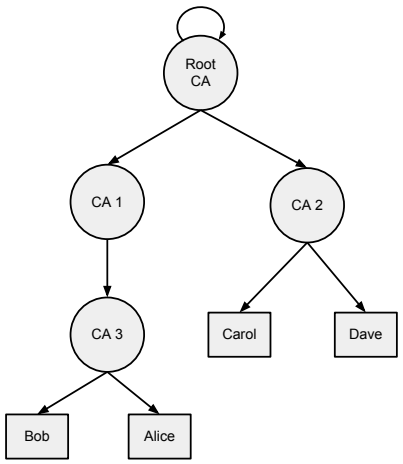


Fig. 2. X.509 PKI Hiearachy Example

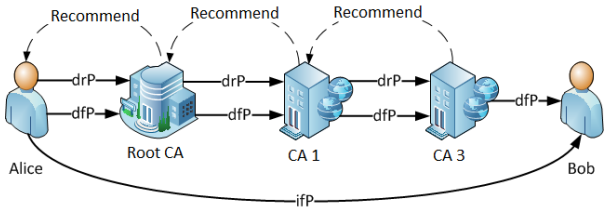


Fig. 3. Certification Path's Trust Network

We interpret (a) as a direct functional trust, specified as (X, Y, dfP), where the purpose P is "being responsible for the public key k". The interpretation of (b) is more complicated. We could interpret it as a functional trust with the purpose of "following the certificate policy p"; however, it would be different from the purpose in (i). As the purposes need to be the same for the transitivity to be possible [7], we interpret (ii) as follow: if X trusts that Y follows its certificate policy, and Y is a certification authority, X trusts that Y correctly

issues certificates. In other words, X trusts in the certificates that Y recommends. This means that X has a direct referral trust in Y, specified as (X, Y, drP), where the purpose P is "recommend some entity as responsible for the public key k".

To give an example of how to interpret a certification path as a trust network, we assume that Alice (A) wants to verify the authenticity of Bob's certificate (B), using the PKI in figure 2. Following the certification path validation procedure [15], we build the certification path shown in table I, where the *Root CA* is R, CA 1 is I and CA 3 is F.

TABLE I: CERTIFICATION PATH

$$\begin{aligned} C_R^R &= (R, R, k_r, p_r, s_r) \\ C_I^R &= (R, I, k_i, p_i, s_r) \\ C_F^I &= (I, F, k_f, p_f, s_i) \\ C_B^F &= (F, B, k_b, p_b, s_f) \end{aligned}$$

Alice's trust in the *Root CA* has the same meaning of the trust in a certificate, except for its signature (Alice does not need proof of her trust relationships). So, we interpret her trust in the root CA as (A, R, dfP) and (A, R, drP). Interpreting the certificates in the certification path, we build the trust network illustrated in Figure 3. Following Jøsang's proposal [8], we calculate the resulting trust transitivity value as shown bellow, the identifier k being for functional trust and the identifier p for referral trust.

$$\omega_{B_k}^{A:F} = (\omega_{R_k}^A \wedge \omega_{R_p}^A) \otimes (\omega_{I_k}^R \wedge \omega_{I_p}^R) \otimes (\omega_{F_k}^I \wedge \omega_{F_p}^I) \otimes \omega_{B_k}^F \quad (5)$$

IV. Trust Management in *X.509*

In this section, we demonstrate how to implement trust quantification and calculation in the *X.509* PKI model. We define *ASN.1* structures that represent trust values, to be used in the existent structures of the *X.509* model. We also define three kinds of trust in the *X.509* PKI: authentication trust, *policy trust* and *PKI trust*. For each of these, we describe procedures that cover the life-cycle of trust management, defined by Ruohomaa: (i) determining initial trust; (ii) observing the trustee's actual behavior and; (iii) updating trust accordingly [9]. The correct semantic interpretation of trust measures is important because trust has a strong context-dependence, and any misinterpretation can lead to a different sample space [11].

```
Trust ::= CHOICE {
   opinon   Opinion}
```

Fig. 4. *ASN.1 Trust* structure

```
Opinion ::= SEQUENCE {
   belief      INTEGER (0..100),
   disbelief   INTEGER (0..100),
   uncertain   INTEGER (0..100)}
```

Fig. 5. *ASN.1 Opinion* structure

To represent a generic type of trust measurement, we define a *Trust ASN.1* structure of type *CHOICE*, which is composed by a set of trust measurement methods. We also define another *ASN.1* structure that represents the opinion based trust measurement. This structure name is *Opinion* and it is a *SEQUENCE* of *INTEGER* values, identified as *belief*, *disbelief* and *uncertain*. Each of these values can be set in the integer interval $[0, 100]$, to represent the percentage of belief, disbelief and uncertainty, from the belief calculus of subjective logic. We show the *ASN.1* specification of these structures in Figures 4 and 5.

A. Authenticity Trust

We call the direct functional trust *authenticity trust*, because it is formed when a certification authority issues a certificate, creating proofs for the bind between the certificate's subject and its public key. In other words, the CA trust in the certificate authenticity. To define the initial trust in this kind of trust relationship, we need to answer the following question: How much does a CA need to believe in the bind between an entity and its public key to issue a certificate for it?

Considering that the certification authority is responsible for the identification and authentication of certificate applicants, the value of CA's trust over the bind is equivalent to the trust over their own procedures. Therefore, in our interpretation, the answer for the question is 100% belief, because the CA can't have doubts about their own procedure (this scenario changes when the CA uses an RA, because the CA have a direct referral trust on the RA). Hence, we use binary value of trust, that in the opinion metrics is expressed as $(1.0, 0.0, 0.0)$ for all issued and not revoked certificates; $(0.0, 1.0, 0.0)$ for all revoked certificates; and $(0.0, 0.0, 1.0)$ for all expired and not issued certificates.

These quantification rules do not change the already used procedures for certificate issuance and revocation (initialization and update), neither do they need new *ASN.1* structures to represent them. During the certificate validation, the values are assumed according to the certificate status (valid, revoked, expired). Moreover, these trust value constraints prevents an attack over a final certification authority to create certificates with differentiated values, which would give to the attacker the power to manipulate the final trust value of the certificate.

If the PKI wants to define different values for the *authenticity trust*, it can use a certificate extension that includes a *Trust* structure as an extension value. However, the CA also needs to specify an update method for this kind of trust, which the certificate owner needs to know how to use. In the next section, we specify some methods for this purpose, however they are specified for certification authorities and may be complicated for an end user to understand. So, we discourage its use in end-user certificates.

B. Policy Trust

The *policy trust* refers to the direct referral trust relationship that is formed when a CA issues a digital certificate for another CA, trusting that the CA will follow its certificate policies. This trust relationship is different from the *authenticity trust*, because an entity needs to trust in another entity behavior, which we interpret as expectation. Therefore, we measure this kind of trust as follows: the trustor CA defines a belief value, b, that the trustee CA will follow its certificate policies. The disbelief value has to be zero, $d = 0$, as the

 ISBN: 978-1-61208-298-1

trustor CA should not issue a certificate if it has any belief that the trustee CA will not follow the policies. Thus, if $d = 0$ and $b+d+u = 1$, then $u = 1-b$, reflecting the uncertain value (not disbelief) about the trustee CA following the policies. Thereby, we define the *policy trust* values as: $(b, 0, 1-b)$.

The *policy trust* initialization happens when a certification authority issues a certificate for another CA. As the values may be different for each CA certificate, the issuer CA needs to use the proposed *Trust ASN.1* structure to bind the trust value to the issued certificate. As the value represents the trust in the CA following a certificate policy, we extend the *Qualifier ASN.1* structure, which is used to define a *PolicyQualifier* for a policy in the certificate policies extension, as defined by the RFC 5280 [15]. We include the *Trust* structure as a possible *Qualifier* choice, as shown in Figure 6. With this structure, the issuer CA can define a trust value as a *PolicyQualifier* of a certificate policy in the certificate. Thereby, the CA may also define different values for each of the policies followed by the subject CA.

```
Qualifier ::= CHOICE {
   cPSuri       CPSuri,
   userNotice   UserNotice,
   trustValue   Trust}
```

Fig. 6. *ASN.1 Qualifier* structure

Updating the *policy trust* is a big challenge. The initial trust value is set in the certificate structure before being issued. As the certificate is an immutable structure, the trust updates need to be done in other ways. The first and simpler way is through a certificate revocation list (CRL). We could expand the CRL usage to support *policy trust* updates. This way, it is possible to recovery the newest trust values from the latest issued CRLs. However, CRLs are considered one of the biggest problems in the *X.509* PKI, and encouraging the usage of CRLs for other purposes, beyond certificate revocation, is not a good practice.

The second way to update the *policy trust* is through recommendations. Recommendation is defined in Maurer's work as a mechanism for a supposed entity X prove his trustworthiness in another entity Y. However, the recommendation does not need to give proofs of Y's authenticity [20]. We can represent this relationship in Jøsang's model as a direct referral trust: (X, Y, drP). We can implement a recommendation as an attribute certificate that defines a *PolicyInformation* as an attribute. The truster CA can issue attribute certificates for all its trustee CAs, defining a *PolicyInformation* attribute with a *Trust Qualifier*, to be used as a *policy trust* update. Thereby, the trustee CAs become responsible for distributing, with its certificates, its newest recommendations.

This brings us to the third way of *policy trust* update, the time. With a decay function, we can reduce the trust value as time passes. This calculation can be done during the certificate path validation. This function may be useful to force the certification authorities to update and distribute their newest recommendations, to ensure that their confidence levels will always be high.

C. PKI Trust

The last kind of trust is the *PKI trust*, which is formed when an end-user trusts in a trust anchor. This kind of trust is composed by the *authenticity* and *policy trust*, which follow the same value rules defined to these kinds of trust. However, their management is different because the end-user does not issue a certificate to the trust anchor. The definition of the trust values need to be done through an out-of-bands method. To discuss how to do this management, we consider two different scenarios: in the first, Alice, an end-user, relies in just one trust anchor; and in the second, Alice relies in a set of trust anchors.

In the first scenario, any value defined by Alice to her relationship with the trust anchor will equally affect all certificates in the PKI. So the trust value of this relationship does not create any evidence that can be used in Alice's decision-making. Therefore, we define the *PKI trust* value, in this scenario, as $(1, 0, 0)$, which represents the *policy trust* of Alice on the trust anchor. This value does not reduce the resultant value of the transitivity trust from the trust anchor to the certificates below. As a fixed value, Alice does not need to manage it. In the certification path validation, the value is assumed when the trust anchor is defined.

By contrast, in the second scenario, the values defined by Alice for each trust anchor will affect only the certificates under the respective trust anchor. So, Alice might use these different values to help in her decision-making. However, Alice has to initialize and update her trust values for every trust anchor that she trusts, what may be a hard task for a end-user. But, as it is already done nowadays, a relying party can be responsible for this management, setting and updating the trust values in a certificate repository that Alice fully trusts.

V. Analysis

In this section, we present an example of using calculation of trust in a *X.509* PKI, following our proposed trust management. All our calculations are based on the work of Jøsang about trust algebra in PKI [8] and in the arbitrary values set for the trust relationships in the PKI illustrated in Figure 2, represented in Table II. Using the table values, we calculate the trust value through the following formula (The calculations were made with the calculator available at [24]:

$$\omega_{B_k}^{A:F} = (\omega_{R_k}^{A} \wedge \omega_{R_p}^{A}) \otimes (\omega_{I_k}^{R} \wedge \omega_{I_p}^{R}) \otimes (\omega_{F_k}^{I} \wedge \omega_{F_p}^{I}) \otimes \omega_{B_k}^{F} = (0.81, 0.00, 0.19)$$

To evaluate the behavior of the trust calculation when a certificate is revoked, we use the update values, ω'^{F}_{B} e ω'^{I}_{F}, with value $(0, 1, 0)$, that represents the Bob's and CA-3's certificate revocation, respectively. the calculus is given below:

$$\omega_{B_k}^{A:F} = (\omega_{R_k}^{A} \wedge \omega_{R_p}^{A}) \otimes (\omega_{I_k}^{R} \wedge \omega_{I_p}^{R}) \otimes (\omega_{F_k}^{I} \wedge \omega_{F_p}^{I}) \otimes \omega'^{F}_{B_k} = (0.00, 0.81, 0.19)$$

$$\omega_{B_k}^{A:F} = (\omega_{R_k}^{A} \wedge \omega_{R_p}^{A}) \otimes (\omega_{I_k}^{R} \wedge \omega_{I_p}^{R}) \otimes (\omega'^{I}_{F_k} \wedge \omega_{F_p}^{I}) \otimes \omega'^{F}_{B_k} = (0.00, 0.00, 1.00)$$

As can be seen, when the *CA 3* revokes Bob's certificate, the result value of belief in the trust transitivity calculation changes to disbelief, while the value of uncertainty remains the same. These values are correct, because we set a value of uncertainty for *CA 3*, which may have incorrectly revoked Bob's certificate. The second scenario shows that when the *CA*

 ISBN: 978-1-61208-298-1

TABLE II: ARBITRARY *POLICY TRUST* VALUES

$$\omega^{A}_{R_p} = (1.00,\ 0.00,\ 0.00)$$
$$\omega^{R}_{I_p} = (0.90,\ 0.00,\ 0.10)$$
$$\omega^{R}_{O_p} = (0.90,\ 0.00,\ 0.10)$$
$$\omega^{I}_{F_p} = (0.80,\ 0.00,\ 0.20)$$

3 has its certificate revoked by the *CA 1*, Bob has its belief and disbelief values set at 0, while the value of uncertainty is set at 1. As the *CA 3* certificate's authenticity is no longer reliable, it is impossible to define which certificate is valid or invalid, making it impossible for us to make any conclusion about Bob's certificate.

In our model, we can define an organizational criteria for the PKI hierarchy, and use it to graduate the trust value through its levels. We can see an example of this graduation in the Table II. The *Root CA* has a belief value of one, its subordinates CAs have a belief value of 0.9, and so on. With this graduation of trust, we can establish an organizational criteria. For example, offline CAs usually are more secure than online CAs. So, offline CAs should appear on the tops levels of the PKI hierarchy (with a higher belief value), while the online CAs should be close to the bottom of the hierarchy (with lower belief value).

Following this organizational criteria, we can give an example of how the PKI can resist to an attack, using the PKI Illustrated in Figure 2. Assume that the *CA 3* is an online CA, and the *CA 2* is an offline one. Alice is an end-user that accesses Carol's web site that has a certificate issued by the *CA 2* (*O*). Using Levien's [22] attack model, we can verify how Alice will be protected if the *CA 3* gets attacked.

Levien defines two attack types, which he calls edge attack and node attack. In the edge attack the attacker can issue a certificate (create an edge) from a specific node, while in the node attack, the attacker can create any number of certificates (edges) from the attacked node. However, in any of these scenarios, if the attacked node is the *CA 3*, the attacker will not be able to issue any certificate with a transitivity trust value higher than $(0.00, 0.81, 0.19)$, because the trust values are defined in the CAs certificates and the hierarchy does not have multiple certification paths for one certificate. Even if the attacker issues several CA certificates with full trust value under the *CA 3*, the resultant transitivity value will be decreased to the value of $(0.00, 0.81, 0.19)$, when it passes through the *CA 3*'s certificate.

Assuming that Alice has already verified Carol's certificate before the attack over the *CA 3* happened, she will know that the certificate has a trust value of $(0.93, 0.00, 0.07)$. So, if the attacker uses a fraudulent certificate, in name of Carol, issued by the *CA 3*, Alice will be able to identify that it is a fraudulent certificate, because of its lower trust value. The only way for the attacker to be successful, is attacking the CA 2. However, it is a much harder task, considering that it is an offline CA.

$$\omega^{A:O}_{C_k} = (\omega^{A}_{R_k} \wedge \omega^{A}_{R_p}) \otimes (\omega^{R}_{O_k} \wedge \omega^{R}_{O_p}) \otimes \omega^{O}_{C_k} = (0.93, 0.00, 0.07)$$

If Alice does not know the trust value of Carol's certificate before the attack, she can establish a minimum value of trust for the context in which she is using the certificate. For example, if Alice is accessing an online bank, she can establish a minimum belief value of 0.9, while if she is accessing a university web site, she can establish a belief value of 0.8. Through these acceptance values, we can use an adaptation of Levien's [22] attack model, to calculate the efficiency of the PKI's organization criteria to resist attacks.

Now, we analyse the efficiency of our model to resist attacks compared to the classical X.509 model. For that, we use a certificate sample retrieved from the last available snapshot (March 2011) of *SSL Observatory*'s certificate database [25]. This database contains a large number of real certificates, which are validated under *Microsoft* and *Firefox* trust anchor repositories. Table III show the number of CAs and the number and percentage of final CAs (a final CA only issues certificates to end-users), for each hierarchy level of the PKIs in the sample.

TABLE III: CA DISTRIBUTION

Level	Total	Final	Final/Total
First (root)	176	50	28.41%
Second	422	364	86.26%
Third	365	352	96.44%
Fourth	21	16	76.19%
Fifth	5	5	100%
Total	989	787	79.57%

To calculate the efficiency of a PKI to resist attacks, we use an adaptation of Levien's attack model. Levien considers all PKI's nodes (certificates) as eligible to attack. We only consider final CAs as eligible nodes, as this kind of CA usually use online systems to issue certificates, which are preferable targets for hackers. In this sample, we have 787 final CAs, which represents 79.57% of all CAs. In the conventional X.509 PKI model, the decision to trust or not in a certificate is made when the certificate is validated by the certification path validation algorithm. As the certificates of our sample were already validated, all 747 final CAs can be a target to issue certificates that will be trusted by an user that trust in *Microsoft* and *Firefox* repositories.

To compare this result with our model, we need to define trust values for the CAs in the sample. As we can not determine which CA is more trustworthy than other, we assume that each level has a trust value lower than the level above. Besides this assignment being arbitrary, it can significantly reduce the number of eligible CAs. For example, to forge a certificate with trust value equals to a certificate issued by a CA at the second level, the attacker needs to successful attack a CA of that level or above. It reduces the number of eligible CAs to 414, representing a reduction of 47.39% compared to the conventional X.509 PKI. However, as the X.509 PKI does not have a real organization criteria, we cannot determine if a certificate at upper levels of the hierarchy will be securer than certificates at the bottom levels. A single compromised CA, at the first or second level, can jeopardize all the PKI, even with trust metrics.

 ISBN: 978-1-61208-298-1

VI. Conclusion and Future Work

In this paper, we demonstrated how to use trust calculation to solve the trust transitivity flaw of the *X.509* PKI. Thereby giving end-users a new tool that helps in his decision-making, allowing him to verify the trust level of a digital certificate and identify possible frauds. As a result, attackers will be discouraged from attacking certification authorities with lower trust level, by reducing the cost-benefit of these attacks. Our work differs from other works by focusing on the interpretation of trust in the context of *X.509* PKI, and by proposing structures and procedures necessary to support the calculation of trust, without jeopardizing its standards and maintaining the compatibility with existing applications.

For future work, we have the studies about the organization criteria that should be used to evaluate the trust level of PKI entities. This criteria must ensure a greater security for the PKI, which can be assessed through the Levien's work [22]. We also need to analyse the behavior of our model when a CA uses a registration authority and when the PKI uses cross and bridge certification to integrate other PKIs. In this case, we can use all existing knowledge about the trust calculus in web PKIs.

References

[1] C. Ellison and B. Schneier, "Ten risks of pki: What you're not being told about public key infrastructure," *Comput Secur J*, vol. 16, no. 1, pp. 1–7, 2000.

[2] M. Burmester and Y. G. Desmedt, "Is hierarchical public-key certification the next target for hackers?" *Communications of the ACM*, vol. 47, no. 8, pp. 68–74, 2004.

[3] Comodo, "Comodo fraud incident," 2011, retrieved: June, 2013. [Online]. Available: http://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html

[4] Vasco, "Diginotar reports security incident," 2011, retrieved: June, 2013. [Online]. Available: http://www.vasco.com/

[5] J. Segura, "Digital certificates and malware: a dangerous mix," 2013, retrieved: June, 2013. [Online]. Available: http://blog.malwarebytes.org

[6] L. H. Shenchangxiang, "Hierarchy-Distribution Combined PKI Trust Model," no. I 00044, pp. 121–124, 1996.

[7] A. Jøsang and S. Pope, "Semantic constraints for trust transitivity," in *Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling-Volume 43*. Australian Computer Society, Inc., 2005, pp. 59–68.

[8] A. Jøsang, "An algebra for assessing trust in certification chains," in *Proceedings of the Network and Distributed Systems Security Symposium (NDSS99). The Internet Society*, 1999.

[9] S. Ruohomaa and L. Kutvonen, "Trust management survey," *Trust Management*, pp. 77–92, 2005.

[10] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision support systems*, vol. 43, no. 2, pp. 618–644, 2007.

[11] J. Huang and D. Nicol, "A formal-semantics-based calculus of trust," *Internet Computing, IEEE*, vol. 14, no. 5, pp. 38–46, 2010.

[12] H. El Bakkali and B. Kaitouni, "A logic-based reasoning about PKI trust model," *Proceedings. Sixth IEEE Symposium on Computers and Communications*, pp. 42–48, 2001. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=935353

[13] C.-N. Ziegler and G. Lausen, "Propagation models for trust and distrust in social networks," *Information Systems Frontiers*, vol. 7, no. 4, pp. 337–358, 2005.

[14] IETF, "Public-key infrastructure (x.509) (pkix)," Internet Engineering Task Force, 2013. [Online]. Available: http://datatracker.ietf.org/wg/pkix/

[15] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280 (Proposed Standard), Internet Engineering Task Force, May 2008, retrieved: June, 2013. [Online]. Available: http://www.ietf.org/rfc/rfc5280.txt

[16] R. Perlman, "An overview of pki trust models," *Network, IEEE*, vol. 13, no. 6, pp. 38–43, 1999.

[17] J. Linn, "Trust models and management in public-key infrastructures," *RSA Laboratories*, vol. 12, 2000.

[18] R. Housley and T. Polk, *Planning for PKI: best practices guide for deploying public key infrastructure*. John Wiley & Sons, Inc., 2001.

[19] C. Adams and S. Lloyd, *Understanding PKI: concepts, standards, and deployment considerations*. Addison-Wesley Professional, 2003.

[20] U. Maurer, "Modelling a public-key infrastructure," in *Computer SecurityESORICS 96*. Springer, 1996, pp. 325–350.

[21] J. Huang and D. Nicol, "A calculus of trust and its application to pki and identity management," in *Proceedings of the 8th Symposium on Identity and Trust on the Internet*. ACM, 2009, pp. 23–37.

[22] R. Levien and A. Aiken, "Attack-resistant trust metrics for public key certification," in *7th USENIX Security Symposium*, 1998, pp. 229–242.

[23] D. Gambetta, "Can we trust trust," *Trust: Making and breaking cooperative relations*, vol. 2000, pp. 213–237, 2000.

[24] S. H. Simon Pope and M. Davey, "Subjective Logic Operators Demo," University of Oslo, 2011, retrieved: June, 2013. [Online]. Available: http://folk.uio.no/josang/sl/Op.html

[25] EFF, "SSL Observatory," Electronic Frontier Foundation, 2013, retrieved: June, 2013. [Online]. Available: https://www.eff.org/observatory

 ISBN: 978-1-61208-298-1

Secure and Fast PIN-entry Method for 3D Display

Mun-Kyu Lee, Hyeonjin Nam
School of Computer and Information Engineering
Inha University
Incheon, Korea
e-mail: mklee@inha.ac.kr, jin0639@hanmail.net

***Abstract*—The personal identification number (PIN) is one of the most well-known user authentication methods. However, there have been concerns about the security of the regular PIN pad against shoulder surfing attacks. Although there are many indirect input methods based on random challenges, most of them have usability issues because they require very long authentication time and their error rates are quite high. In this paper, we introduce our ongoing work to develop a solution to this problem using a 3D display. The proposed method transmits a random challenge to a user more effectively and securely by displaying a specific challenge object with different 3D depth from the other decoy objects. A prototype of the proposed method was implemented on a smartphone with a parallax barrier-based glasses-free 3D LCD, which guarantees the physical security because only a legitimate user who is located at the right position in front of the device can recognize the 3D effect correctly, while an attacker cannot. According to our pilot test, the average authentication time and the average error rate of the proposed method are as small as 8.4 seconds and 5.0%, respectively.**

Keywords-personal identification number, authentication, smart device, 3D display

I. INTRODUCTION

User authentication is a procedure that verifies the identity and access permission of a claimed user. It is well known that there are three categories for user authentication; knowledge-based authentication, hardware token-based authentication, and biometric verification. In this paper, we are interested in the first category, which is the most prevalent one. Especially, we will focus on personal identification number (PIN).

Although PINs are widely used for smart devices, ATMs, and digital door locks, there are concerns about the security of PINs [1]. That is, anyone who observes the authentication procedure over the user's shoulder can easily impersonate the user, because the traditional PIN pad asks a user to directly input his/her PIN and this PIN is short enough for an ordinary human attacker can memorize to use in the next authentication session. This kind of attack is known as a shoulder surfing attack [2].

We can find many proposals to solve this issue in the security literature [2-6]. They were designed so that a user may enter a response to a random challenge given by the system instead of entering the PIN digits directly. The response is computed by combining properly the challenge and the secret PIN. However, the previous methods have usability issues. That is, they require too long time to enter a PIN or their error rates are too high. The major reason for this is that the procedure where the user computes an appropriate response from the given challenge is very complex due to security requirement.

In this work, we propose a solution to this problem, which is applicable to devices with a 3D display. The proposed method effectively transmits a challenge to a user by showing a challenge object with distinct 3D aspects from the other decoy elements. The challenge object defines a simple mapping between PIN digits and some alphabet characters. Then, the user only has to remember this mapping, and inputs the mapped characters though a keypad. On the other hand, the attackers without access to the 3D information cannot obtain any information about the challenge object. According to the pilot test, the PIN-entry time and error rate of the proposed method are significantly smaller than those of the existing methods. We name our method Map-3D after the design principle.

The remainder of this paper is organized as follows. Section 2 gives a brief introduction to the 3D display technology that we used to implement our proposal. The proposed method is explained in Section 3. Section 4 and Section 5 analyze the security and practical performance of the proposed method, respectively. Section 6 compares the proposed method with previously known methods. Section 7 enumerates the issues that we still have to resolve and propose future work.

II. 3D DISPLAY

Nowadays, we see many devices with 3D display, such as a 3D TV, a 3D monitor, a 3D game console, a 3D smartphone, and a 3D smart pad. Although the proposed method may be applicable to any of these devices, we consider a 3D smartphone as the implementation platform for this paper. We used an LG Optimus 3D smartphone with a parallax barrier 3D LCD whose resolution is 800 by 480 pixels. The parallax barrier [7] is a well-known method to realize glasses-free 3D display, where many thin vertical slits are regularly placed in front of an LCD so that the left and right eyes may see different sets of pixels. The difference between the images for left and right eyes produces a stereoscopic effect. However, this effect is only realized at a specific point in front of the screen, where the eyes of the legitimate user should be located. If the eyes are not at this position, the user will not recognize the 3D effect correctly. Naturally, the attacker will not be able to feel the 3D effect

 ISBN: 978-1-61208-298-1

because s/he cannot be at this sweet spot. In this way, the security of our method is physically guaranteed.

III. MAP-3D: PROPOSED METHOD

Figure 1 shows the layout of Map-3D, the new method. Map-3D is composed of two stages; the challenge display stage and the response input stage. Figure 1 is the first stage, where a 10 by 10 matrix of alphabets is given to the user as well as the row index from 1 to 0 and two touch buttons labeled '3D' and 'Next.' Each column is a random permutation of 10 alphabet characters from A through K. (We do not use letter I because it may be confused with integer 1.) As a result, each character appears exactly 10 times in the matrix. Initially, all 100 letters are displayed with the same 3D depth. When the user touches the '3D' button, the device changes the depths of letters. To be precise, a random one among 100 letters is chosen by the device, and it is displayed with depth +1, and all the other letters with depth -1, where positive and negative depths mean that an object is displayed as a prominent and depressed letter, respectively. In the example of Fig.1, the 'F' at the intersection of row 9 and column 2 is at depth +1, and it is shown to the user as a prominent letter. (Unfortunately, we cannot express the 3D effect properly on this printed paper. In fact, the picture in Fig. 1 that was taken using an ordinary 2D camera is roughly what the attacker sees.) The other 99 letters are displayed on the plane with depth -1, and the user will feel as if these letters are depressed. While the user keeps touching the '3D' button, this 3D effect is maintained.

Figure 1. Challenge display stage of Map-3D.

After recognizing the prominent letter F, the user releases the button, which eliminates the 3D effect. But, the arrangement of letters remains the same. Then, the user's task is to remember 4 specific letters in his/her short-term memory. For example, let's assume that the PIN is '6421.' The user focuses on the second column where the prominent F was located, and remembers the 4 letters at rows 6, 4, 2, and 1, which are H, K, E, and D, respectively. In the case that the user forgets the prominent letter, s/he may touch again the '3D' button to reshow the 3D effect. But, at this time, the device chooses again a new random letter to be displayed with depth +1. The user finishes the challenge display stage by touching the 'Next' button. This immediately starts the response input stage.

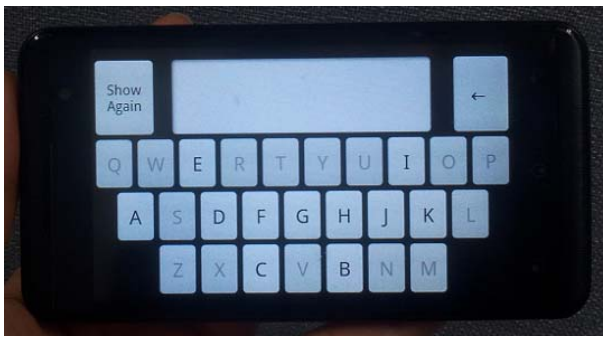


Figure 2. Response input stage of Map-3D using QWERTY keypad.

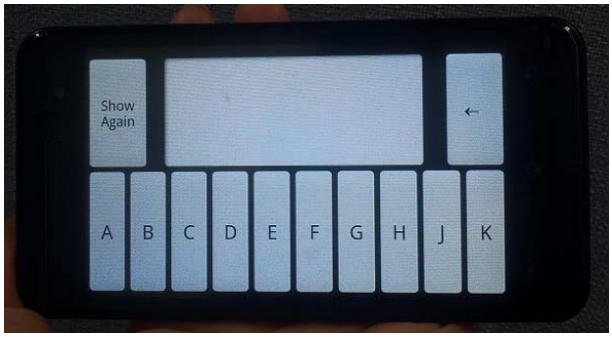


Figure 3. Response input stage of Map-3D using linear keypad.

In the response input stage, the device displays an ordinary keypad as in Fig. 2. However, the input of letters that do not belong to the challenge alphabet set is disabled. The user only has to input the four letters that s/he memorized in the first stage. An alternative input method shown in Fig. 3 is also possible. In this case, the 10 possible letters are displayed in a row.

IV. SECURITY

For security analysis, we may consider two criteria, because an attacker may have two different attack strategies; random guessing and shoulder surfing [8]. With the random guessing strategy, the attacker tries to pass the authentication test by randomly selecting the possible responses. We easily see that the success probability of this strategy is 1/10,000, which is the same as that of a regular PIN pad.

On the other hand, in a shoulder surfing attack scenario, the attacker tries to obtain useful information on the PIN by observing the authentication session of a legitimate user. Because the depths of all letters in the challenge display stage will look the same to the attacker who is not at a right position, the only thing the attacker can do is to carefully remember the arrangements of 100 letters and the 4 letters that the user enters in the response input stage. However, it is well known that a human user can remember only 7 ± 2 items in the short-term memory [9], and recent research results show that the number of items is actually even less than this value [10]. However, at this point, let's assume very pessimistically, that an attacker may memorize 10 items at the same time. Then, the best strategy will be to randomly

 ISBN: 978-1-61208-298-1

choose one column and to memorize the 10 elements in that column. Combining this information and the user's response, the attacker can recover the PIN digits with a probability, 1/10. This is 10 times less than that of the regular PIN pad. Therefore, we conclude that the proposed method is significantly safer than the regular PIN pad against a shoulder surfing attack. However, we remark that even this estimation is a very pessimistic one, and we conjecture that the real probability will be far less than 1/10.

V. IMPLEMENTATION AND PERFORMANCE ANALYSIS

For our pilot test, we used LG Optimus 3D smartphone. The software for the proposed method was programmed using Java over Android 2.3.3. We start this section by explaining the implementation details for 3D effects. As explained in the Section 3, a prominent object is given a positive depth, while a negative depth implies that an object is located at a deeper plane. This difference in depth is realized by showing different images to left and right eyes. If the depth is 0, two eyes see the same images. However, for positive and negative depths, the object is horizontally shifted by proper amount. For example, if the depth of an object is +1, the object should go to right by one position in the image for the left eye. Similarly, it should go left by the same amount in the image for the right eye.

We designed a pilot test to verify the validity of our idea. We recruited four subjects from our local university. Their ages were between 21 and 40. Two of them were male. Before the test, we explained the rationale for our proposal and the authentication procedure. The participants easily understood the working principle of the proposed method, and we did not give them any opportunity for practice. But we measured the timing and error rate from their first trials. Each participant performed ten authentication sessions with the QWERTY keypad shown in Fig. 2. Then, ten additional sessions were performed with the linear keypad in Fig. 3.

According to the experimental results, the average authentication time among the 40 sessions with the QWERTY keypad was 8.4 seconds, and the best one was as fast as 5.2 seconds. There was no significant difference in the case with the linear keypad. The average was 8.5 seconds, and the minimum was 5.4 seconds. The error rates in two experiments were 5.0% and 7.5%, respectively, where an error is defined as the case that the user completes the session but the entered PIN is different from the correct one. We remark that there were only 10 among 80 sessions where a participant touched the '3D' button more than once.

VI. COMPARISON WITH RELATED WORKS

In Table 1, we compare the performance of the proposed method with those of the previous methods including the legacy PIN pad. Table 1 lists the authentication times and error rates of various methods including the proposed one. We remark that the data for the previous methods are not from our own experiments, but they are from the papers that presented each method. But we recalculated error rates of some methods because their definition of error rate is different from ours. That is, they defined an error as the case that the user enters incorrect PINs in three consecutive trials, which is a common practice in ATMs. If we adopt this definition, the error rate of the proposed method is 0. However, we define an error as the case that the user enters an incorrect PIN in one trial.

TABLE I. COMPARISON OF SPEED AND ERROR RATE

Method	Authentication time	Error rate
Regular PIN pad	< 3	≈ 0
Binary [2]	23.2	9.0
Undercover [3]	32-45	> 31.5
VibraPass [4]	8.2	> 14.8
Haptic Wheel [5] [a]	23.0	16.4
ColorPIN [6] [a]	13.3-13.9	N.A.
Proposed: Qwerty	8.4	5.0
Proposed: linear keypad	8.5	7.5

a. Audio-based version

According to Table 1, the speed of the proposed method is much faster than those of the existing methods, except the regular PIN pad, which does not have shoulder-surfing resistance and VibraPass whose error rate is too high. The error rate of the proposed method is also very low compared to the other challenge-response-based methods.

VII. FUTURE WORK

This paper is a description of work in progress that still has several factors to be addressed as follows:

- In the first stage of the current version, the digits from 1 to 0 are arranged vertically. The horizontal arrangement could be considered. In addition, the number of letters assigned to a digit should not be always 10, but it may be customized.
- The depth difference between the prominent letter and the remaining letters are 2 in the current version. We may try other values for better 3D effect.
- In the first stage, only the letters from A to K are given to the user. We may consider a complete set of capical letters, i.e., A to Z. In this case, the impact on the security and performance should be analyzed.
- The pilot test in this paper only involves 4 subjects. We need to perform a complete test with more participants with diverse demographic backgrounds. Especially, applicability to the subjects with vision problems would be an interesting issue.
- It would be meaningful to verify if there is any significant learning effect. Therefore, we may analyze the transition of authentication times and error rates according to the accumulated number of trials through a long-term study.
- We may also redesign the experiment so that the device may present instructions that participants should follow. In the current experiments, we directly explained the new system to the participants.
- We may also try to apply the new method to non-hand-held devices such as an ATM. There are several devices available on the market to simulate this situation, e.g., 3D smart pads, although most of them require 3D glasses.

 ISBN: 978-1-61208-298-1

In our future work, we will continue our research to present a complete version of the proposed method as well as more extensive analysis results.

ACKNOWLEDGMENT

This research was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the ITRC (Information Technology Research Center)) support program (NIPA-2013-H0301-13-1003) supervised by the NIPA (National IT Industry Promotion Agency) and the IT R&D program of MOTIE/KEIT. [10039180, Intuitive, convenient, and secure HCI-based usable security technologies for mobile authentication and security enhancement in mobile computing environments].

REFERENCES

[1] J. Bonneau, S. Preibusch, and R. Anderson, "A birthday present every eleven wallets? The security of customerchosen banking PINs," Financial Cryptography and Data Security 2012, Feb. 2012, pp. 25-40.

[2] V. Roth, K. Richter, and R. Freidinger, "A PIN-entry method resilient against shoulder surfing," Proceedings of the 11th ACM Conference on Computer and Communications Security, Oct. 2004, pp. 236-245.

[3] H. Sasamoto, N. Christin, and E. Hayshi, "Undercover: Authentication Usable in Front of Prying Eyes," CHI 2008, April 2008, pp. 183-192.

[4] A. D. Luca, E. V. Zezschwitz, and H. Hußmann, "VibraPass-Secure Authentication Based on Shared Lies," CHI 2009, April 2009, pp. 913-916.

[5] A. Bianchi, I. Oakley, J. K. Lee, and D. S. Kwon, "The Haptic Wheel: Design & Evaluation of a Tactile Password System," CHI 2010, April 2010, pp. 3625-3630.

[6] A. D. Luca, K. Hertzschuch, and H. Hussmann, "ColorPIN – securing PIN entry through indirect input," CHI 2010, April 2010, pp.1103-1106.

[7] Parallax barrier, http://en.wikipedia.org/wiki/Parallax_barrier. [retrieved: May, 2013]

[8] Q. Yang, J. Han, Y. Li, and R. Deng, "On limitations of designing leakage-resilient password systems: attacks, principles and usability," The 19th Annual Network & Distributed System Security Symposium (NDSS 2012), Feb. 2012.

[9] G. A. Miller, "The magical number seven, plus or minus two: some limits on our capacity for processing information," Psychological Review, vol. 63, 1956, pp. 81–97.

[10] G. A. Alvarez and P. Cavanagh, "The capacity of visual short-term memory is set both by visual information load and by number of objects," Psychological Science, vol. 15, no. 2, 2004, pp. 106–111.

 ISBN: 978-1-61208-298-1

Privacy-Preserving SVM Classification using Non-metric MDS

Khaled Alotaibi and Beatriz de la Iglesia
School of Computing Sciences
University of East Anglia, Norwich, UK
{K.Alotaibi,B.Iglesia}@uea.ac.uk

***Abstract*—Privacy concerns are a critical issue in outsourcing data mining projects. Data owners are often unwilling to release their private data for analysis, as this may lead to data disclosure. One possible solution to address such concerns is to perturb the original data values so that they become hidden, thereby preserving privacy. This paper proposes a privacy-preserving technique using Non-metric Multidimensional Scaling, which not only preserves privacy but also maintains data utility for Support Vector Machine (SVM) classification. The perturbed data are subject to high uncertainty and have no information that can be exploited to disclose the original data. They also exhibit better class separation and compactness, which greatly eases the SVM task. The results show that the accuracy of the original and the perturbed data is similar, as the distances between the data objects both before and after the perturbation are well-preserved.**

***Keywords*—Privacy; Classification; Data Perturbation; SVM.**

I. INTRODUCTION

Multidimensional scaling (MDS) is a dimensionality reduction technique used to project data into a lower dimensional space, so that better data visualisation can be achieved [1]. The basic idea of the MDS technique is as follows: Given a matrix of proximities (similarities or dissimilarities) between data objects, find a configuration of data points (usually in a lower dimensional space) whose distances fit these proximities best. In non-metric MDS, the interpoint distances between the data points in the new space approximate a non-linear transformation derived from those proximities. The main motivation for using non-metric MDS as a data perturbation technique is that it has the ability to generate data in a new space thereby hiding the original data as the data points are located in their positions using only the rank-order distances so their original position cannot be inferred from the perturbed data. Furthermore, each data object is represented by completely different data values and each data variable has a different pdf.

The concept of "data perturbation" refers to transforming data, and thereby concealing any private details whilst preserving the underlying probabilistic properties, so that the inherent patterns can still be accurately extracted. However, in real cases, achieving these two objectives is a challenging task, as they are naturally in conflict. In this paper, we investigate whether our privacy model proposed in [2] preserves data utility for SVM classification whilst maintaining privacy. We distort the original data values using non-metric MDS transformation. Then, we use the perturbed data to carry out the classification analysis using SVM with three kernels–linear, polynomial and radial basis function, and show that the results are similar (if not better) to those obtained from the original data.

The structure of this paper is as follows. Section II presents some related literature. Section III introduces an overview of SVM. Section IV presents the proposed privacy-preserving data mining method, and discusses its privacy and utility preservation. Experimental results are introduced in Section V. Finally, Section VI presents our conclusion.

II. RELATED WORK

Most works on data perturbation are based on linear transformations using additive or multiplicative noise. Additive perturbation was designed for microdata protection where a random noise is added to the value of each attribute to produce new data values representing the perturbed data [3]. Multiplicative perturbation can provide, to some extent, a good data utility for data mining algorithms. Here, the basic idea is to multiply the original data matrix by either a rotation matrix or a projection matrix. Chen and Liu [4] proposed a rotation-based perturbation technique that generates the perturbed data by multiplying the original data with an orthogonal rotation matrix. They showed that the SVM classifier with the most popular kernels (polynomial, radial basis and neural network) is invariant to rotation transformation. Similarly, [5] suggests a geometric perturbation technique where extra components are added to the rotation model. These components are a random translation matrix and addition of noise so more data protection can be achieved while preserving the basic geometric properties of the data for SVM classification. In [6], [7], the original data are projected into a lower dimension using random projection matrix. This perturbation model was also performed using a PCA-based approach [8].

The drawback of additive perturbation is that the added noise will distort the distances between data points and therefore poor results will be obtained when applying data mining algorithms on the perturbed data. Furthermore, the additive noise can be filtered out and the privacy can then be compromised [9], [10]. Although multiplicative perturbation can provide a better solution to overcome the shortcomings of additive perturbation, the privacy model is not secure enough. The attacker can exploit some theoretical properties of the random matrices (they usually have a predictable structure) to disclose the original data values [11], [12].

Our work, in this paper, is categorised as a perturbation-based approach, where all data are distorted before they are released to a third party for analysis. However, unlike other methods, the proposed method preserves much of the statistical properties for the classification task using SVM and provides perfect data protection as the perturbed data are generated under a high level of uncertainty.

 ISBN: 978-1-61208-298-1

III. OVERVIEW OF SVM

The SVM is a distance-based learning approach that is widely used in data classification [13]. The basic idea is to find a hyperplane that separates the data into two classes with as great a margin as possible. The optimal hyperplane (decision boundary) is the one that separates these two classes and that maximizes the distance between the two closest points from either class (known as *support vectors*). Assume that the classes of data are separable. Consider a binary classification problem consisting of m pairs of training examples $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$, where $x_i \in \mathbb{R}^n$ and $y_i \in \{-1, 1\}$; the hyperplane is defined by

$$\mathbf{w} \cdot \mathbf{x} + b = 0, \tag{1}$$

where $\mathbf{w}$ is the weight vector and b is the bias. "." denotes the dot product in the feature space. Both parameters $\mathbf{w}$ and b must be chosen in such a way that the following two conditions are met:

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_i + b \geq 1 \ \text{ when } \ y_i = 1, \\ \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \ \text{ when } \ y_i = -1. \end{aligned} \tag{2}$$

The classification rule of an unseen test object x' is defined by

$$g(x') = sign(\mathbf{w} \cdot \mathbf{x}' + b). \tag{3}$$

Maximizing the distance from a point $\mathbf{x}$ to the hyperplane in (1) determines the optimal hyperplane which creates the maximal margin between the negative and positive training examples. The distance from a hyperplane $H(\mathbf{w}, b)$ to a given data point $\mathbf{x}_i$ is simply

$$d(H(\mathbf{w}, b), \mathbf{x}_i) = \frac{\mathbf{w} \cdot \mathbf{x}_i + b}{||\mathbf{w}||} \geq \frac{1}{||\mathbf{w}||}. \tag{4}$$

That is, SVM finds the hyperplane that maximizes the margin by minimizing the squared norm of the hyperplane

$$\begin{aligned} \min_{\mathbf{w}} \ \frac{1}{2} ||\mathbf{w}||^2 \\ \text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \, i = 1, 2, \ldots, m. \end{aligned} \tag{5}$$

For non-separable data, SVM can also deal with overlapping classes by maximizing the margin, allowing any misclassified data points to be penalised using a method known as the *soft margin* approach [14]. The misclassification bias can be defined by so-called *slack variables*, $\xi = \xi_1, \xi_2, \ldots, \xi_s$. Let $\xi_i \geq 0$; the constraints of the optimisation can be rewritten as

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_i + b \geq 1 - \xi_i \ \text{ when } \ y_i = 1, \\ \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 + \xi_i \ \text{ when } \ y_i = -1, \end{aligned} \tag{6}$$

and the learning task in SVM can be formalized as follows:

$$\begin{aligned} \min_{\mathbf{w}} \ \frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{l} \xi_i \\ \text{subject to } \begin{cases} y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \ i = 1, 2, \ldots, m, \\ \xi_i \geq 0, \ \Sigma \xi_i \leq C \end{cases} \end{aligned} \tag{7}$$

where the constant C is a regularisation parameter used to create a balance between a maximum margin and a small number of misclassified data points.

The SVM described so far finds linear boundaries in the input space. However, in many real problems, data may have non-linear decision boundaries, which would make finding a hyperplane that can successfully separate two overlapping classes a difficult task. One solution to this problem is to use the so-called *kernel trick*. The trick here is to transform the data X in d-dimensional input space into a higher D-dimensional feature space $\mathcal{F}$ (also known as *Hilbert* space), $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^D$ where $D \gg d$. This would make the overlapping classes separable in the new space $\mathcal{F}$. The transformation is performed via a kernel function K that satisfies Mercer's condition [15] so that better class separation can be achieved [16]. The function K can be defined by

$$K(\mathbf{u}, \mathbf{v}) = \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}), \tag{8}$$

where $\Phi : X \rightarrow \mathcal{F}$ and "." denotes the dot product in the feature space $\mathcal{F}$. By defining a proper K, we simply replace all occurrences of $\mathbf{x}_i$ in the SVM model with $\Phi(\mathbf{x}_i)$. That is, the feature space $\mathcal{F}$ is never explicitly dealt with, but rather we evaluate the dot product, $\Phi(\mathbf{x}_i) \ . \ \Phi(\mathbf{x}_j)$, directly using function K in the input space. Intuitively, computing only the dot product using K, in the feature space, is substantially cheaper than using the transformed attributes. For example, the Radial Basis Function (RBF) kernel unfolds into an infinite-dimension Hilbert space.

IV. DATA PERTURBATION

To disguise the original data values and provide unreal data values (synthetic data) that preserve as much as possible data properties for data mining task, we used the perturbation model proposed in [2], which is defined by some transformation T,

$$Y = T(X), \tag{9}$$

where $T : \mathbb{R}^n \rightarrow \mathbb{R}^p$ is a non-metric MDS transformation [17] such that

1) T preserves the rank ordering of the distances between objects in X and Y, i.e.

$$\begin{aligned} ||x_i - x_j|| < ||x_k - x_l|| \Longleftrightarrow \\ ||T(x_i) - T(x_j)|| < ||T(x_k) - T(x_l)||, \end{aligned} \tag{10}$$

and

2) T minimizes the sum of squared differences of the distances, i.e., it minimizes

$$\sum_{i,j} (||x_i - x_j|| - ||T(x_i) - T(x_j)||)^2. \tag{11}$$

For presentation convenience, we use different notation to distinguish between the dissimilarities in the original space, X, and the perturbed space, Y. The distances between points in Y are $||T(x_i) - T(x_j)|| = d_{ij}$. The above first condition (10) is satisfied through a monotonic function, f, that maintains a monotone relationship between the dissimilarities, δ_{ij}, in the original space, $\mathbb{R}^n$, and the distances, d_{ij}, in the lower space, $\mathbb{R}^p$, i.e., $d_{ij} = f(\delta_{ij})$. The estimates of point locations in the lower dimensional space should yield predicted distances, d_{ij}, between the points that "closely approximate" the observed dissimilarities, δ_{ij}. To quantify the discrepancy (the stress) and to find the best solution, the second condition (11) should be applied.

The monotone relationship is obtained by a non-linear approach (monotonic regression) which fits a non-linear function, $f : \delta_{ij} \mapsto d_{ij}$, and minimizes the stress, S. Let $M = m(m-1)/2$ be the number of possible dissimilarities, δ_{ij}, that can be calculated from the data matrix X. The stress is given by

$$S = \sqrt{\sum_{i,j}^{M} (\hat{d}_{ij} - d_{ij})^2 / \sum_{i,j}^{M} d_{ij}^2}, \qquad (12)$$

where $\hat{d}_{ij}$ (also known as *disparities*) are numbers representing a monotone least-square regression of d_{ij} on δ_{ij} . That is, the disparities are merely an admissible transformation of d_{ij}, chosen in optimal way, to minimize S over the data configuration matrix, Y.

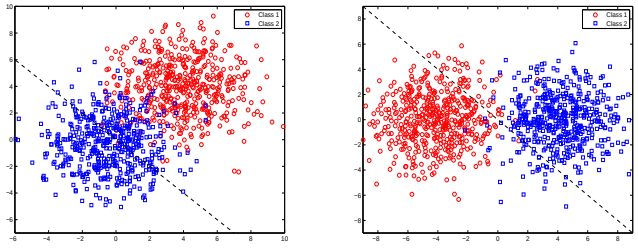
Non-metric MDS is quite similar to non-parametric procedures that are based on ranked data. The dissimilarities, δ_{ij}, are ranked by ordering them from lowest to highest and the disparities, $\hat{d}_{ij}$, should also follow the same monotonic ordering. This constraint implies the so-called *monotonicity* requirement

$$\text{if } \delta_{ij} < \delta_{kl} \text{ then } \hat{d}_{ij} \leq \hat{d}_{kl}. \qquad (13)$$

A. Data Utility Preservation

Non-metric MDS attempts to produce a new compact feature space with higher discriminative power [18]. In some senses, it may be considered similar to the kernel trick which generates a higher dimensional feature space to achieve better separation of the classes. It is expected that non-metric MDS can achieve good separation of negative examples from positive ones and as well as better class compactness (minimizing the intra-class distances while maximizing the inter-class separation) [19]. That is, we want to test whether the perturbed data, Y, can be as useful for SVM classification as the original data by measuring classification accuracy (or analogously the generalisation error) on both the original and the perturbed data.

The wider the margin between two groups of data, the better the SVM model will be at predicting the group for new instances. Figure 1 gives an insight into how non-metric MDS is able to discriminate two overlapping classes in a toy dataset, providing high data utility to the classification algorithm. In this example, the original dataset consists of 1,000 points in 4-dimensional space. We transform the data and generate a new dataset in 2-dimensional space. Then we train the SVM classifier on both datasets using a 50% training set, and test it on the remaining examples. As we expected, the perturbed data exhibit better class compactness and separation. It turns out that an optimal hyperplane that can successfully separate the two classes can be easily found in the lower dimensional space.



(a) Original data X (b) Perturbed data Y

Figure 1. Toy dataset, class separation in the original data X and the perturbed data Y using SVM classifier. For data X, the obtained hyperplane (dashed line) fails to separate the two classes whereas, for data Y, a better class separation has been achieved with a small relative error.

B. Privacy Preservation

For rigorous privacy analysis, we proposed a distance-based attack in order to disclose the location of a given point in the perturbed data using the technique of *Multilateration* [20], which uses knowledge about the location of $n+1$ data points. However, unlike [21], we do not require any prior knowledge beyond the known $n+1$ data points in the perturbed data. Then we show how this attack would fail to disclose the original data values because the perturbed data are subject to high uncertainty particularly in placing data points in the lower dimensional space.

1) Distance-Based Attack: Let $X \in \mathbb{R}^n$ be an $m \times n$ data matrix and x be unknown point for which we want to find the location. Given a set of $n+1$ known reference points, $R = \{r_1, r_2, \cdots, r_{n+1}\}$. Let d_{xr_i} be the Euclidean distance between the point x and each reference point r_i. The location of the point x is determined by minimizing

$$G(x) = \sum_{i=1}^{n+1} g_i(x)^2, \qquad (14)$$

where

$$g_i(x) = [(x_1 - x_{i1})^2 + (x_2 - x_{i2})^2 + \ldots + (x_n - x_{in})^2]^{1/2} - d_{xr_i}$$

is a non-linear function of n variables representing the coordinates of the point x. That is, we choose estimates $\hat{x_1}, \hat{x_2}, \ldots, \hat{x_n}$ that minimize $G(x)$. To solve this problem and find the minimum of the sum of squares, we use Gauss-Newton method which starts with a guess for x and iteratively move toward a better solution along the gradient of $G(x)$ until convergence.

 ISBN: 978-1-61208-298-1

Input: A set of $n+1$ known points, $r_1, r_2, \ldots, r_{n+1}$, an initial guess, x^0, a tolerance, $t > 0$, and a maximum number of iterations, $maxItr$.
Output: An estimation, $\hat{x}$, of the unknown point, x.

repeat
  Calculate $n+1$ distances, $d_{xr_1}, d_{xr_2}, \ldots, d_{xr_{n+1}}$, from the current x^k to each reference point, r;
  Evaluate $g_i(x^k)$ and $\nabla g_i(x^k)$ for $i = 1, 2, \ldots, n+1$;
  Move x^k a bit toward better location along the gradient, $x^{k+1} = ((A^k)^T A^k)^{-1}(A^k)^T b$;
  Calculate the error, err;
until *the error becomes less than the tolerance, $err < t$, or maximum number of iterations is exceeded, $k > maxItr$;*

Figure 2. Distance-Based Attack Algorithm.

If $g(x)$ is differentiable, then the refinement of the point x at iteration k can be achieved by the following linear approximation:

$$g(x) \approx g(x^k) + \nabla g(x^k)^T (x - x^k), \tag{15}$$

where $\nabla G(x^k)$ is the gradient (Jacobian) matrix that composes all first derivatives of x. To find x^{k+1} from x^k, we should minimize the sum of the squares of the linearized residuals, i.e.

$$\sum_{i=1}^{n+1} \Big(g_i(x^k) + \nabla g_i(x^k)^T (x - x^k) \Big)^2, \tag{16}$$

which is equivalent to solve the system $A^k x - b^k = 0$ which is defined by $(A^k)^T A^k x = (A^k)^T b^k$ and always consistent even when $A^k x = b^k$ is not consistent [22]. If A^k is non-singular, then there is a unique solution for x which represents the new position for the point x,

$$x^{k+1} = ((A^k)^T A^k)^{-1}(A^k)^T b. \tag{17}$$

To attack any given point, we develop a simple but effective search algorithm that can estimate the location of the unknown point while minimizing the sum of least-squares. The main steps are as follows: Start with an initial guess and move around in the direction where the relative error is minimized. The process is then repeated until convergence as described in Figure 2. The algorithm requires $O((n+1)^2 m)$ assuming that $m > n+1$.

To quantify the privacy for any given point x, we compute the ratio of the differences between x and its estimate $\hat{x}$ to the average distance from x to the $n+1$ known points $r_1, r_2, \cdots, r_{n+1}$, i.e.

$$\rho^* = \frac{||x - \hat{x}||}{\frac{1}{n+1} \sum_{j=1}^{n+1} ||x - r_j||}. \tag{18}$$

The overall privacy is then given by

$$\rho = \frac{1}{N} \sum_{i=1}^{N} \frac{||x_i - \hat{x}_i||}{\frac{1}{n+1} \sum_{j=1}^{n+1} ||x_i - r_j||}, \tag{19}$$

where N is the number of the remaining unknown points.

2) Uncertainty Measure: The process of placing points in our model is not straightforward, but rather it depends on preserving the order of dissimilarities. This implies that there is uncertainty about the exact location of any given point in the lower dimensional space, Y, and hence, a better protection against distance-based disclosure is achieved. To illustrate the basic idea, consider the following example. Let x be an unknown point and on distances d_{xr_1}, d_{xr_2} and d_{xr_3} from three other known points, r_1, r_2 and r_3, respectively. Assume that d_{xr_1}, d_{xr_2} and d_{xr_3} confirm the following order

$$d_{xr_1} < d_{xr_2} < d_{xr_3}.$$

To preserve the ordering (monotonicity), the point x should be placed somewhere within an area where the above order is satisfied. Assume that each point, here in this example, represents a single value, say salary. Assume also that $r_1 = 2K, r_2 = 5K$ and $r_3 = 7K$. If this information together with the distances from each point, r, to the point x are available to the attacker, she can guess that x is more likely to fall in an interval, say $[1K, 3K]$ with an assumption that the minimum salary is $1K$, as this would ensure the above ordering. Indeed, the probability that any attacked point locates within any given area is a measure of how well the original data are hidden. The probability P that the point x locates in area E, where $E \in \mathbb{R}$ is the domain of all possible outcomes, is

$$P(E) = \int_E f(x) dx. \tag{20}$$

This suggests that the probability of finding a given point x is inversely proportional to the area where the rank order is satisfied. The information available from the rank ordering would make the solution of non-metric MDS highly uncertain, as the points are not located to specific locations but rather to areas where the ordering is preserved. Therefore, the distance-based adversary attacks would fail to determine the exact location of the point.

V. EXPERIMENTS

For an empirical evaluation of training the SVM classifier on our privacy-preserving model, we use four numerical datasets collected from the UCI machine learning repository [23]. The datasets are Breast Cancer (699/9), Credit Approval (690/14), Pima Diabetes (768/8) and Hepatitis (155/19). All datasets have a binary class, i.e., positive and negative groups.

As we are not interested in the visual representation of data in the lower space, rather in achieving high data utility as far as possible, for some of the experiments the number of dimensions p was fixed to $n-1$, and the data projected into that lower dimensional space. Then, we used the generated data, Y, in p-dimensions, to carry out the SVM classification and to compare it with the results obtained from the original data, X.

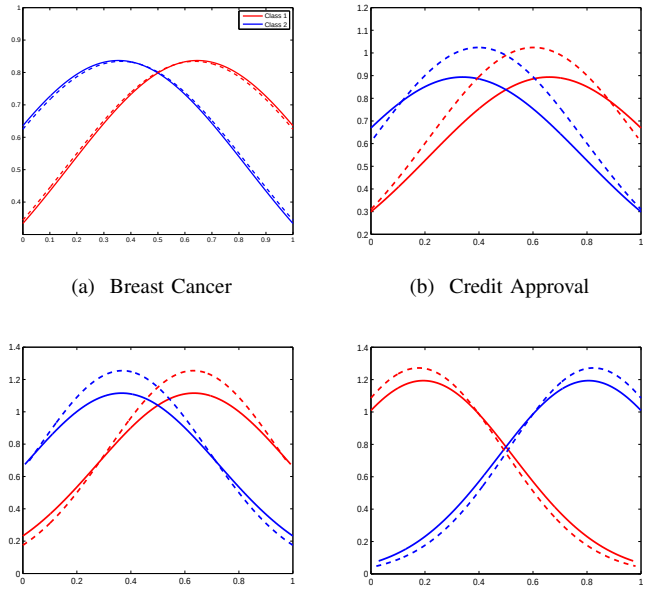
 ISBN: 978-1-61208-298-1

TABLE I. THE ACCURACIES OF LINEAR SVM ON THE ORIGINAL DATA, X, AND THE PERTURBED DATA, Y, AT REDUCED DIMENSIONS.

Dimension(n)	BreastCancer	CreditApproval	PimaDiabetes	Hepatitis
$X(n)$	0.9685	0.8623	0.7800	0.8099
$Y(n-1)$	0.9692	0.8638	0.7698	0.8314
$Y(n-2)$	0.9557	0.8584	0.7693	0.7930
$Y(n-3)$	0.9749	0.8725	0.7396	0.7721
$Y(n-4)$	0.9718	0.8649	0.7464	0.7721
$Y(n-5)$	0.9690	0.8557	0.7326	0.7876
$Y(n-6)$	0.9618	0.8630	0.7235	0.7876

To perturb data X and perform the classification, we used an implementation in Matlab. The dissimilarities, δ_{ij}, between the objects in X were first calculated. Then, we transformed the dissimilarities and generated Y. The initial configuration was chosen randomly. The stress S (12) was used as a data utility measure, as it determines the size of change in the interpoint distances in data Y as a result of the transformation.

We evaluated the classification accuracy on both the original and the perturbed data. 10-fold cross-validation was performed and the error rates of the testing set were evaluated for both data. The regularisation parameter, C, was set to 1 in all experiments because, in this set of experiments, our main concern is not to get an optimal SVM model but rather to compare the SVM models applied on the original and the perturbed data. Table I summarises the average accuracies of all the datasets at different dimensions using a linear SVM. For Pima Diabetes dataset, the accuracy on both the original and the perturbed data is lows because the data have imbalanced classes which inherently biased toward the majority concept. However, the difference in accuracy at the $n-1$ dimensional space was still plausible (0.01) exhibiting low distortion.

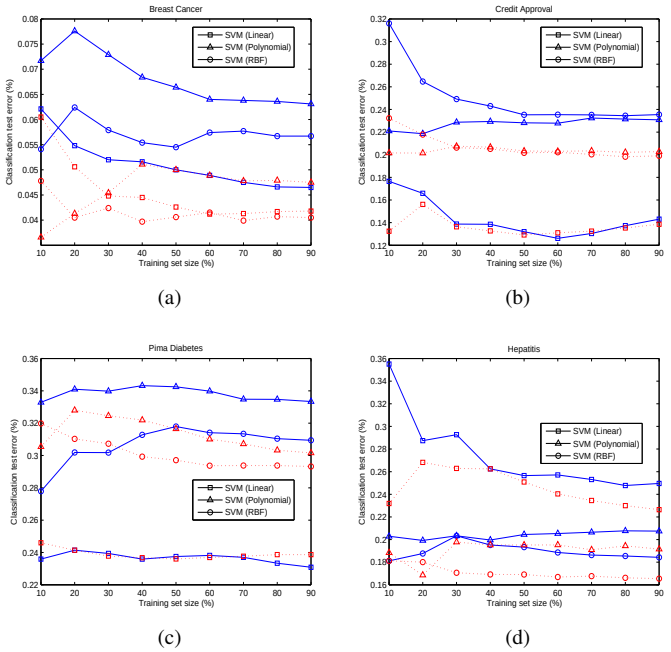


(a) Breast Cancer (b) Credit Approval

(c) Pima Diabetes (d) Hepatitis

Figure 3. An estimation of the classes' posterior probability in the original data (solid lines) and the perturbed data (dashed lines).

To show the usefulness of the perturbed data in terms of data utility, we assume independence of the variables and plot the distribution of the estimated posterior probability of assigning an object x to a class C_i. From naïve Bayes theorem with strong (naive) independence assumptions, the posterior probability is $P(C_i|x) = P(x|C_i)P(C_i)/P(x)$. This would help in estimating the quality of the classification on the perturbed data in comparison with the original data because it may give an insight of the classes overlapping in the original and in the perturbed space. As the outputs of the classifier are expected to approximate the corresponding posteriori class probabilities if it is reasonably well trained, $P(C_i|x)$ may also help to discover any potential decision boundaries that are typically expected to be close to Bayesian decision boundaries [24]. The results are shown in Figure 3. The distributions of classes before and after the perturbation almost coincide for all datasets. For the Hepatitis dataset, the separation of classes in both perturbed and original data appears to be better so an optimal hyperplane should be easily found. For instance, the accuracy on the perturbed data at the $n-1$ dimensional was 2% better than on the original data.



(a) (b)

(c) (d)

Figure 4. Classification error of SVM (with different kernels linear, polynomial and radial basis function) using training set with different sizes on the original data (solid lines) and the perturbed data (doted lines).

To evaluate the generalisation error when the SVM classifier is trained using a different training set size, we split each dataset into a training set containing $10\%, 20\%, \ldots, 90\%$ of the samples and a testing set containing the remaining corresponding samples. The results are depicted in Figure 4. The SVM error rates on the perturbed data were markedly lower, suggesting improved performance of the classifier on the perturbed data compared with the original data.

Finally, to assess the privacy of our model, we transformed the data into 6 different dimensions and attempted to estimate the original data values using distance-based attack. The number of known points was also incrementally varied to see its effect on the accuracy of locating unknown points. The average privacy of each dataset is shown in Figure 5. The results

 ISBN: 978-1-61208-298-1

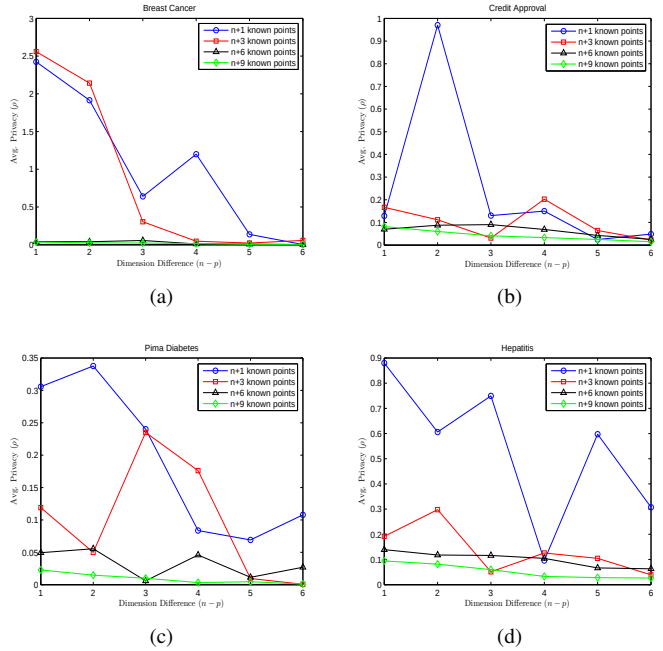


Figure 5. Average privacy (ρ) at different dimensions using different numbers of the known points.

indicate more resistance to the attack, especially at higher dimensions, and also confirm that the error of determining the location of an unknown point increases when the number of dimensions increases. That is, transforming the data into the few lower dimensions gives reasonable utility and privacy.

VI. Conclusion

The experiment results confirm that the perturbation method using non-metric MDS can provide high uncertainty for privacy preservation without affecting the accuracy of the SVM model and hence the utility. The perturbed data are still good enough to provide for reasonable discrimination between classes for SVM, and in some cases the data in the lower dimensionality provides improved classification performance. The main advantages of our method are that the perturbed data are independent from the original data and subject to a high degree of uncertainty; only the rank-order of dissimilarities are non-linearly mapped into distances in the perturbed data. In other words, the attacker knows nothing about the mapping function. Finally, most privacy-preserving data mining methods attempt to modify the learning algorithms in order to protect the original values from disclosure. This could decrease the efficiency of the algorithms, compromising the quality of the results. In contrast, the proposed method allows one to apply the algorithms directly to the perturbed data without any modification.

References

[1] M. Cox and T. Cox, "Multidimensional scaling," Handbook of data visualization, 2008, pp. 315–347.

[2] K. Alotaibi, V. J. Rayward-Smith, W. Wang, and B. de la Iglesia, "Non-linear dimensionality reduction for privacy-preserving data classification," in Proceedings of IEEE Fourth International Conference on Privacy, Security, Risk and Trust (PASSAT 2012). IEEE, 2012, pp. 694–701.

[3] R. Agrawal and R. Srikant, "Privacy-preserving data mining," ACM Sigmod Record, 2000, vol. 29, no. 2, pp. 439–450.

[4] K. Chen and L. Liu, "Privacy preserving data classification with rotation perturbation," in Proceedings of the Fifth IEEE International Conference on Data Mining, 2005, pp. 589–592.

[5] K. Chen, G. Sun, and L. Liu, "Towards attack-resilient geometric data perturbation," in Proceedings of the 2007 SIAM Data Mining Conference. SDM'07, 2007, pp. 78–89.

[6] K. Liu, H. Kargupta, and J. Ryan, "Random projection-based multiplicative data perturbation for privacy preserving distributed data mining," IEEE Transactions on Knowledge and Data Engineering, 2006, vol. 18, no. 1, pp. 92–106.

[7] S. Oliveira and O. Zaïane, "Privacy-preserving clustering to uphold business collaboration: A dimensionality reduction based transformation approach," International Journal of Information Security and Privacy, 2007, vol. 1, no. 2, pp. 13–36.

[8] R. Banu and N. Nagaveni, "Preservation of data privacy using pca based transformation," in Proceedings of the International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom'09). IEEE, 2009, pp. 439–443.

[9] D. Agrawal and C. Aggarwal, "On the design and quantification of privacy preserving data mining algorithms," in Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM, 2001, pp. 247–255.

[10] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar, "Random-data perturbation techniques and privacy-preserving data mining," Knowledge and Information Systems, 2005, vol. 7, no. 4, pp. 387–414.

[11] S. Guo and X. Wu, "Deriving private information from arbitrarily projected data," Advances in Knowledge Discovery and Data Mining, 2007, pp. 84–95.

[12] K. Liu, C. Giannella, and H. Kargupta, "An attackers view of distance preserving maps for privacy preserving data mining," Knowledge Discovery in Databases, 2006, pp. 297–308.

[13] H. Drucker, D. Wu, and V. Vapnik, "Support vector machines for spam categorization," IEEE Transactions on Neural Networks, 1999, vol. 10, no. 5, pp. 1048–1054.

[14] K. Veropoulos, C. Campbell, and N. Cristianini, "Controlling the sensitivity of support vector machines," in Proceedings of the international joint conference on artificial intelligence, 1999, pp. 55–60.

[15] V. Vapnik, "The support vector method of function estimation," Nonlinear Modeling: Advanced Black-Box Techniques, 1998, vol. 55, pp. 55–85.

[16] T. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," IEEE Transactions on Electronic Computers, 1965, no. 3, pp. 326–334.

[17] J. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," Psychometrika, 1964, vol. 29, no. 1, pp. 1–27.

[18] T. Cox and G. Ferry, "Discriminant analysis using non-metric multidimensional scaling," Pattern Recognition, 1993, vol. 26, no. 1, pp. 145–153.

[19] A. Gorban and A. Zinovyev, "Principal manifolds and graphs in practice: From molecular biology to dynamical systems," International Journal of Neural Systems, 2010, vol. 20, no. 3, pp. 219–232.

[20] W. Navidi, W. Murphy, and W. Hereman, "Statistical methods in surveying by trilateration," Computational statistics & data analysis, 1998, vol. 27, no. 2, pp. 209–227.

[21] E. Turgay, T. Pedersen, Y. Saygın, E. Savaş, and A. Levi, "Disclosure risks of distance preserving data transformations," in Scientific and Statistical Database Management. Springer, 2008, pp. 79–94.

[22] C. Meyer, Matrix analysis and applied linear algebra. Society for Industrial Mathematics, 2000, no. 71.

[23] A. Frank and A. Asuncion, "UCI machine learning repository," 2010, university of California, Irvine, School of Information and Computer Sciences. [Online]. Available: http://archive.ics.uci.edu/ml

[24] K. Tumer and J. Ghosh, "Analysis of decision boundaries in linearly combined neural classifiers," Pattern Recognition, 1996, vol. 29, no. 2, pp. 341–348.

 ISBN: 978-1-61208-298-1

Security and Confidentiality Solutions for Public Cloud Database Services

Luca Ferretti, Fabio Pierazzi, Michele Colajanni, and Mirco Marchetti
Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia, Italy
e-mail: {luca.ferretti, fabio.pierazzi, michele.colajanni, mirco.marchetti}@unimore.it

***Abstract*—The users perception that the confidentiality of their data is endangered by internal and external attacks is limiting the diffusion of public cloud database services. In this context, the use of cryptography is complicated by high computational costs and restrictions on supported SQL operations over encrypted data. In this paper, we propose an architecture that takes advantage of adaptive encryption mechanisms to guarantee at runtime the best level of data confidentiality for any type of SQL operation. We demonstrate through a large set of experiments that these encryption schemes represent a feasible solution for achieving data confidentiality in public cloud databases, even from a performance point of view.**

Keywords-Cloud; Database; Confidentiality; Adaptivity; Encryption

I. INTRODUCTION

The Database as a Service (DBaaS) [1] is a novel paradigm through which cloud providers offer the possibility of storing data in remote databases. The main concerns that are preventing the diffusion of DBaaS are related to data security and confidentiality issues [2]. Hence, the main alternative seems the use of cryptography, which is an already adopted solution for files stored in the cloud, but that represents an open issue for database operations over encrypted data.

Fully homomorphic encryption theoretically supports any kind of computation over encrypted data [3], but it is computationally unfeasible, because it increases the computational cost of any operation by many orders of magnitude. Other schemes which allow the execution of computations over encrypted data limit the type of allowed operations (e.g., order comparions in [4], sums in [5], search in [6]). Although these methods were successfully deployed in some DBaaS contexts [7], they require the anticipatory choice of which encryption scheme can be used for each database column and for a specific set of SQL commands.

In this paper, we propose a cloud database architecture based on *adaptive encryption techniques* [8] that encapsulate data through different layers of encryption. This adaptive encryption architecture is attractive because it does not require to define at design time which operations are allowed on each column, and because it can guarantee at runtime the maximum level of data confidentiality for different SQL operations. Unfortunately, this scheme is affected by high computational costs. However, through a prototype implementation of an encrypted cloud database, we show that adaptive encryption can be well applied to a cloud database paradigm, because most performance overheads are masked by network latencies. This study represents the first performance evaluation of adaptive encryption methods applied to cloud database services. Other experiments [8] assumed a LAN scenario and no network latency.

The paper is structured as follows. Section II describes the proposed adaptive encryption scheme for cloud database architectures. Section III presents the results of the experimental evaluations for different network scenarios, workload models and number of clients. Section IV outlines main conclusions and possible directions for improvement.

II. ARCHITECTURE

We describe the architecture we propose to guarantee data confidentiality through adaptive encryption methods in cloud database environments.

A. Architecture model

We refer to the distributed architecture represented in Fig. 1, where we assume that independent and distributed clients (Client 1 to N) access a public cloud database service [9]. All information (i.e., data and metadata) is stored encrypted in the cloud database. The proposed architecture manages five types of information.

- *plain data*: the informative content provided by the client users.
- *encrypted data*: the encrypted data that are stored in the cloud database.
- *plain metadata*: all the information required by the clients to manage encrypted data on the cloud database.
- *encrypted metadata*: the encrypted metadata that are stored in the cloud database.
- *master key*: the encryption key of the encrypted metadata. We assume that it is distributed to all legitimate clients.

A legitimate client can issue SQL operations (SELECT, INSERT, UPDATE, DELETE) to the encrypted cloud database by executing the following steps. It retrieves encrypted metadata, and obtains plain metadata by decrypting them through the master key. The metadata are cached locally in a volatile representation that is used for improving performance. Then, the client can issue SQL operations over the encrypted data (i.e., the real informative content), because it is able to encrypt the queries, their parameters, and decrypt their results by using the local plain metadata.

This architecture guarantees confidentiality of data in a *security model* in which the WAN network is untrusted (malicious), while client users are trusted, that is, they do not reveal any information about plain data, plain metadata, and

 ISBN: 978-1-61208-298-1

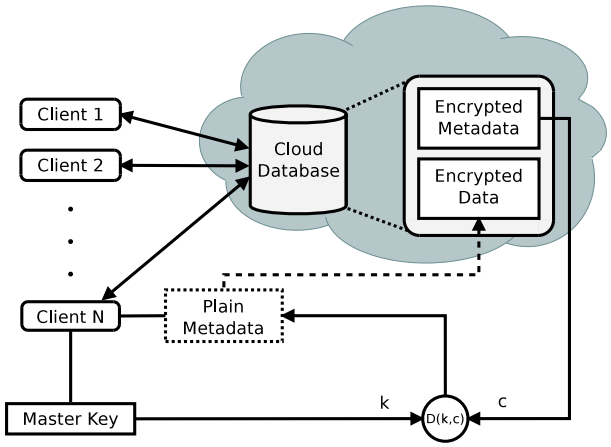


Figure 1. Cloud database architecture

the master key. The cloud provider administrator is *semi-honest* [10] (also called *honest-but-curious*), because he could try accessing information stored in the database, but he does not modify internal data and SQL operations results.

B. Adaptive encryption techniques

We consider SQL-aware encryption algorithms that guarantee data confidentiality and allow the cloud database server to carry out a large set of SQL operations over encrypted data. Each algorithm supports a specific subset of SQL operators. This paper refers to the following encryption schemes.

Deterministic (Det): it deterministically encrypts data, so that the encryption of an input value always guarantees the same output value. It supports the equality operator.

Order Preserving Encryption (OPE) [4]: this encryption scheme preserves in the encrypted values the numerical order of the original unencrypted data. It supports the following SQL operators: equal, unequal, less, less or equal, greater, greater or equal.

Sum: this encryption algorithm is homomorphic with respect to the sum operation: summing unencrypted data is equivalent to multiplying the correspondent encrypted values. It supports the sum operator between integer values.

Search: it supports equality check on full strings (i.e., the *LIKE* operator) that do not include fragments of words.

Random (Rand): it is a semantic secure encryption (IND-CPA) that does not reveal any information of the original plain value. It does not support any SQL operator.

Plain: a special kind of "encryption" that leaves values unencrypted. It supports all SQL operators, and it is included to store publicly available data, or some anonymous values that do not require any data confidentiality.

If each column data was encrypted through only one of these algorithms, the database administrator would have to decide at the design time which operations must be supported on each database column. This assumption is impractical in most cases. Hence, we need to define adaptive schemes that allow our architecture to support at runtime the SQL operations issued by the clients, while preserving a high level of confidentiality on the columns that are not involved in any operation. For this reason, we organize the encryption schemes into structures called *Onions*. Each Onion is composed by different encryption algorithms, called *(Encryption) Layers*, one above the other. Outer Layers guarantee higher data confidentiality and lower number of allowed operations, and each Onion supports a specific set of operators. When additional SQL operations are to be executed on a column, the outer Layers are dynamically decrypted. In this paper, we consider and design the following Onions, which are also represented in Fig. 2.

Onion-Eq: it manages the equality operator.

Onion-Ord: it manages the following operators: less, less or equal, greater, greater or equal, equal, unequal.

Onion-Sum: it manages the sum operator.

Onion-Search: it manages the string equality operator.

Onion-Single-Layer: a special type of Onion that supports only a single Encryption Layer. It is recommended for columns in which operations to be supported are known at design time.

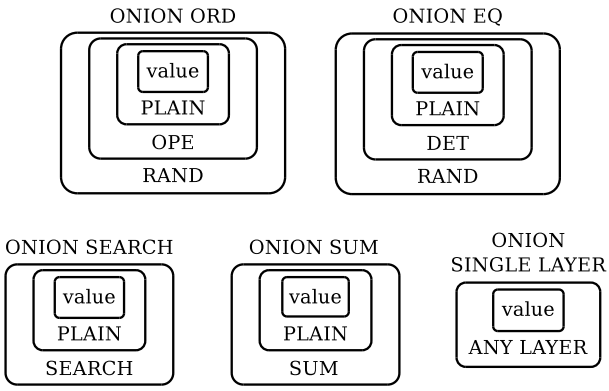


Figure 2. Onions and layers structure

In our architecture, each plain database column is encrypted into one or more encrypted columns, each one corresponding to a different Onion, depending on the SQL operations that must be supported on that column. The most external Encryption Layer of an Onion is called *Actual Layer*, which by default corresponds to its strongest encryption algorithm.

Each data type is characterized by a default set of supported Onions, depending on the operations supported by the data type and the compatibility between the encryption algorithms and the data type itself. Each database column can be defined through three parameters: column *name*, *data type*, and *confidentiality parameters*. The confidentiality parameters of a column define the set of Onions to be associated with it, and their starting Actual Layers. The Onions associated to a column must be compatible with the column data type. For example, integer columns can be associated to Onion-Eq, Onion-Ord and Onion-Sum, because integer values support equality checks, order comparisons and sums, but they cannot be associated to Onion-Search, which manages the string equality operator. At the time of a table creation, the database administrator (DBA) has the possibility to specify only a column's name and data type, as in normal relational databases, because our architecture can automatically choose

 ISBN: 978-1-61208-298-1

the default set of Onions with regard to the column data type. However, the DBA can also manually specify the confidentiality parameters of a column, when the SQL operations to be supported on the column are known at design time.

Fig. 3 represents an example of the Onions in the structure of the encrypted database table. Each column's Onion corresponds to a different encrypted column. We have two columns: an integer column *id*, with Onion-Eq and Onion-Ord, and a string column *name*, with Onion-Search. We observe that the representation of the encrypted table in this figure is just for clarity, because in the real implementation the table and the columns names should be encrypted too.

Table 1 (Plain)

id	name
1	Mark
3	Luke
4	John

Table 1 (Encrypted)

id-OnionEq	id-OnionOrd	name-OnionSearch
x4cx52	x2bx3xc2	xz53j2hzfap3hx
xd2vsd	xbcv3b3f	x34k2x3243mgj3
x34ds2	x4nj3h3x	x45h23x3cxfhx2

Figure 3. Onions in the encrypted database

The main benefit of the Onions is to allow our architecture to adapt the level of data confidentiality to the current SQL workload by decrypting an encrypted column's outer Layer(s). In such a way, it supports at runtime any SQL operation issued by a user. We refer to the Onion adaptation process as the *automatic column re-encryption*. The proposed architecture is designed so that the column re-encryption is executed on the cloud database through User Defined Functions (i.e., stored procedures) that, when required, are automatically invoked by the clients. Only trusted clients that know decryption keys can invoke column re-encryption. For security reasons, they cannot request any column re-encryption that would expose the Plain Layer of an Onion. Hence, all information stored in the cloud database is always encrypted, and the cloud provider does not have access to plain data.

The two main phases involved in the column re-encryption operation are *re-encryption invocation* on the client side and the *re-encryption execution* on the cloud database side. We describe these two phases with reference to Fig. 4.

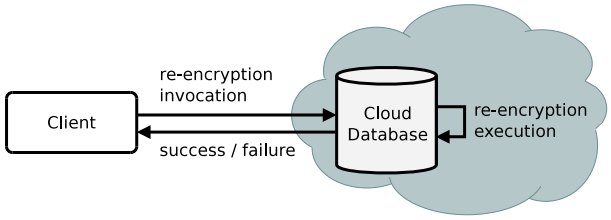


Figure 4. Automatic column re-encryption

In the **re-encryption invocation** phase, the client examines the plaintext query issued by the user (which can also be an external application) and evaluates whether the involved SQL operators (e.g., equality checks and order comparisons) are supported with respect to the Actual Layers of the Onions available on the involved columns. If it is necessary to adjust the Actual Layer of one or more Onions in order to support the operators, the client issues a request for re-encryption to the cloud database through a stored procedure invocation. Only trusted clients know the decryption key that is required by the stored procedure to decrypt the outer Layer of the Onion. The invocation phase is repeated for each column that requires re-encryption.

In the **re-encryption execution** phase, the cloud database engine executes a properly defined stored procedure that diminishes the Actual Layer of an Onion by decrypting its row values one by one. After the stored procedure execution, the cloud database sends the information about its outcome (success or failure) to the client that issued the request for re-encryption. We observe that any new execution of the same SQL operator on the column does not require to invoke the re-encryption process again, because the cloud database does not encrypt the Onion back to the upper Layer.

For example, let us consider a client that issues an SQL operation involving an equality check on a column that has only Onion-Eq with Actual Layer Rand. The Rand Encryption Layer does not support any operator, and therefore the equality check could not be performed on Rand. In the proposed architecture, a client is able to understand whether the issued query involves an equality check, and therefore it can automatically issue to the cloud database a request to lower down the Actual Layer of Onion-Eq from Rand to Det (re-encryption invocation), which supports the equality operator. After the re-encryption execution, the cloud database sends to the client the information about the outcome of the re-encryption.

We observe that the automatic column re-encryption mechanism is the most critical part of the proposal in terms of deployment and performance. The deployment problem is that the proposed solution requires to install some encryption libraries on the cloud database, in order to allow the stored procedures to be able to decrypt values with the proper encryption schemes (the same used by the clients). Compatibility with possibly any cloud database can be achieved by designing a column re-encryption mechanism that may be disabled anytime. This solution preserves compatibility at the price of losing adaptivity. The performance problem is that re-encryption increases the response time for operations requiring column re-encryption, especially if the column to be re-encrypted consists of a large number of rows. In highly dynamical contexts in which the SQL workload changes frequently, this overhead may be severe.

C. Discussion

The proposed data confidentiality architecture is inspired by the solutions presented in [8] and [7]. Nevertheless, this is the first that allows to leverage adaptive encryption mechanisms while avoiding the use of any intermediate (trusted) proxy server to manage encryption details.

There are several benefits characterizing the proposed architecture. It guarantees confidentiality of information stored in the cloud database, while allowing the execution of SQL operations over encrypted data. It simplifies database configuration, because it does not require to manually define

 ISBN: 978-1-61208-298-1

at design time which operations should be allowed on each column. It guarantees best level of data confidentiality for any SQL workload, thanks to the automatic column re-encryption mechanism. It does not require any intermediate (trusted) proxy to manage encryption details.

We observe that adaptive encryption is also affected by two major drawbacks. The first problem is that each plain column must be encrypted into one or more encrypted columns (Onions), thus increasing the overall database size up to one order of magnitude. This cost may be considered acceptable, or it can be reduced by the database administrator through a suitable tuning of the confidentiality parameters. The second problem is the performance overhead characterizing adaptive encryption, that has to encrypt all parameters and decrypt the results of every SQL operation through all the Encryption Layers of each involved Onion. These costs prevent the use of adaptive encryption methods on most real contexts. However, in Section III we show that this overhead becomes less significant when an encrypted database is used in the cloud, because in these scenarios realistic network latencies tend to mask the CPU time of expensive operations.

III. Performance Evaluation

A. Experimental Testbed

We design a suite of performance tests in order to evaluate the impact of adaptive encryption methods on response times and throughput for different network latencies (from 0 to 120 ms) and number of clients (from 5 to 20). The experiments are carried out in Emulab [11], which provides us with a set of machines in a controlled LAN environment. The TPC-C standard benchmark is used as the workload model for the database services.

In Emulab, we design and implement the testbed as a simulated network that connects 20 clients with one server. Each client machine runs the Python client prototype of our architecture on a pc3000 machine having single 3GHz processor, 2GB of RAM and two 10,000 RPM 146GB SCSI disks. The server machine hosts a database server implemented in PostgreSQL 9.1 on a d710 machine having a quad-core Xeon 2.4 GHz processor, 12GB of RAM and a 7,200 RPM 500GB SATA disk. Each machine runs a Fedora 15 image.

The current version of the prototype supports the main SQL operations (SELECT, DELETE, INSERT and UPDATE) and the WHERE clause expressions. The prototype integrates the following encryption algorithms (see Section II).

- *Deterministic*: implemented through the standard AES algorithm [12] in CBC mode using a constant initialization value (PyCrypto 2.6 library);
- *Random*: implemented through the standard AES algorithm in CBC mode using a random initialization value (PyCrypto 2.6 library);
- *OPE*: implementation based on the OPE algorithm proposed by [4] and used by [8].

Hence, we implemented the following Onions: Onion-Eq, Onion-Ord and Onion-Single-Layer.

In the PostgreSQL database server, the stored procedures required for column re-encryption (see Section II) are implemented with the *PL/Python* and the *PL/pgSQL* languages. The PyCrypto 2.6 library has been installed in the database server machine in order to support Rand Layer decryption, by using the same functions defined in the client prototype.

In the experiments, we consider three TPC-C compliant databases having ten warehouses and a scale factor of five.

- *PostgreSQL Plain*. This database contains plaintext data.
- *Encrypted*. This configuration refers to an encrypted database where each column is defined as an Onion-Single-Layer. This configuration improves system performance, but does not guarantee adaptive encryption, because each data is encrypted through only one Encryption Layer. Plain, Det, OPE and Rand algorithms are used.
- *Adaptive Encryption*. This database integrates Onion-Eq, Onion-Ord and Onion-Single-Layer.

In the encrypted databases, each column is set to the highest Encryption Layer required to support the SQL operations from the TPC-C workload. In the adaptively encrypted database, we consider the default configuration in which each column is encrypted into all the Onions supported by its data type.

As expected, adaptive encryption schemes increment the overall encrypted database size, but we consider acceptable a larger database size if it guarantees data confidentiality in a cloud environment. In particular, the size of the plaintext database is 490MB, and the size of the adaptively encrypted database is about 4.3GB.

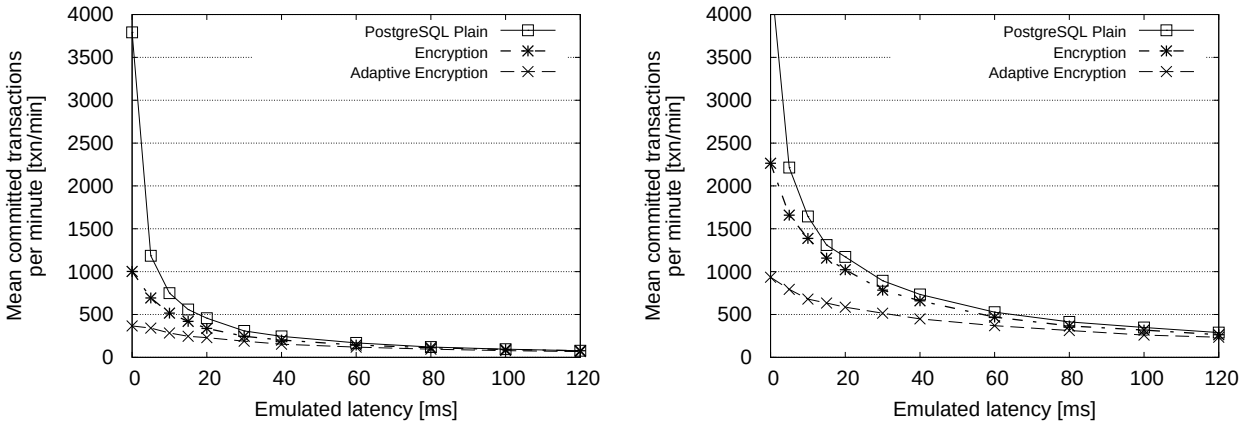
During each TPC-C test lasting for 300 seconds, we monitor the number of executed TPC-C transactions, and the response times of all the SQL operations from the standard TPC-C workload. We repeat the test for each database configuration (plain, encrypted, and adaptively encrypted), for increasing number of clients (from 5 to 20), and for increasing network latencies (from 0 to 120). In order to guarantee data consistency, the three databases have been set to repeatable read (snapshot) isolation level [13]. Since the standard TPC-C workload does not modify the database structure, the experiments consider a situation in which the Onions Actual Layers do not change dynamically at runtime.

B. Experimental results

The experiments aim to evaluate the overhead caused by adaptive encryption and network latencies in terms of system throughput and response time.

In Fig. 5, we report the number of committed TPC-C transactions per minute executed on the three cloud databases. The results refer to 5 and 20 active clients (Fig. 5a and Fig. 5b, respectively) for different network latencies. We can appreciate that in both cases (and in others not reported for space reasons) the throughput of the encrypted database is close to that of the plain database. Moreover, as the network emulated latency increases, even the performance of the database with adaptive encryption tend to that of the other two systems, and it is close to that for latencies higher than 60ms, which are realistic for typical cloud database scenarios. This is an extremely positive

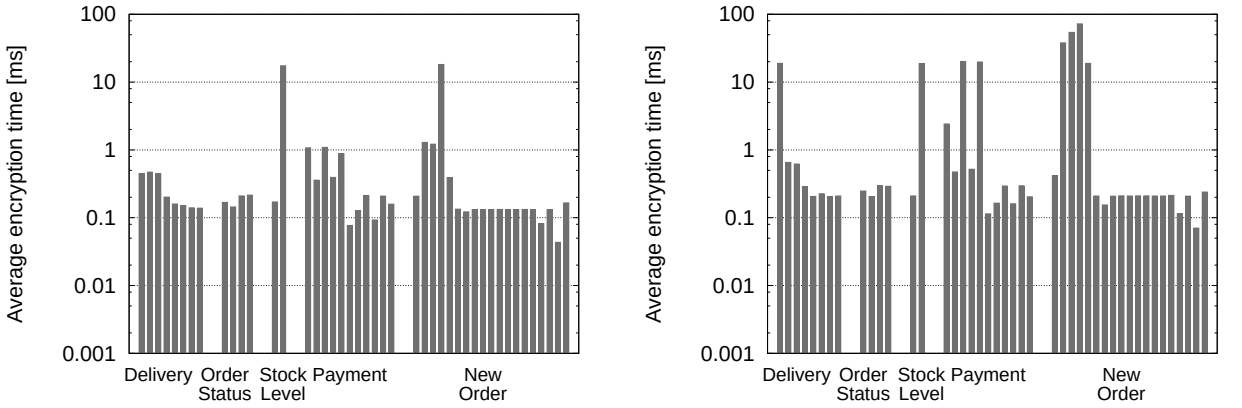
 ISBN: 978-1-61208-298-1



(a) Mean committed txn/m - 5 clients

(b) Mean committed txn/m - 20 clients

Figure 5. Throughput of the three types of cloud database



(a) Non adaptive configuration

(b) Adaptive configuration

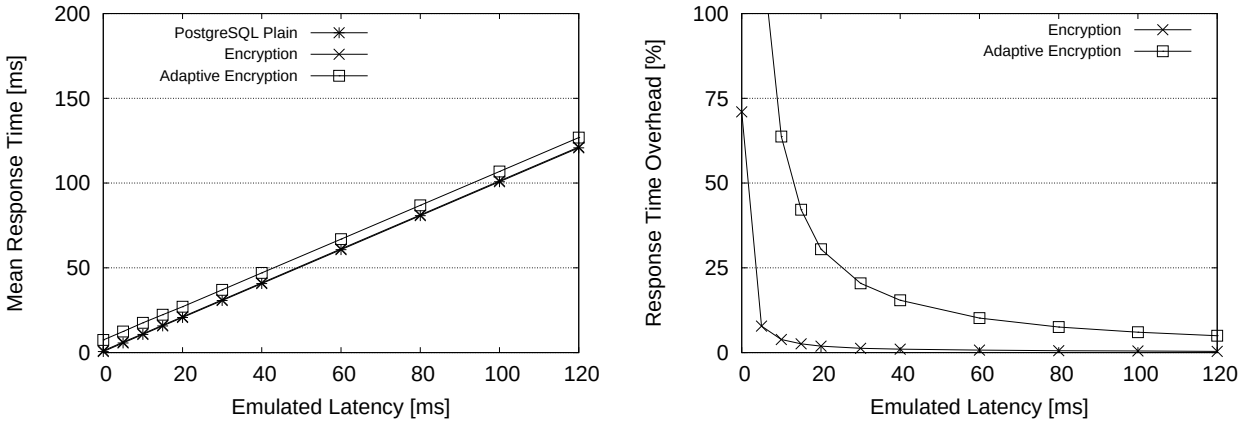
Figure 6. Average encryption times grouped by TPC-C transactions

result, because it demonstrates that adaptive encryption can be realistically used in cloud database services. We observe that the database throughput increases proportionally as a function of the number of clients, because in our experiments the clients do not saturate the database capacity.

In Fig. 6, we report two histograms in which we represent the mean encryption times required by each SQL operation composing the TPC-C workload. These results refer to the mean time that is required by a client to encrypt the parameters involved in each SQL operation of the TPC-C workload. The results are grouped on the basis of classes of TPC-C operations. The histogram in Fig. 6a considers the mean encryption times in the non adaptive configuration, while the other histogram in Fig. 6b considers the adaptive configuration. We can observe how some SQL operations in the adaptive configuration require higher encryption times with respect to those of the non adaptive case. Further analyses show that the two peaks in the non adaptive configuration (Fig. 6a) are related to SQL operations which require OPE encryption, and that most of the additional peaks in the adaptive configuration (Fig. 6b) are related to INSERT operations combined with OPE encryption. The OPE algorithm requires an encryption time that is two or three orders of magnitude higher than that related to Rand and Det algorithms [8]. In the adaptive encryption configuration, if the database administrator does not specify otherwise, every integer column is associated by default with Onion-Eq and Onion-Ord (which has an OPE Layer). Hence, every insertion of a value into an integer column requires an OPE encryption, and this increases significantly the overall encryption overhead. On the other hand, in the non adaptive configuration, the OPE Layer is associated only to the columns in which the database administrator explicitly requires the support for order comparison operations. However, we observe that most of the TPC-C queries encryption times have not been affected by the introduction of adaptive encryption methods (Fig. 6).

In the following set of experiments, we investigate the impact of network latency on the SQL operations response time,

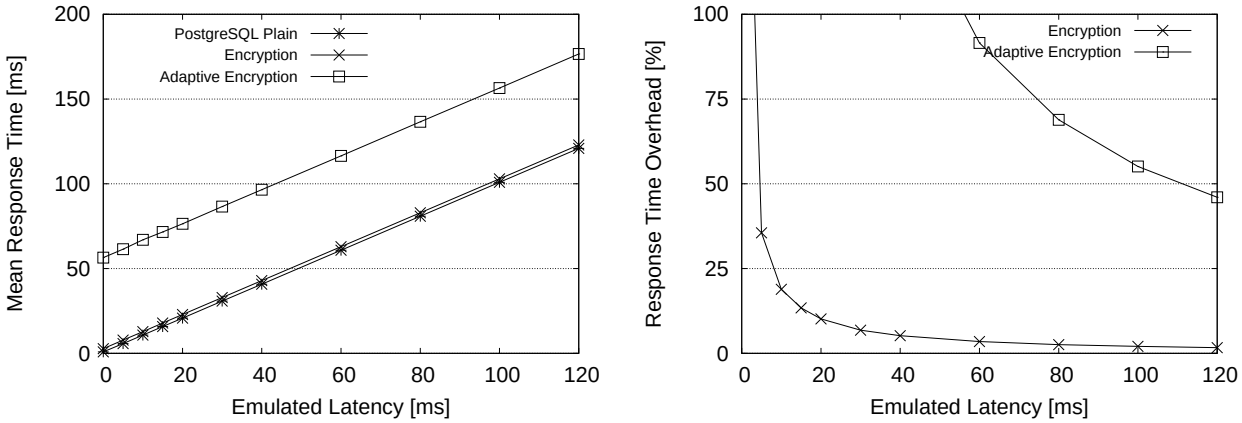
 ISBN: 978-1-61208-298-1



(a) SELECT - 10 clients

(b) SELECT overhead - 10 clients

Figure 7. Response time and overhead of the chosen SELECT operation (10 clients)



(a) INSERT - 10 clients

(b) INSERT overhead - 10 clients

Figure 8. Response time and overhead of the INSERT operation (10 clients)

with regard to the overhead caused by adaptive encryption. For this reason, we evaluate the response time of the most popular SELECT, DELETE, INSERT and UPDATE operations chosen from the TPC-C workload.

In Fig. 7a and Fig. 7b, we report the mean response times and the overhead of the chosen SELECT operation for 10 active clients as a function of increasing network latencies. The overhead of encryption and adaptive encryption is measured with respect to the plain database response time. These two figures confirm that the response time is quite similar for any type of database, and that the overhead of adaptive encryption tends to be masked for latencies higher than 60ms. This is an important conclusion because the results of the chosen SELECT query are also representative of the performance related to the chosen DELETE and UPDATE operations. Completely different results are obtained for the chosen INSERT operation in Fig. 8. In such a case, the adaptively encrypted version of the cloud database has response time and overhead much higher than those of the encrypted and plain databases. While for the encrypted database we can have similar conclusions to those achieved for the other SQL operations, the cost of the chosen INSERT combined with adaptive encryption is very high, and it remains high even for latencies superior to 120ms.

In order to understand the motivation of the overhead of this adaptive INSERT, in Table I, we report the number of encryptions (kE) required to encrypt the parameters of the operations, and the number of decryptions (kD) required to decrypt the results (if any), for each of the two configurations of the cloud database: encrypted and adaptively encrypted. We use the symbol E* to denote the number of OPE encryptions in Table I, because the encryption time required by the OPE algorithm is two or three orders of magnitude higher than that related to Rand and Det algorithms [8]. From this table, we can see that the high overhead of the INSERT in the adaptively encrypted configuration is caused by the higher number of encryptions required by the adaptive architecture to encrypt all parameters of the different Onions, and by the very high computational cost characterizing the OPE encryption.

 ISBN: 978-1-61208-298-1

TABLE I. NUMBERS OF ENCRYPTIONS (kE) AND DECRYPTIONS (kD) INVOLVED IN THE CHOSEN SELECT AND INSERT OPERATIONS FOR THE TWO ENCRYPTED DATABASE CONFIGURATIONS

Query	Layers	Non adaptive configuration	Adaptive configuration
SELECT	DET	2E	2E + 3D
	OPE	none	none
	RAND	3D	3D
INSERT	DET	2E	6E
	OPE	none	3E*
	RAND	7E	7E

C. Main conclusions of the performance study

All experimental results show that network latencies higher than 60 ms, which are typical of most cloud database environments, make the adaptive encryption overhead almost negligible when considering the overall set of operations of the TPC-C standard benchmark. However, in the adaptively encrypted database configuration, for some SQL operations involving the OPE encryption or for the encryption of a high number of parameters through several Encryption Layers (e.g., INSERT), the impact on the response time is visible even for network latencies higher than 120 ms. We can conclude that the proposed approach must be improved for contexts characterized by a large number of these operations. We are working on parallelized encryption schemes through multithreading over different cores.

If we refer to a scenario for increasing numbers of clients, we can observe that the (adaptive) encryption overheads (e.g., Fig. 7b and Fig. 8b) remain constant, while the cloud database throughput (e.g., Fig. 5a and Fig. 5b) increases proportionally as a function of the number of active clients.

It is worth to observe that the adaptively encrypted configuration represents a *worst case scenario*, in which each database column is encrypted into all the Onions supported by its data type (each Onion is assumed at its highest Encryption Layer). This is a scenario in which no user manually configure any encryption detail, and our architecture automatically chooses for each column all the supported Onions in order to guarantee adaptivity. This completely adaptive configuration requires a high number of encryptions and decryptions per query that may affect system performance especially for some SQL operations. On the other hand, the non adaptively encrypted configuration represents a *best case scenario*, in which the user manually defines which encryption scheme to use on each database column (see Section II). This configuration guarantees best performance, but no adaptivity. As a consequence, we can claim that realistic workloads are characterized by performance results falling between the two extreme scenarios presented in this paper.

IV. Conclusion and Future Work

We proposed an architecture that supports adaptive data confidentiality in cloud database environments without requiring any intermediate trusted proxy. Adaptive encryption mechanisms have two main benefits: they guarantee at runtime the maximum level of data confidentiality for any SQL workload, and they simplify database configuration at design time. However, they are affected by high computational costs with respect to non adaptive encryption schemes.

This paper demonstrated that applying adaptive encryption methods to cloud database services is a suitable solution, because network latency masks the overhead caused by adaptive encryption for most SQL operations. If we consider the overall set of queries belonging to the TPC-C standard benchmark, the overhead becomes negligible for network latencies that are typical of most intra-continental distances, and lower than those of inter-continental distances that often characterize cloud services.

Our results also show that the overhead of some SQL operations requiring more encryption steps and more parameters are not masked by Internet latencies. If the workload is characterized by many similar operations, the present alternative is to accept this cost when data confidentiality is more important than performance. As a future solution, we are also studying encryption parallelization solutions that can leverage multithreading over different processor cores.

References

[1] H. Hacigümüş, B. Iyer, and S. Mehrotra, “Providing database as a service,” in Proc. of the 18th IEEE International Conference on Data Engineering, February 2002, pp. 29–38.

[2] T. Mather, S. Kumaraswamy, and S. Latif, “Cloud security and privacy: an enterprise perspective on risks and compliance”. O’Reilly Media, Incorporated, 2009.

[3] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in Proc. of the 41st annual ACM symposium on Theory of computing, May 2009, pp. 169–178.

[4] A. Boldyreva, N. Chenette, and A. O’Neill, “Order-preserving encryption revisited: Improved security analysis and alternative solutions,” in Proc. of the Advances in Cryptology – CRYPTO 2011. Springer, August 2011, pp. 578–595.

[5] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in Proc. of the Advances in Cryptology – EUROCRYPT99. Springer, May 1999, pp. 223–238.

[6] D. Song, D. Wagner, and A. Perrig, “Practical techniques for searches on encrypted data,” in Proc. of the IEEE Symposium on Security and Privacy, May 2000, pp. 44–55.

[7] L. Ferretti, M. Colajanni, and M. Marchetti, “Distributed, concurrent, and independent access to encrypted cloud databases,” IEEE Transactions on Parallel and Distributed Systems, vol. 99, no. PrePrints, 2013.

[8] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, “CryptDB: protecting confidentiality with encrypted query processing,” in Proc. of the 23rd ACM Symposium on Operating Systems Principles, October 2011, pp. 85–100.

[9] L. Ferretti, M. Colajanni, and M. Marchetti, “Supporting security and consistency for cloud database,” in Proc. of the 4th International Symposium on Cyberspace Safety and Security. Springer, December 2012, pp. 179–193.

[10] O. Goldreich, Foundations of Cryptography: Volume 2, Basic Applications. Cambridge university press, 2004.

[11] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, “An integrated experimental environment for distributed systems and networks,” in Proc. of the 5th USENIX Conference on Operating Systems Design and Implementation, December 2002, pp. 255–270.

[12] J. Daemen and V. Rijmen, The design of Rijndael: AES – the advanced encryption standard. Springer, 2002.

[13] A. Fekete, D. Liarokapis, E. O’Neil, P. O’Neil, and D. Shasha, “Making snapshot isolation serializable,” ACM Transactions on Database Systems, vol. 30, no. 2, June 2005, pp. 492–528.

 ISBN: 978-1-61208-298-1

Active Shield with Electrically Configurable Interconnections

Umut Guvenc
National Research Institute of Electronics and Cryptology
BILGEM TUBITAK
Kocaeli, Turkey
e-mail: umut.guvenc@tubitak.gov.tr

***Abstract*—An active shielding method is hereby introduced, providing security to security critical integrated circuits against some physical attacks such as probing, manipulation and modification. Active shields satisfy a physical clearance by preventing the reachability of the circuitry while they, themselves, are subject to modification in most cases. The proposed active shield, which is also subject to a patent application, provides the ability to detect any physical modification made on the shield itself by utilizing electrically configurable interconnections between shielding metal lines. By changing the selected interconnection configuration of the shielding lines, the proposed active shield provides a self detection ability as a countermeasure against the vulnerability to physical modification made on the active shield itself.**

Keywords-active shield; secure IC; cryptography; physical attack countermeasure

I. INTRODUCTION

In security critical integrated circuits, some security countermeasures are implemented to provide safety of the critical information against some analysis and attack techniques. Some of these attack techniques aimed to obtain the information in an unauthorized way are known as physical attacks since they require actual physical access to the inner layers of the integrated circuit. Upon this attack concept, one should try to obtain the secret information via probing, manipulation or modification. These attack techniques include probing the critical information by making connections to the metal lines of the integrated circuit, faulting the integrated circuit by forcing from these outside connections and changing the connections of the internal metal lines permanently by using Focused Ion Beam (FIB) [1]. Active shield [2] is a countermeasure against these kinds of attacks, to be applied to the security critical integrated circuits which are physically accessible.

In active shield method, the whole surface of the integrated circuit is covered by metal lines on the top metal layer. These metal lines are supplied with a predefined or random test data from a transmitter and observed with a number of receivers located at certain points of the integrated circuit. The receivers are also supplied with the same test data internally thus they can compare the data on the shielding metals and the actual test data. According to the result of the comparison, these receiver circuitries verify the integrity of the top layer metal lines. Since any physical attack will disturb the integrity of these shielding lines by making them open or short circuit, the receiver circuitries do not receive the correct test data pattern from the shield, thus detect the physical attack.

It is believed that [3], [4], [5], [6] and [7] provide sufficient information on the background of the active shield method. In [3], a way of implementing the active shield method without requiring an additional metal layer is introduced, and in [4] [5], improvements are aimed to reduce the power consumption due to active shield. Derouet [6] introduced some improvements mainly on the detection circuitry part of the active shield method, not on the protection of the top metal layer shield itself.

Although being used in integrated circuits to detect any physical attack, active shield itself has still vulnerability against physical modification. Since the top layer metal lines of the active shield have fixed interconnections, it is possible to make shortcut connections between the lines and remove the parts covering the whole integrated circuit or a part of it, to perform the actual attack without being detected by the active shield. Some improvements can be made to decrease the vulnerability of the active shield to physical modification, like randomization of the connections of the top layer metal lines and increasing the number of receiver circuitries, however it is not possible to prevent the vulnerability completely.

In [7], a novel countermeasure against physical modification on the active shield is introduced. A capacitive measurement between the top layer metal lines of the active shield is performed along with the verification of the test data, to check whether the top layer metal lines are integral in their actual shapes. However, since the mentioned capacitive measurement between the top layer metal lines cannot be performed precisely, it is still possible for an attacker to perform partial physical modifications on the active shield while still satisfying sufficient capacitive coupling between the top layer metal lines.

In this work, an active shield aimed to prevent the vulnerability caused by the fixed interconnections of the shielding metal lines, by introducing a method using electrically configurable interconnections is proposed. Using electrically configurable interconnections provides the opportunity to select from more than one interconnection scheme during the operation of the integrated circuit. This dynamic configurability introduces a precise self-integrity checking mechanism to the active shield method. Thus, it is

 ISBN: 978-1-61208-298-1

prevented to bypass and remove the metal lines of the active shield by making fixed shortcut connections between them.

II. IMPLEMENTATION

A regular active shield implementation is shown in Fig. 1. In this implementation, the whole surface of the integrated circuit is covered by parallel shielding metals which are connected randomly yet fixed interconnections to each other to form the bit lines carrying the test data. The transmitter, receivers and the internal data paths between them are supposed to be hidden in the other circuitry thus cannot be recognized. The receiver circuitries verify the integrity of the shield by comparing the data received from the shielding lines with the actual data received internally. However, one should attempt to uncover the security critical circuit by making some external shortcut connections between the lines of the active shield as illustrated in Fig. 2. In such a case, the active shield would still claim that the shield is integral. By increasing the number of bit lines and the number of receivers, forming shortcut connections can become harder; however, the vulnerability is not prevented completely.

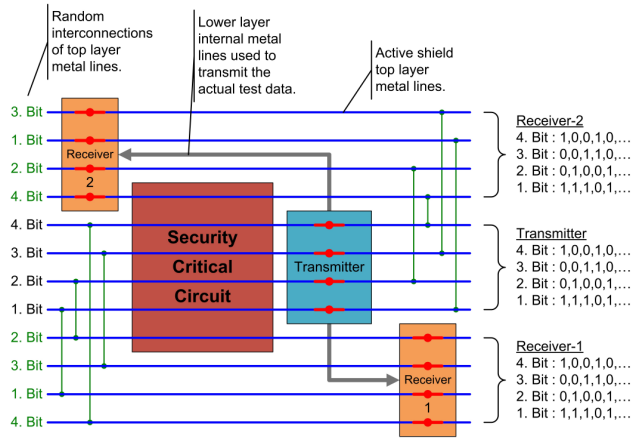


Figure 1. A regular active shield implementation with fixed interconnections.

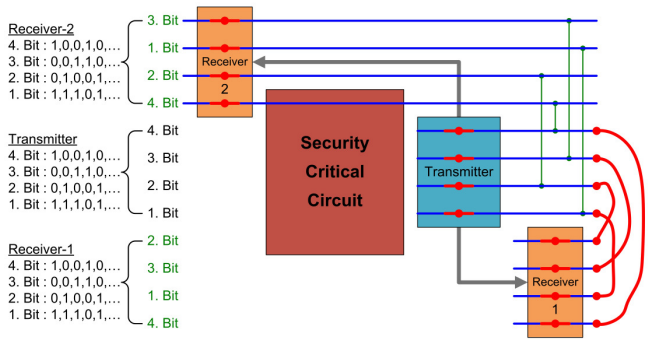


Figure 2. Illustration of bypassing the regular active shield.

In Fig. 3, an implementation of the proposed active shield is shown. The top layer metal lines are arranged in the same manner to cover the whole surface of the integrated circuit. Interconnections between the parallel metal lines are realized by using electrically switching circuits selecting from different random interconnection configurations. In the embodiment shown in Fig. 3, two different interconnection configurations are realized with switching circuits, which are in fact sufficient for the purpose of the active shield. The transmitter transmits a test data along with a select signal used to determine which interconnection configuration is selected. The receivers receive the test data from the bit lines and reorder the bits of the data received according to the select signal produced by the transmitter. The receivers also receive the same test data from the transmitter through internal data buses. The receivers verify the integrity of top metal lines of the active shield by comparing the test data received from the bit lines with the actual test data received from the internal data buses. Generally, the internal data buses carrying the actual test data the select signal, and the transmitter and the receivers themselves are arranged as a part of the integrated circuit such as distributed within the whole layout and not easily recognizable for the sake of security. Thanks to the electrically controllable switching circuits used to construct different interconnection configurations, the proposed active shield verifies the test data received from the bit lines with the actual test data for detection of the physical attacks focused on the integrated circuit, while providing the ability to detect any fixed modification made to bypass the shielding pattern and remove at least a part of it. In order to satisfy the latter purpose, the transmitter changes the selected interconnection configuration during the operation of the integrated circuit by changing the select signal regularly or randomly. Thus, any fixed physical modification on the upper layer conductive lines leads to an error in the verification of the test data.

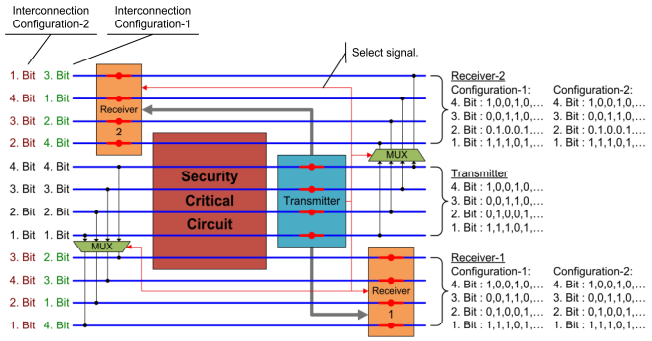


Figure 3. Proposed active shield implementation with electrically configurable interconnections.

Fig. 4 illustrates a bypassing attack on the proposed active shield. In the illustration, shielding metal lines are partly removed from the top of the security critical circuit by making fixed shortcut connections. These shortcut connections between the transmitter and one of the receivers are arranged to preserve the integrity of the bit lines according to the first interconnection configuration. Although the fixed shortcut connections satisfy the correct transmission of the test data when the first interconnection

 ISBN: 978-1-61208-298-1

configuration is selected, they do not satisfy the correct transmission of the test data when the second interconnection configuration is selected. Since the transmitter changes the select signal during the operation of the integrated circuit, the receivers would test the integrity of the shielding metal lines for all of the interconnection configurations and detect the attack. Thus, the vulnerability of the active shield to physical modification is prevented.

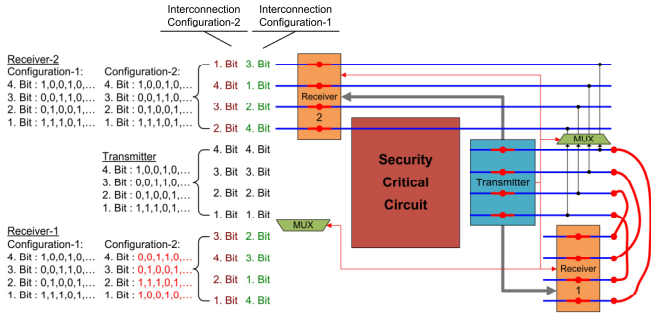


Figure 4. Illustration of bypass connections made on the proposed active shield and the detection of the modification.

Fig. 5 shows an exemplary embodiment of the electrically controllable switching circuits. Four two-input multiplexers are used to construct a part of the two different interconnection configurations of four bit lines as an example. The select signal determines in which order the inputs of the multiplexers are connected to the outputs.

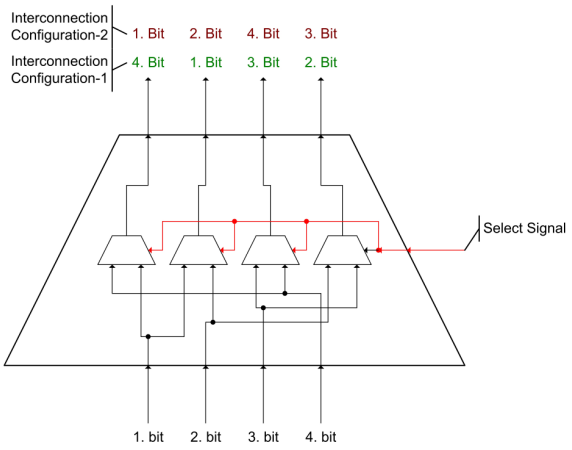


Figure 5. Exemplary embodiment of the electrically controllable switching circuits.

As seen in Fig. 5, electrically controllable switching circuits can be realized easily within a digital design, since they consist of basic multiplexer standard cells. The realization and the placement of multiplicity of these circuitries can be done by even automatic place and route approach or some scripting in the layout generation step.

III. CONCLUSION

A novel active shielding method is proposed, which has the ability to detect any bypassing attempt made by fixed physical modifications. Electrically controllable switching circuits are used to construct dynamically selectable different interconnections schemes between shielding metal lines.

ACKNOWLEDGMENT

The proposed active shield method is subject to a patent application numbered as 2011/11432 by Turkish Patent Institute.

REFERENCES

[1] M. Witteman, “Advances in smartcard security”, Information Security Bulletin 7.2002: 11-22.

[2] S. Briais, et al. “Random Active Shield”, Fault Diagnosis and Tolerance in Cryptography (FDTC), 2012 Workshop on. IEEE, 2012.

[3] A. Beit-Grogger and J. Riegebauer, “Integrated circuit having an active shield”, US Patent 92848 A1, published in May 5, 2005.

[4] G. Cutrignelli and R. Malzahn, “Circuit Arrangement, data processing device comprising such circuit arrangement as well as method for identifying an attack on such circuit arrangement”, US Patent 24890 A1, published in January 22, 2009.

[5] J. Ziomek, “Method to reduce power in active shield circuits that use complementary traces”, US Patent 150574 A1, published in January 26, 2008.

[6] O. Derouet, “Integrated circuits including reverse engineering detection using differences in signals”, US Patent 244749 A1, published in October 2, 2008.

[7] P. Laackmann and H. Taddiken, “Apparatus for protecting an integrated circuit formed in a substrate and method for protecting the circuit against reverse engineering”, US Patent 132777 A1, published in July 17, 2003.

 ISBN: 978-1-61208-298-1

BSPL: A Language to Specify and Compose Fine-grained Information Flow Policies

Stéphane Geller, Valérie Viet Triem Tong, Ludovic Mé
Team Cidre
SUPELEC
Rennes, France
Email: stephane.geller@supelec.fr, valerie.viettriemtong@supelec.fr, ludovic.me@supelec.fr

***Abstract*—We tackle the problem of operating systems security. The seriousness of the vulnerabilities in today's software underlines the importance of using a monitor at the operating system level to check the legality of operations executed by (un)trusted software. Information flow control is one way to track the propagation of information in order to raise an alert when a suspicious flow, consequence of an attack, occurs. We propose here BSPL, a language to specify a fine-grained information flow policy. We present how BSPL enables to precisely specify the expected behavior of applications relatively to sensitive pieces of information. We also propose a way to compose such information flow policies.**

Keywords*—*Information flow policies, specification language, composition of information flow policies

I. INTRODUCTION

Information flows characterize all the operations on the system resulting in data movements from one location to another, such as when a process reads or writes to a file.

Over the past decade, many worksd [1], [2], [3], [4], [5] have focused on providing strong information security guarantees by tracking and/or controlling information flows.

Basically, information flow tracking identifies sensitive data with a dedicated mark called taint, label or tag. The marks are propagated along the flow to taint objects in the system and a monitor uses the marks to check the legality of the information flows, with respect to a previously defined "information flow policy".

We believe that there is a need for a language to precisely define such policies, as proposed by Efstathopoulos and Kohler, who were the first to present such a language [6].

In this paper, we focus on the `Blare` monitor [7]. `Blare` monitors information flows at the operating system level. It uses two marks (called tags) associated with any container of information in the operating system. This monitor has shown to be helpful to detect attacks, especially those that generate data leakage [8]. Though, as with other existing monitors, `Blare` suffers from a major drawback: the information flow policy that manages the monitor is either a toy policy [2], set by hand [9], or automatically computed from an access control policy [8].

In this article, we introduce the `Blare` Security Policy Language (BSPL). This language has been created to be a convenient way to specify an information flow security policy, in particular in contexts where several types of data coexist in the same system, like on a smartphone. BSPL allows a high level expression of the policy while keeping it simple and clear. A BSPL policy can be used to configure an information flow monitor (in our case `Blare`) to detect illegal read or write accesses to sensitive information by any malicious software.

On smartphones, the global information flow policy associated to the system can be viewed as a composition of multiple smaller policies provided by the application developers, the administrators and the users. This motivates our work around the composition of policies that is also presented in this paper.

The two major contributions of this paper are, thus: (1) the BSPL language and (2) the composition definition for policies written in this language. We also propose a consistency property for such information flow policies and prove that a policy constructed using BSPL is consistent and it is easy to prove that the composition of consistent policies is consistent.

The remainder of this paper is organized as follows: Section II presents the related work. Section III introduces our underlying model of information flow. In Section IV, we present the BSPL syntax and semantics, and Section V shows how to compose policies. Lastly, before concluding the paper, in Section VI, we illustrate the use of BSPL to express information flow policies.

II. RELATED WORK

As far as we know, the work most related to our approach has been proposed by Efstathopoulos and Kohler [6]. They proposes a policy description language where application developers can express their security requirements in terms of communication relationships. They claim that concentrating the policy specification in one place makes policies easier to write and reason about. They also provide a parser to translate these communication relationships considered as high-level policy to the equivalent Asbestos labels [10].

Unfortunately, they note that their language does not capture the full expressiveness of the Asbestos labels. More precisely, a process can be impacted by different policies but the language does not permit the construction of a global policy, which includes all these fragments of policy. In other words, their language does not support the composition of different policies concerning the same process. Our composition

 ISBN: 978-1-61208-298-1

mechanism could solve this type of problem. They also have trouble inferring the high-level policy from the labels of the system.

We want to pursue the efforts initiated in [6] to concentrate the policy specification in one place. However, contrarily to what has been proposed by Efstathopoulos, our high-level language allows such a centralized definition while being very close to the actual labels managed by the `Blare` monitor. Hence, it would be easy to rebuild the system global policy in BSPL given the different labels found in that system. In addition, we also propose a consistency property, which permits the elimination of ill-formed information flow policies.

In the following section, we detail the model of labels used by the `Blare` monitor and thus describe the requirements for our specification language, BSPL.

III. THE UNDERLYING MODEL OF INFORMATION FLOW

`Blare` is a model of information flow created to accomplish mainly two objectives: (1) the tracking of the content of the containers of the system and (2) the easy verification that this content respects our information flow policy. To accomplish these goals, we distinguish the containers from their content, potentially composed of multiple pieces of information depending on the time of the observation.

We separate containers of information into three categories: $\mathcal{PC}$ denotes the set of persistent containers (typically files), and $\mathcal{E}xecute(\mathcal{X})$ (or shortly $\mathcal{E}$) denotes the set of processes executing a code in $\mathcal{X}$. We also consider users as containers of information. We use $\mathcal{U}$ to denote the set of users. In the following we will thus use the notation $\mathcal{C} = \mathcal{PC} \cup \mathcal{U} \cup \mathcal{E}$ to deal with *any container of information*.

To differentiate between a piece of information which has been executed (a running code) and a piece of information which has not, we define two disjoint sets of meta-information $\mathcal{I}$ and $\mathcal{X}$ to track these pieces of information. The elements of the first set correspond to non-executed pieces of information, while those of the second set correspond to pieces of information executed by processes. Once a piece of information identified by a meta-information $\mathcal{I}$ is executed, the meta-information used to track it will be changed thanks to the function $running : \mathcal{I} \rightarrow \mathcal{X}$. For instance, if `app` is an application considered as sensitive, we attach $i \in \mathcal{I}$ to its code viewed as a data stored in a file; once `app` is launched by a process p we consider that p is running the code denoted by $x \in \mathcal{X}$ such that x is equal to $runnning(i)$. This distinction permits to distinguish the policy associated to a code viewed as data, and to the same code when it runs. The identifiers in $\mathcal{I}$ and $\mathcal{X}$ are either defined at the initialization of the model or during the execution of the system if new content is added.

Any container existing in the operating system is characterized with two tags: an *information tag* and a *policy tag*. The *information tag* is a collection of elements of $\mathcal{I} \cup \mathcal{X}$ that denotes the origins of the current content of the container (it is the tag used to accomplish the objective (1)). The policy tag is a set of elements of $\mathscr{P}(\mathcal{I} \cup \mathcal{X})$ that denotes the information flow policy attached to the container (it is the tag used to accomplish the objective (2)).

The information flow policy defines for each of these containers (let us remind that users are viewed as containers) which data mixture they are allowed to contain. The information flow policy uses the function $\mathbb{P}_{\mathcal{C}} : \mathcal{C} \rightarrow \mathscr{P}(\mathscr{P}(\mathcal{I} \cup \mathcal{X}))$ as building block. This function defines the data mixture allowed to flow into any container of information. For instance, $\mathbb{P}_{\mathcal{C}}(\mathtt{c}) = \{\{i_1, i_2, x_1\}, \{i_1, i_3, x_1\}\}$ means that a process executing the tainted program x_1 is allowed to write into the container `c`; it also means that `c` can either contain data computed from the tainted data i_1 and i_2 or data computed from i_1 and i_3.

When a process is launched during the execution, the value of its policy tag depends on the process owner and the policy tag attached to the code the process is currently running. More precisely, let us consider a process p_{new} resulting from the execution of the binary file `bf` by a process p. Let us also consider that p has an information tag equal to $\mathtt{I}_p \subseteq \mathcal{I} \cup \mathcal{X}$ and p executes a binary file `bf`, which has an information tag equal to $\mathtt{I}_{bf} \subseteq \mathcal{I} \cup \mathcal{X}$. The information tag $\mathtt{I}_p$ tells us what information is transmitted (in the worst case) by p to its child p_{new}. The information tag $\mathtt{I}_{bf}$ tells us which code will be executed in p_{new}. The information tag of p_{new} is thus equal to the union of the information tag of its father process and the function $running$ applied to the information tag of the executed binary file `bf`: $\mathtt{I}_p \cup running(\mathtt{I}_{bf})$. Its policy tag $\mathtt{P}_{p_{new}}$ depends on the policy tag $\mathtt{P}_u \subseteq \mathscr{P}(\mathcal{I} \cup \mathcal{X})$ of the process owner $u \in \mathcal{U}$ and on the value of $\mathbb{P}_{\mathcal{C}}(\mathcal{E}xecute(running(i)))$ for any i in $\mathtt{I}_{bf}$. $\mathtt{P}_u$ is defined by $\mathbb{P}_{\mathcal{C}}(u)$. Lastly, the policy tag $\mathtt{P}_{p_{new}}$ is defined by $\sqcap_{i \in \mathtt{I}_{bf} \setminus \mathcal{X}} \mathbb{P}_{\mathcal{C}}(\mathcal{E}xecute(running(i))) \sqcap \mathtt{P}_u$ ($A \sqcap B = \{a \cap b | a \in A, b \in B\}$).

The policy tag $\mathtt{P}_c$ of a container of information (whatever its type is) represents the set of combinations of sensitive pieces of information which can legally be found in the container. The fact that the content of a container c is legal in accordance with an information flow policy can be verified by checking if its information tag $\mathtt{I}_c$ is included in at least one element of its policy tag $\mathtt{P}_c$.

The function $\mathbb{P}_{\mathcal{C}}$ permits defining which data mixture is allowed to flow into any type of containers of information. This function implicitly defines a second function that associates to sensitive pieces of information i their possible mixes and locations. We denote this function $\mathbb{P}_{\mathcal{I} \cup \mathcal{X}}$, its type is $\mathcal{I} \cup \mathcal{X} \rightarrow \mathscr{P}(\mathcal{C} \times \mathscr{P}(\mathcal{I} \cup \mathcal{X}))$. This function associates to any identifier of sensitive piece of information $ix \in \mathcal{I} \cup \mathcal{X}$ a set of pairs (c, s) where c can be either a persistent container, a user or a process and s is a set of identifiers of pieces of information. A pair $(c, s) \in \mathbb{P}_{(\mathcal{I} \cup \mathcal{X})}(ix)$ means that a data computed from the piece of information ix is allowed to flow into the container (persistent, user or process) c mixed with any data computed from any subset of pieces of information appearing in s.

We have proved in [11] that this model of information flows was complete: if a violation of the policy occurs, it is detected. We can also prove that the model is sound. Nevertheless, this proof of soundness is less relevant, as it supposes that any information flow can be observed. This "total observability" is in practice impossible to obtain. This means that the implementation of our model is not exempt of false positives. Nevertheless, our experiments have shown that this false positive rate is lower with a monitor implementing our model than with more classical monitors.

 ISBN: 978-1-61208-298-1

In previous works [2], [9], we have proposed constructing an information flow policy in specifying for each container the list of data mixtures that the container is allowed to contain. In other words, the definition of the policy was guided by containers and we directly defined the policy through the specification of $\mathbb{P}_{\mathcal{C}}$. In this case, the function $\mathbb{P}_{\mathcal{I}\cup\mathcal{X}}$ is defined by reversing $\mathbb{P}_{\mathcal{C}}$. We believe that it is interesting to permit guiding the definition of an information flow policy by specifying where data mixtures are allowed to flow and thus in partially defining the function $\mathbb{P}_{\mathcal{I}\cup\mathcal{X}}$. BSPL offers such a feature as it is detailed in the following section.

If we permit defining an information flow policy through the definition (even partial) of the function $\mathbb{P}_{\mathcal{I}\cup\mathcal{X}}$ we may face an inconsistency problem of the policy. Intuitively, an information flow policy is consistent if when a piece of information is allowed to flow into a container with a data mixture, it is equivalent to two things : the concerned container is also allowed to receive this data mixture and all the pieces of information of the mixture are allowed to flow in the container with the same mixture. Definition 1 makes this notion clear.

Definition 1: An information flow policy is consistent if the following formulas are equivalent :
$\forall c \in \mathcal{C}, \forall s \in \mathscr{P}(\mathcal{I} \cup \mathcal{X})$,

$$\exists ix \in s, (c, s \setminus \{ix\}) \in \mathbb{P}_{\mathcal{I}\cup\mathcal{X}}(ix) \quad (1)$$
$$\forall ix \in s, (c, s \setminus \{ix\}) \in \mathbb{P}_{\mathcal{I}\cup\mathcal{X}}(ix) \quad (2)$$
$$s \in \mathbb{P}_{\mathcal{C}}(c) \quad (3)$$

When a policy is consistent, we can express the two parts of the policy in function of the other: $\mathbb{P}_{\mathcal{I}\cup\mathcal{X}}(ix) = \{(c, I) | c \in \mathcal{C} \wedge I \cup \{ix\} \in \mathbb{P}_{\mathcal{C}}(c)\}$ and $\mathbb{P}_{\mathcal{C}}(c) = \bigcup_{ix\in\mathcal{I}\cup\mathcal{X}} \{I \cup \{ix\} | (c, I) \in \mathbb{P}_{\mathcal{I}\cup\mathcal{X}}(ix)\}$. An information flow policy uniquely specified through one of the two is necessarily consistent since the last function is in this case entirely defined by reversing.

In the following section, we propose a dedicated language, BSPL, associated to this model.

IV. THE BSPL LANGUAGE

A. BSPL Grammar

A `Blare` policy is basically composed of two elements: the definition of sensitive data and their corresponding policy, and the definition of containers and their corresponding policy.

The first element on the policy is called *Data policy* and defines the policy attached to sensitive data. Pieces of information listed in this element belong to the user/the application/the system that defines the current policy. These pieces of information are thus considered as sensitive. This element is composed of a sequence of element `data` (each one characterizing a single piece of information, for example the initial content of a file), which is composed of:

- an element `data_identification` that permit the exact characterization of the sensitive piece of information (it gives its alias, its original container, its owner, its type (executable (X) or not (I)), the expected `reaction` of the monitor if it observes an illegal information flow: either it has to block the incriminated information flow or it has to raise an alert. Lastly an element `data_identification` has an attribute `case_of_unknown_containers` that is helpful for policy composition. This last attribute specifies the behaviour of the information identified above when the policy has to be composed with another policy that specifies information flows with this piece of information. It can be either `Ask` for *ask to the owner of the piece of information* (encouraged default value), `AlwaysAccept` (very permissive) or `NeverAccept` (very protective).

- a list of elements `can_flow` each of them having two attributes: `into` that denotes a container and `mixed_with` that denotes a list of sets of identifiers of sensitive data.

The attributes composing the element `can_flow` mean that the sensitive data identified by the previous element `identification` can be mixed with any set of pieces of information listed in `mixed_with` in the container in the attribute `into`. We can use the particular identifier `all_data`, which will mean all the possible sensitive data. The precise semantics associated with the group Data_policy is given in Section IV-B.

The second element defines the information flow policy attached to containers of information and is called *Containers_policy*. This element is composed of a list of element `container`. An element `container` is defined by:

- an element `container_identification` with five attributes that permit the exact characterization of the container (it gives an alias for the container, its original location, its owner and its type (file, process executing a code identified previously in `data_policy`) or user) , the expected `reaction` of the monitor if it observes an illegal information flow and lastly a behaviour. Dually to the case of unknown containers, an element `container_identification` has an attribute `case_of_unknown_data`, which specifies the behaviour of the container identified above when the policy has to be composed with another policy that specifies information flows with this container. It can be either `Ask` for *ask to the owner of the container* (encouraged default value), `AlwaysAccept` (very permissive) or `NeverAccept` (very protective).

- an element `can_receive` composed of one attribute: `mixture_of_data`, which indicates the mixture of data allowed to flow into the concerned container: `mixture_of_data` has to be a list of list of pieces of information separated with commas or it can be simply `all_data` (all the possible sensitive data).

In the next subsection, we describe the sense of the syntax given in this section, i.e., the semantics of BSPL, using the notations introduced in Section III.

 ISBN: 978-1-61208-298-1

B. BSPL semantics

We define two tables $\mathbb{T}_{\mathcal{I}\cup\mathcal{X}}$ and $\mathbb{T}_{\mathcal{C}}$. The first table $\mathbb{T}_{\mathcal{I}\cup\mathcal{X}}$ is used to store information permitting to characterize sensitive data (their owners, their initial location, their type (executable or not) and the reaction attached to an illegal flow with the data.. The second table $\mathbb{T}_{\mathcal{C}}$ is used to store the origin, the owner and the reaction attached to a container defined in the policy.

The monitor refers to these tables when it detects an illegal information flow, since the expected reaction is stored in theses tables. These tables will be used during the installation process and will be helpful for the enforcement of the policy. These tables can be deduced from the definition of $\mathbb{P}_{\mathcal{I}\cup\mathcal{X}}$ and $\mathbb{P}_{\mathcal{C}}$.

The four rules given hereafter define the semantics of the BSPL language.

V. Composition of Policies

In this section, we present how to compose two information flow policies designed by different owners of data and containers. We give a formula that permits to compute an information flow policy $\mathbb{P}^f$ resulting from the composition of two policies $\mathbb{P}^1$ and $\mathbb{P}^2$ and we denote it by:

$$\mathbb{P}^f = (\mathbb{P}^f_{\mathcal{C}_f}, \mathbb{P}^f_{\mathcal{I}_f\cup\mathcal{X}_f}, \mathbb{T}^f_{\mathcal{C}_f}, \mathbb{T}^f_{\mathcal{I}_f\cup\mathcal{X}_f}) = \mathbb{P}^1 \oplus \mathbb{P}^2 = (\mathbb{P}^1_{\mathcal{C}_1}, \mathbb{P}^1_{\mathcal{I}_1\cup\mathcal{X}_1}, \mathbb{T}^1_{\mathcal{C}_1}, \mathbb{T}^1_{\mathcal{I}_1\cup\mathcal{X}_1})) \oplus (\mathbb{P}^2_{\mathcal{C}_2}, \mathbb{P}^2_{\mathcal{I}_2\cup\mathcal{X}_2}, \mathbb{T}^2_{\mathcal{C}_2}, \mathbb{T}^2_{\mathcal{I}_2\cup\mathcal{X}_2})$$

The tables $\mathbb{T}_{\mathcal{C}}$ and $\mathbb{T}_{\mathcal{I}\cup\mathcal{X}}$ are used for the behaviour in case of unknown containers/data and to extract the owners of the containers/data and the authors of the policies. We suppose here that data are uniquely identified (through their initial location for instance) and that an identifier only corresponds to a piece of information. We also suppose that users, persistent containers and processes are uniquely identified. The sensitive non-executed pieces of information manipulated by $\mathbb{P}_f$ is denoted by $\mathcal{I}_f$ and is equal to $\mathcal{I}_1 \cup \mathcal{I}_2$. Codes that will later be executed by processes and that are considered as sensitive for the policy $\mathbb{P}_f$ are $\mathcal{X}_1 \cup \mathcal{X}_2$. Consequently $(\mathcal{I}_f \cup \mathcal{X}_f)$ is equal to $((\mathcal{I}_1 \cup \mathcal{I}_2) \cup (\mathcal{X}_1 \cup \mathcal{X}_2))$.

In the same way, containers of information manipulated by $\mathbb{P}_f$ are $\mathcal{C}_f = \mathcal{PC}_f \cup U_f \cup \mathcal{E}xecute(\mathcal{X}_f)$ where users are $\mathcal{U}_f = \mathcal{U}_1 \cup \mathcal{U}_2$, persistent containers are $\mathcal{PC}_f = \mathcal{PC}_1 \cup \mathcal{PC}_2$ and lastly processes $\mathcal{E}xecute(\mathcal{X}_f)$ are $\mathcal{E}xecute(\mathcal{X}_1) \cup \mathcal{E}xecute(\mathcal{X}_2)$.

The final tables are $\mathbb{T}^f_{\mathcal{C}_f} = \mathbb{T}^1_{\mathcal{C}_1} \cup \mathbb{T}^2_{\mathcal{C}_2}$ and $\mathbb{T}^f_{\mathcal{I}_f\cup\mathcal{X}_f} = \mathbb{T}^1_{\mathcal{I}_1\cup\mathcal{X}_1} \cup \mathbb{T}^1_{\mathcal{I}_2\cup\mathcal{X}_2}$.

We define the function information owner $ownerl : \mathcal{I}_f \to \mathcal{U}$ ($\mathcal{U}$ is the set of users owning containers or data in one of the two policies, we can build it with the field owner of the tables). $ownerl(i) = \mathbb{T}^f_{\mathcal{I}_f\cup\mathcal{X}_f}(i)^3$ (the third field of the table, the owner). We also define the function container owner $ownerc : \mathcal{C}_f \to \mathcal{U}$. $ownerc(c) = \mathbb{T}^f_{\mathcal{C}_f}(c)^3$. The function $behaviour : \mathcal{I}_f \cup \mathcal{C}_f \to \{AlwaysAccept; NeverAccept; Ask\}$ is used to access the field behaviour of the tables. We define the function $authors$, which returns the set of the authors of

Rule 1: construction of the data policy ($\mathbb{P}_{\mathcal{I}\cup\mathcal{X}}$)

```
data_identification   = {
        alias = al;
        origin = or;
        owner = own;
        data_type = type;
        reaction = reac;
        case_of_unknown
        _containers = beha; },
[[can_flow   = {
        into = c;
        mixed_with_data = mix; }
 can_flow    = {...     }]]
```

if $(al, \_, \_, \_, \_) \notin \mathbb{T}_{\mathcal{I}\cup\mathcal{X}}$
then $\mathbb{T}_{\mathcal{I}\cup\mathcal{X}} = \mathbb{T}_{\mathcal{I}\cup\mathcal{X}} \cup (al, or, own, type, reac, beha)$;
for each `can_flow` element do
$\mathbb{P}_{\mathcal{I}\cup\mathcal{X}}(al) \leftarrow (c, mix)$
$\forall ix \in mix\ \mathbb{P}_{\mathcal{I}\cup\mathcal{X}}(ix) \leftarrow (c, (mix \cup \{al\}) \setminus \{ix\})$
$\mathbb{P}_{\mathcal{C}}(c) \leftarrow \mathbb{P}_{\mathcal{C}}(c) \cup (mix \cup \{al\})$ done

Rule 2.1: Construction of the Container Policy ($\mathbb{P}_{\mathcal{C}}$): User's Case

```
container_identification   = {
        alias = u;
        origin = "";
        owner = "";
        container_type = User;
        reaction = reac;
        case_of_unknown
        _data = beha; }
can_receive   = {
        mixture_of_data = mix; }
```

if $(u, "", "", User, \_) \notin \mathbb{T}_{\mathcal{C}}$
then $\mathbb{T}_{\mathcal{C}} = \mathbb{T}_{\mathcal{C}} \cup (u, "", "", User, reac, beha)$;
$\mathbb{P}_{\mathcal{C}}(u) \leftarrow mix$;
$\forall ix \in mix,\ \mathbb{P}_{\mathcal{I}\cup\mathcal{X}}(ix) \leftarrow (c, (mix \setminus \{ix\}))$

Rule 2.2: Construction of the Container Policy ($\mathbb{P}_{\mathcal{C}}$): File's Case

```
container_identification   = {
        alias = al;
        origin = or;
        owner = own;
        container_type = File;
        reaction = reac;
        case_of_unknown
        _data = beha; }
can_receive   = {
        mixture_of_data = mix; }
```

if $(al, \_, \_, \_, \_) \notin \mathbb{T}_{\mathcal{C}}$
then $\mathbb{T}_{\mathcal{C}} = \mathbb{T}_{\mathcal{C}} \cup (al, or, own, reac, beha)$;
$\mathbb{P}_{\mathcal{C}}(al) \leftarrow mix$
$\forall ix \in mix,\ \mathbb{P}_{\mathcal{I}\cup\mathcal{X}}(ix) \leftarrow (c, (mix \setminus \{ix\}))$

Rule 2.3: Construction of the Container Policy ($\mathbb{P}_{\mathcal{C}}$): Process Case

```
container_identification   = {
        alias = al;
        origin = or;
        owner = own;
        container_type = Process;
        reaction = reac;
        case_of_unknown
        _data = beha; }
can_receive   = {
        mixture_of_data = mix;  }
```

if $(al, \_, \_, \_, \_) \notin \mathbb{T}_{\mathcal{C}}$
then $\mathbb{T}_{\mathcal{C}} = \mathbb{T}_{\mathcal{C}} \cup (\mathcal{E}xecute(al), or, own, reac, beha)$;
$\mathbb{P}_{\mathcal{C}}(\mathcal{E}xecute(al)) \leftarrow mix$
$\forall ix \in mix,\ \mathbb{P}_{\mathcal{I}\cup\mathcal{X}}(ix) \leftarrow (c, (mix \setminus \{ix\}))$

Fig. 1: Semantics rules of BSPL

a policy (the set of owners of data/containers appearing in the table of this policy). We finally use $request : \mathcal{U} \times (\mathcal{I}_f \cup \mathcal{C}_f) \times \mathscr{P}(\mathcal{I}_f \cup \mathcal{X}_f) \rightarrow \{True; False\}$, boolean function as the answer from the user to the request of a policy.

The building function $\mathbb{P}^f_{\mathcal{C}_f}$ is formally defined by the definition 2. Intuitively this definition states that the sets I associated with c are either in the policy $\mathbb{P}^1$ (resp. $\mathbb{P}^2$) with every data and c owned by an author of $\mathbb{P}^1$ (resp. $\mathbb{P}^2$) or correspond to an intersection of the two policies on the containers for which both policies specify mixes of information. The rest of the sets in the definition are used to collect the cases in which the behaviour specified for an unknown container or an unknown data by an owner allows the result policy to contain an element.

Definition 2 explains the construction of $\mathbb{P}^f_{\mathcal{I}_f \cup \mathcal{X}_f}$. This definition is dual (thanks to the coherence property) with the definition of $\mathbb{P}^f_{\mathcal{C}_f}$.

With this definition, we are able to prove that the policy computed by composition of two consistent policies is still a consistent policy. The proof is not detailed here due to the limitation of space.

Definition 2 (composition of policies):
$\mathbb{P}^1$ and $\mathbb{P}^2$ two information flow policies. The policy $\mathbb{P}^f = (\mathbb{P}^f_{\mathcal{C}_f}, \mathbb{P}^f_{\mathcal{I}_f \cup \mathcal{X}_f})$ obtained by composition of $\mathbb{P}^1$ and $\mathbb{P}^2$ is denoted by $\mathbb{P}^1 \oplus \mathbb{P}^2$ and is defined by $\mathbb{P}^f_{\mathcal{C}_f}$ and $\mathbb{P}^f_{\mathcal{I}_f \cup \mathcal{X}_f}$ where : $(\mathcal{I}_f \cup \mathcal{X}_f) = ((\mathcal{I}_1 \cup \mathcal{I}_2) \cup (\mathcal{X}_1 \cup \mathcal{X}_2))$, and $\mathcal{C}_f = \mathcal{C}_1 \cup \mathcal{C}_2$

$$
\begin{aligned}
&\mathbb{P}^f_{\mathcal{C}_f}(c) = \\
&\quad \{I \in \mathbb{P}^1_{\mathcal{C}_1}(c) | \forall i \in I, \\
&\quad owner l(i) \in authors(\mathbb{P}^1) \wedge (ownerc(c) \in authors(\mathbb{P}^1))\} \\
&\cup \{I \in \mathbb{P}^2_{\mathcal{C}_2}(c) | \forall i \in I, \\
&\quad owner l(i) \in authors(\mathbb{P}^2) \wedge (ownerc(c) \in authors(\mathbb{P}^2))\} \\
&\cup \{(I_1 \cap I_2) | I_1 \in \mathbb{P}^1_{\mathcal{C}_1}(c) \\
&\quad \wedge I_2 \in \mathbb{P}^2_{\mathcal{C}_2}(c)\} \\
&\cup \{I \in \mathbb{P}^1_{\mathcal{C}_1}(c) | (ownerc(c) \in author(\mathbb{P}^2)) \\
&\quad \wedge (behaviour(c) = AlwaysAccept \\
&\quad \vee (behaviour(c) = Ask \\
&\quad \wedge request(ownerc(c), c, I))))\} \\
&\cup \{I \in \mathbb{P}^2_{\mathcal{C}_2}(c) | (ownerc(c) \in author(\mathbb{P}^1)) \\
&\quad \wedge (behaviour(c) = AlwaysAccept \\
&\quad \vee (behaviour(c) = Ask \\
&\quad \wedge request(ownerc(c), c, I \cup \{ix\})))\} \\
&\cup \{I \in \mathbb{P}^1_{\mathcal{C}_1}(c) | \exists ix \in I, owner l(ix) \in authors(\mathbb{P}^2) \\
&\quad \wedge (ownerc(c) \in author(\mathbb{P}^1)) \\
&\quad \wedge (behaviour(ix) = AlwaysAccept \\
&\quad \vee (behaviour(ix) = Ask \\
&\quad \wedge request(owner l(ix), ix, I \cup \{ix\})))\} \\
&\cup \{I \in \mathbb{P}^2_{\mathcal{C}_2}(c) | \exists ix \in I, owner l(ix) \in authors(\mathbb{P}^1) \\
&\quad \wedge (ownerc(c) \in author(\mathbb{P}^2)) \\
&\quad \wedge (behaviour(ix) = AlwaysAccept \\
&\quad \vee (behaviour(ix) = Ask \\
&\quad \wedge request(owner l(ix), ix, I \cup \{ix\})))\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathbb{P}^f_{\mathcal{I}_f \cup \mathcal{X}_f}(ix) = \\
&\quad \{(c, I) \in \mathbb{P}^1_{\mathcal{I}_1 \cup \mathcal{X}_1}(ix) | \forall i \in (I \cup \{ix\}), \\
&\quad owner l(i) \in authors(\mathbb{P}^1) \wedge (ownerc(c) \in authors(\mathbb{P}^1))\} \\
&\cup \{(c, I) \in \mathbb{P}^2_{\mathcal{I}_2 \cup \mathcal{X}_2}(ix) | \forall i \in (I \cup \{ix\}), \\
&\quad owner l(i) \in authors(\mathbb{P}^2) \wedge (ownerc(c) \in authors(\mathbb{P}^2))\} \\
&\cup \{(c, I_1 \cap I_2) | (c, I_1) \in \mathbb{P}^1_{\mathcal{I}_1 \cup \mathcal{X}_1}(ix) \\
&\quad \wedge (c, I_2) \in \mathbb{P}^2_{\mathcal{I}_2 \cup \mathcal{X}_2}(ix)\} \\
&\cup \{(c, I) \in \mathbb{P}^1_{\mathcal{I}_1 \cup \mathcal{X}_1}(ix) | (ownerc(c) \in author(\mathbb{P}^2)) \\
&\quad \wedge (behaviour(c) = AlwaysAccept \\
&\quad \vee (behaviour(c) = Ask \\
&\quad \wedge request(ownerc(c), c, I \cup \{ix\})))\} \\
&\cup \{(c, I) \in \mathbb{P}^2_{\mathcal{I}_2 \cup \mathcal{X}_2}(ix) | (ownerc(c) \in author(\mathbb{P}^1)) \\
&\quad \wedge (behaviour(c) = AlwaysAccept \\
&\quad \vee (behaviour(c) = Ask \\
&\quad \wedge request(ownerc(c), c, I \cup \{ix\})))\} \\
&\cup \{(c, I) \in \mathbb{P}^1_{\mathcal{I}_1 \cup \mathcal{X}_1}(ix) | owner l(ix) \in authors(\mathbb{P}^2) \\
&\quad \wedge (ownerc(c) \in author(\mathbb{P}^1)) \\
&\quad \wedge (behaviour(ix) = AlwaysAccept \\
&\quad \vee (behaviour(ix) = Ask \\
&\quad \wedge request(owner l(ix), ix, I \cup \{ix\})))\} \\
&\cup \{(c, I) \in \mathbb{P}^2_{\mathcal{I}_2 \cup \mathcal{X}_2}(ix) | owner l(ix) \in authors(\mathbb{P}^1) \\
&\quad \wedge (ownerc(c) \in author(\mathbb{P}^2)) \\
&\quad \wedge (behaviour(ix) = AlwaysAccept \\
&\quad \vee (behaviour(ix) = Ask \\
&\quad \wedge request(owner l(ix), ix, I \cup \{ix\})))\}
\end{aligned}
$$

VI. USING BSPL: EXAMPLES

A. Example of policy in BSPL

We propose in this section a part of what could be a flow policy of a game application for Android, e.g., AngryBirdsSpace. We have studied this application and found that the pieces of information linked to the application (as its code, its own data) can flow to 61 containers of information (more precisely 22 processes and 39 files). Our study has also shown that once launched the game receives piece of information from more than 30 differents origins.

If we were developers of this application, we would have to specify a BSPL policy that takes into account this behavior. We present in Fig. 2 a part of this policy. This part defines a sensitive piece of information, which is the content of the application package file `/data/app/com.rovio.angrybirds-space.ads-1.apk` aliased as *AngryBirdSpace*. This piece of information is owned by an owner defined by the Android operating system during the installation process. It is not an executed information (i.e., it is of type I). In case of detection of an illegal flow, the monitor raises an alert but does not forbid the flow. The default behavior is used in case of an unknown container. The rest of the `<Data-policy>` defines the containers this piece of information can flow into. More precisely it specifies that this application may create a file where it will store its cookies, will create a file named `highscores.lua.tmp` that will contain the scores, will use the vibrate facility and will ask the music player to play its own music. This application will create at least one container of information (a process) named `ybirdsspace.ads` that will receive data, in particular it will receive the piece of information identified just above *AngryBirdSpace*.

 ISBN: 978-1-61208-298-1

```
<BSPL-policy>
  <Data-policy>
    <data>
      <data_identification
      alias ="AngryBirdSpace"
      origin="/data/app/com.rovio.angrybirdsspace
                .ads-1.apk"
      owner="App_2"
      type="I"
      reaction="Alert"
      case_of_unknown_containers="Ask"
      />

      <can_flow
          into="/data/data/com.rovio.angrybirdsspace
                  .ads/databases/cookiedb-journal"
          mixed_with_data=""
        />
        <can_flow
          into="/data/data/com.rovio.angrybirdsspace
                  .ads/files/highscores.lua.tmp"
          mixed_with_data=""
        />
        <can_flow
          into=""
          mixed_with_data="/sys/devices/virtual/
                           timed_output/vibrator
                           /enable"
        />
        <can_flow
          into="android.musicfx"
          mixed_with_data=""
        />
      <can_flow>
          into="system_server"
          mixed_with_data=""
        />
        <can_flow
          into="system_server"
          mixed_with_data=""
        />
    </data>
  </Data-policy>

  <Containers-policy>
    <container>
      <container_identification
        alias=" ybirdsspace.ads"
        origin=" ybirdsspace.ads"
        owner="app_2"
        container_type="Process"
        reaction="Alert"
        case_of_unknown_data="Ask"
      />
      <can_receive
          mixture_of_data="AngryBirdSpace"
        />
      </can_receive>
    </container>
  </Containers-policy>

</BSPL-policy>
```

Fig. 2: A part of a BSPL policy for Angry Bird Space

```
<BSPL-policy>
  <Data-policy>
    <data>
      <data_identification
        alias ="file"
        origin="/data/local/tmp/file"
        owner="me"
        type="I"
        reaction="Alert"
        case_of_unknown_containers="Ask"
      />
      <can_flow>
          into="md5sum"
          mixed_with_data=""
        />
        <can_flow
          into="hash"
          mixed_with_data=""
        />
    </data>
  </Data-policy>
<Containers-policy>
    <container>
      <container_identification
        alias="hash"
        origin="/data/local/tmp/hash"
        owner="root"
        container_type="File"
        reaction="Alert"
        case_of_unknown_data="NeverAccept"
      />
    </container>
    <container>
      <container_identification
        alias="md5sum"
        origin="/system/bin/md5sum"
        owner="me"
        container_type="Process"
        reaction="Alert"
        case_of_unknown_data="AlwaysAccept"
      />
    </container>
  </Containers-policy>
</BSPL-policy>
```

Fig. 3: Hashing Example: Initial Policy

B. Example of Composition

In this subsection, we describe another example to illustrate a simple case of composition. We propose a scenario involving a hashing function. Notice that in this example the owner of all data and all containers is root, but it could be anything else and in particular the two processes could be owned by different users. The hashing function `md5sum` is installed in our system. We have also a file named FILE containing sensitive values ; we want to forbid the reading of these values. However, we trust the application `md5sum` and we permit to compute and publish the hash value of FILE using `md5sum` in HASH. Fig. 3 gives this simple policy.

With such a policy, using the command `md5sum /data/local/tmp/file > /data/local/tmp/hash` does not raise an alert while

 ISBN: 978-1-61208-298-1

```
<Containers-policy>
  <container>
    <container_identification
      alias="sha1sum"
      origin="/system/bin/sha1sum"
      owner="root"
      container_type="Process"
      reaction="Alert"
      case_of_unknown_data="NeverAccept"
    />
    <can_receive
        mixture_of_data="file"
      />
  </container>
</Containers-policy>
```

Fig. 4: Hashing Example: sha1sum Policy

the command `sha1sum /data/local/tmp/file > /data/local/tmp/hash` does. Indeed, the policy does not take `sha1sum` into account at this point. Notice that the behavior "`ask`" is associated to unknown containers.

Fig. 4 illustrates: `md5sum` is now considered unsafe and the owner of `sha1sum` wants to replace `md5sum` by `sha1sum`. This policy expresses that the owner of `sha1sum` requests the right for the processes running `sha1sum` to contain *file*.

We can now compose these two last policies. In our scenario, this composition requires the intervention of the owner of *file* since the behavior specified in the case of an unknown container is "`ask`". If this behavior was "`alwaysaccept`" or "`neveraccept`", the composition could be realized automatically. The owner of *file* accepts since he wants to replace `md5sum` by `sha1sum`. Fig. 5 shows the result of the composition.

In this composed policy, both `md5sum` and `sha1sum` have the right to contain *file* and `hash` can still contain *file* too. There are three containers and all of them can contain *file*. Now the command `sha1sum /data/local /tmp/file > /data/local/tmp/hash` no longer raises an alert. However, the command `md5sum /data/local/tmp/file > /data/local/tmp/hash` neither raises an alert. The final step is thus to remove the right of `md5sum` to contain *file*. This step can only be realized by a manual intervention from the owner of `md5sum`. The final resulting policy is almost the same as in Fig. 5, but `md5sum` can no longer contain *file*. After this step, the command `md5sum /data/local/tmp/file > /data/local/tmp/hash` raises an alert. The lines of the policy that disappear are all the lines concerning `md5sum`.

This scenario illustrates a simple use of BSPL, but it shows the expressiveness of the language. Each owner can express a policy for its data and containers and let the composition mechanism merge the policies to produce the policy to be finally enforced by Blare on the system. This example also shows that it is possible to restrain access to some pieces of information to specific processes executing a code considered as safe.

```
<?xml version="1.0" encoding="UTF-8"?>
<BSPL-policy>
  <Data-policy>
    <data>
      <data_identification
      alias ="file"
      origin="/data/local/tmp/file"
      owner="root"
      type="I"
      reaction="Alert"
      case_of_unknown_container="Ask"
      />
      <can_flow
        into="md5sum"
        mixed_with_data=""
      />
      <can_flow
        into="sha1sum"
        mixed_with_data=""
       />
       <can_flow
        into="hash"
        mixed_with_data=""
        />
    </data>
  </Data-policy>
  <Containers-policy>
    <container>
      <container_identification
      alias="hash"
      origin="/data/local/tmp/hash"
      owner="root"
      container_type="File"
      reaction="Alert"
      case_of_unknown_data="NeverAccept"
      />
    </container>
    <container>
      <container_identification
      alias="md5sum"
      origin="/system/bin/md5sum"
      owner="root"
      container_type="Process"
      reaction="Alert"
      case_of_unknown_data="AlwaysAccept"
      />
    </container>
    <container>
      <container_identification
      alias="sha1sum"
      origin="/system/bin/sha1sum"
      owner="root"
      container_type="Process"
      reaction="Alert"
      case_of_unknown_data="NeverAccept"
      />
    </container>
  </Containers-policy>
</BSPL-policy>
```

Fig. 5: Hashing Example: Composed Policy

VII. Conclusion and future work

In this paper, we have presented BSPL, a language designed to specify an information flow policy for developpers of

applications of users. This language is dedicated to the `Blare` monitor.

The information flow policies expressed using BSPL specify which and how sensitive pieces of information may be combined and in which containers of information these data mixtures can flow.

The language has been presented here and we have proposed precise semantics based on a solid model of information flow published earlier.

We have defined a consistency property for a policy specified in this model and have shown how to compose two consistent policies. Our definition of composition respects the consistency property: a composition of two consistent policies leads to a third consistent policy.

We already have a tool allowing to compute blare tags using a BSPL policy. Thus, we are able to monitor flows spawned by applications running over Android with respect to a BSPL policy.

For our future work, we plan to focus on adding the possibility for users to declassify sensitive data in certain contexts and to administrate their policy.

REFERENCES

[1] N. Zeldovich, S. Boyd-Wickizer, E. Kohler, and D. Mazières, "Making information flow explicit in histar," in *OSDI*, 2006, pp. 263–278.

[2] V. V. T. Tong, A. J. Clark, and L. Mé, "Specifying and enforcing a fine-grained information flow policy : model and experiments," vol. 1, no. 1, 2010, pp. 56–71, this paper was part of the 2nd International Workshop on Managing Insider Security Threats (MIST 2010); Morika, Iwate, Japan (14-15 June 2010).

[3] A. Sabelfeld and A. C. Myers, "Language-based information-flow security," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 1, pp. 5–19, 2003.

[4] P. Efstathopoulos, M. N. Krohn, S. Vandebogart, C. Frey, D. Ziegler, E. Kohler, D. Mazières, M. F. Kaashoek, and R. Morris, "Labels and event processes in the asbestos operating system," in *SOSP*, 2005, pp. 17–30.

[5] W. Enck, P. Gilbert, B. gon Chun, L. P. Cox, J. Jung, P. McDaniel, and A. Sheth, "Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones," in *OSDI*, 2010, pp. 393–407.

[6] P. Efstathopoulos and E. Kohler, "Manageable fine-grained information flow," in *EuroSys*, 2008, pp. 301–313.

[7] S. Geller, C. Hauser, F. Tronel, and V. V. T. Tong, "Information flow control for intrusion detection derived from mac policy," in *ICC*, 2011, pp. 1–6.

[8] G. Hiet, V. V. T. Tong, L. Mé, and B. Morin, "Policy-based intrusion detection in web applications by monitoring java information flows," pp. 53–60, 2008.

[9] "Designing information flow policies for android's operating system," in *ICC*, 2012, pp. 976–981.

[10] S. Vandebogart, P. Efstathopoulos, E. Kohler, M. N. Krohn, C. Frey, D. Ziegler, M. F. Kaashoek, R. Morris, and D. Mazières, "Labels and event processes in the asbestos operating system," *ACM Trans. Comput. Syst.*, vol. 25, no. 4, 2007.

[11] S. Geller, "Information flow and execution policy for a model of detection without false negatives," in *Network and Information Systems Security (SAR-SSI), 2011 Conference on*, 2011, pp. 1–9.

 ISBN: 978-1-61208-298-1

Firewalls Usability: An Experiment Investigating the Usability of Personal Firewalls

Bander AlFayyadh
Queensland University of Technology, Australia
Email: b.alfayyadh@student.qut.edu.au

Mohammed AlZomai
K.F. Security College, KSA
Email: zomaim@kfsc.edu.sa

Audun Jøsang
University of Oslo, Norway
Email: josang@mn.uio.no

***Abstract*—Poor usability of IT security systems and applications represents a serious security vulnerability, which can be exploited to compromise systems that otherwise could be considered technically secure. This problem is of particular concern with the huge number of users regularly connecting to the Internet but who know very little about the principles of IT security. Personal firewalls are important security mechanisms for protecting users against Internet security threats. However, the knowledge and skills required to effectively operate some aspects of a personal firewall may surpass the capability of the average user. In previous work, we conducted a usability evaluation of personal firewall by cognitive walkthrough against a set of security usability principles. We concluded that there are many usability issues of personal firewalls that can cause security vulnerabilities. In this paper, we report the results of a practical usability experiment with participants using commercial firewalls in a controlled environment. The experiment setup is described and participants' feedback and behaviour are analysed to evaluate the impact of usability of a modern firewall on the overall security of personal workstations.**

Keywords–Usability; Security; Firewalls.

I. INTRODUCTION

The number of computer and Internet users is huge and still growing worldwide, with client terminals roaming between wireless and wired networks that potentially enable access from the Internet to processes or to private data on the client terminal. Protecting private data in this environment is becoming more and more important. As computer users face threats and attacks they look for tools to protect their data. According to the Computer Crime and Security Survey [7], one of the most popular tools used for this purpose is a personal firewall. A firewall is defined on an abstract level as "an integrated collection of security measures designed to prevent unauthorized electronic access to a networked computer system" [2]. More specifically, a firewall is a checkpoint that controls and filters traffic between two separate networks or network segments, according to specific rules based on content or network parameters. Threats are prevented by denying access to or from specific hosts or processes when it is in conflict with the firewall rules. Generally, if used correctly a personal firewall should provide reasonably good security against network threats. The crucial point of operating a firewall is to specify the appropriate network filtering rules. Personal firewalls have become an essential part of online security. But as with many other security tools, a sound security system could be compromised by users' ignorance or carelessness about firewall operation [8], [9].

Inappropriate operation of security tools can in general be the result of human apathy towards security policies, but in many cases it may very well be due to the poor usability of the security tool or system itself. Usability of personal firewalls is especially important and critical. The target market for personal firewalls is typically the normal Internet user with little or no knowledge about IT security, so given the inherent sophisticated nature of personal firewall configurations, it is easy to understand why usability is a concern.

Poor usability of a security system can lead to serious consequences as pointed out by several authors. Whitten and Tygar's study [11], [10] on the usability of PGP showed that the security vulnerabilities were a direct result of usability problems. The same could be said about personal firewalls; personal firewalls usually run in the background and alert the user if needed, the alert can be as clear as a pop-up window or as subtle as a color change of a small icon in the system tray. The user typically reads for example the content of the pop-up window. Based on his/her understanding of the message from the firewall the user must make a security decision and potentially take some action that will affect the security of the system.

A novice computer user may not have the required level of knowledge to manage a firewall properly. They may not understand the terminology used by the firewall or the consequences of some of the decisions he/she is required to make. Users may often click away just to continue on with their computer related task, thus exposing themselves to possible threats that the firewall could have prevented. This behavior can be attributed to poor usability or users' apathy or maybe both. We aim to test the usability of firewalls by observing users' interaction with firewalls, specifically, what decision/action they make during that interaction and why they make it.

To conduct usability evaluation, we designed an experiment in a controlled lab environment. We configured several machines with selected firewalls and observed participants while interacting with the firewalls. We tried to make the experiment as realistic as possible by creating a familiar scenario that resemble normal computer usage for the participants. The scenario was not directly aimed at performing security tasks since security normally is not the primary goal of users when accessing the Internet. We asked participants to perform a simple task such as playing a game, during which several events (e.g., a pop-up message) caused by the firewall would occur, and that requires their attention. The participants were observed during the events, and based on the information provided by the firewall we examined whether they understood or did not understand the events.

 ISBN: 978-1-61208-298-1

In a previous work, we evaluated the usability of personal firewalls by cognitive walkthrough against a set of eight usability principles proposed by Jøsang *et al.* [5] which in turn were inspired by security principles suggested by the Belgian cryptographer Auguste Kerckhoffs [3], [6]. Jøsang *et al.*'s security usability principles are divided into principles for security action and security conclusion described as follows:

- A *security action* is when users are required to produce information and security tokens, or to trigger some security relevant mechanism. For example, typing and submitting a password is a security action.
- A *security conclusion* is when users observe and assess some security relevant evidence in order to derive the security state of systems.

The eight security usability principles are:

1) Security Action Usability Principles
   a) The users must understand which security actions are required of them.
   b) The users must have sufficient knowledge and the practical ability to make the correct security action.
   c) The mental and physical load of a security action must be tolerable.
   d) The mental and physical load of making repeated security actions for any practical number of transactions must be tolerable.
2) Security Conclusion Usability Principles
   a) The user must understand the security conclusion that is required for making an informed decision. This means that users must understand what is required of them to support a secure transaction.
   b) The system must provide the user with sufficient information for deriving the security conclusion. This means that it must be logically possible to derive the security conclusion from the information provided.
   c) The mental load of deriving the security conclusion must be tolerable.
   d) The mental load of deriving security conclusions for any practical number of service access instances must be tolerable.

Whenever one or more of these principles is violated during user interaction with the firewall we considered that to be a usability problem. Violating these principles will not necessarily indicate a security risk when using non-security software (e.g., a word processor) but it may cause security vulnerability when using security software (e.g., anti-virus filter or a firewall). The difference between normal usability and security usability has been pointed out in the literature [11]. Personal firewalls normally have decent usability from a traditional CHI point of view. However, when tested against these security usability principles, specific usability problems were detected.

The rest of the paper is organized as follows: Sec. II gives a brief description of firewalls interface. Sec. III describes the study design and experiment procedure. Sec. IV presents the experiment results; the results are analyzed in sec. V and discussed in sec. VI. The paper is concluded in sec. VII.

II. Personal Firewalls Interface

In this section we provide a brief description of general firewall interface and the way a user can interact with it. Personal firewalls are transparent and usually work in the background. When they need to communicate with the user they do it through a pop-up window or an alert message. When they communicate with the user it is either to alert for a possible threat or to ask them to make a decision regarding the function of the firewall.

Users can interact with firewalls through the following interface channels:

A. Main Menu

The main way to control and configure the firewall is through its main menu. The user can check here the status of the firewall, security status of the system, recent events, logs and other information.

B. Pop-Up Notifications

Pop-up notifications are commonly used by the firewall to inform a user of a current event which requires the user's attention. The pop-ups occur when the firewall requires the user to make a decision, or needs to inform the user of a decision or an event.

C. System Tray Notifications

It is common for firewalls to display a small icon on the bottom right of the screen in what is known in Windows as "System tray". The purpose of this icon is to provide quick access to the firewall menu; it also serves as warning mechanism or status alert through a change of color. Usually, it is green for "System Safe" status and red otherwise.

We suspect that the subtle change in color could go on unnoticed by users. We addressed that issue in our experiment.

III. Study Design

In this section, we will describe our study in detail. The goal of the study is to investigate whether users with little knowledge about IT security would be able to understand the information provided by a firewall when they are asked to make a decision.

We conducted a firewall usability experiment where we had participants operating computers in a lab environment while we observed how they dealt with firewall alerts and messages. In addition, the participants were asked to answer several questions before, during and after the experiment. A description of the questions is in section III-B. The result is shown in section IV.

 ISBN: 978-1-61208-298-1

A. Participants

The experiment target were computer users who were not skilled in IT security. For our selection process, we asked for participants who are not studying IT or working in the IT field. Most of our participants were from disciplines such as Law or Business. Another screening excluded some participants that had high IT skills gained from personal experience.

The participants age, sex and ethnicity varied. We did not record any personal details except their discipline/profession. However, the participants signed a consent form which contains their name, these forms are kept separate from the questionnaire forms.

There was no risk to participants other than those inherent to using a computer for about an hour. After the experiment, the participants were briefed about their performance and whatever questions they had were answered.

B. Questions

The questionnaire was divided into three groups of questions denoted A,B and C.

- **Question Group A.** These questions were given before the experiment, with the goal of determining the background knowledge each participant has or thinks that he/she has about firewalls. The questions are:
  Q1) Do you know what a firewall is?
  Q2) Do you know the purpose of a firewall?
  Q3) Do you know how to operate a firewall?

- **Question Group B.** These questions were given during the experiment, and were repeated for every event during the experiment. From the beginning of the event (e.g., A Pop-Up would appear) until the end of the event (When the user makes a decision and takes action). The goal was to determine how much the participant understood from the information contained in the alert, if the participant did not understand, then how would that affect his/her decision.
  These events are:
  1) Warning message alerting the user that a web browser is trying to access the interenet.
  2) Warning message alerting the user that an application from another machine is trying to access local files using a File Transfer Prorocol (FTP) client.
  3) Warning message alerting the user that a game he is playing is trying to send some information to an outside server.
  4) Change in color in the system tray icon of the firewall indicating a change in the system security status.

- **Question Group C.** These questions were given after the experiment. The participants answered several questions that should reflect their evaluation of the usability of personal firewalls during the experiment. The questions are listed below.
  While doing the experiment:
  Q1) Was it easy to make the decisions when prompted by the firewall?
  Q2) Did you make a decision that you thought might have been wrong but did it anyway in order to get on with your task?

C. Experiment Procedure

We prepared the experiment in a computer laboratory at QUT (Queensland University of Technology). The PCs were identical and all participants were familiar with the Windows operating system on the machines.

We had 30 participants for our experiment. Participants would come in and we would ask them to use the computer for a while until they are comfortable with the environment. Afterwards, we would explain to them the nature of the experiment and ask them to answer questions from group A of the questionnaire. We instructed the participants to behave as they normally do when using their own computers.

The participants were asked to perform simple tasks such as browsing the internet or playing a game. While they are doing that, an event caused by the firewall would occur. The event could be as direct as a firewall pop-up message that prompt for a decision or a subtle change of color in the firewall system tray icon. During each event the participants' behavior was observed and we would ask them several questions regarding each event. The questions would be about things such as their understanding of the event, why it occurred and what decision they made.

After several events, we conclude the experiment by asking participants to answer the last part of the questionnaire. Every participant did the exact experiment and dealt with the same events as every other participant. The duration of the experiment had an average of 40 minutes.

IV. Results

The study consisted of three stages. In the first stage the participant were asked to answer group A questions, their answers can be seen in Table I and Fig. 1.

TABLE I - Stage 1 results

Question no.	Yes	Somewhat	No	Total
Q1	14	9	7	30
Q2	10	11	9	30
Q3	6	10	14	30

Stage two represents the interactive part of the study. Participants had to deal with four firewall generated events while using a computer. Three of theses events were represented by firewall pop-ups or warning message that required the participant to make a decision and take action. In each of these three events, the participant must either allow the event or block it. The participant also may chose to close the warning window without taking an action.

The purpose of these three events is to evaluate the participant's ability to understand events created by the firewall and

 ISBN: 978-1-61208-298-1

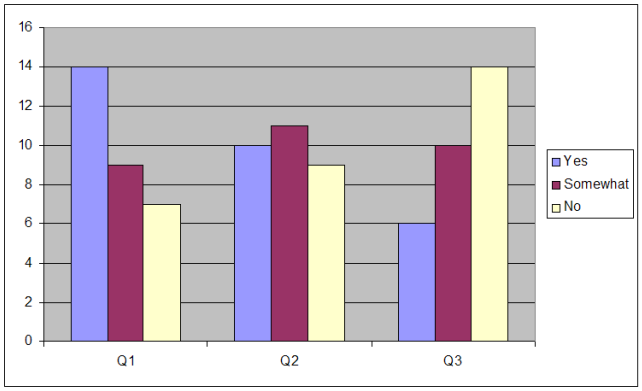


Fig. 1. Stage 1 results

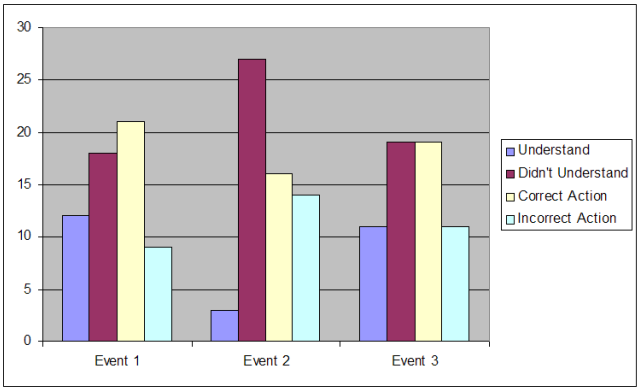


Fig. 2. Stage 2 results

hence make correct decisions. Accordingly, we interpret our data in terms of how many participants understood the event and how many took the correct decision.

We classify the data from the participant's interaction with these three events into the following four categories:

- **Category 1 (Understand)** Number of participants who understood the event: This occurs when a participant states that he/she understood the event's warning message, explains it correctly and makes the correct decision.
- **Category 2 (Didn't Understand)** Number of participants who didn't understand the event: This occurs when a participant states that he/she didn't understand the event's warning message or fails to explain it correctly.
- **Category 3 (Correct Action)** Number of participants who made the correct decision.
- **Category 4 (Incorrect Action)** Number of participants who made incorrect decision: This occurs when a participant made an incorrect decision or closed the event's warning message without making a decision.

Tables II, III and Fig. 2 show the data collected from observing participants interaction with the firewall and their answers to group B questions during their interaction.

TABLE II - STAGE 2 RESULTS (A)

Event No.	Understand	Didn't Understand	Total
Event 1	12	18	30
Event 2	3	27	30
Event 3	11	19	30
Total	**26**	**64**	**90**

TABLE III - STAGE 2 RESULTS (B)

Event No.	Correct Action	Incorrect Action	Total
Event 1	21	9	30
Event 2	16	14	30
Event 3	19	11	30
Total	**56**	**34**	**90**

In category 2 (participants who didn't understand the event's warning messages), when we asked why they didn't understand the event's warning message, the participants gave one of two reasons: The first was that the message language was unclear (e.g., too technical) and the second was insufficient information provided in the message contents. Table IV and Fig. 3 show the breakdown of the participants answers.

TABLE IV - REASONS FOR NOT UNDERSTANDING THE EVENT WARNING MESSAGE

Event No.	Unclear Language	Insufficient Information	Total
Event 1	12	6	18
Event 2	20	7	27
Event 3	13	6	19
Total	**45**	**19**	**64**

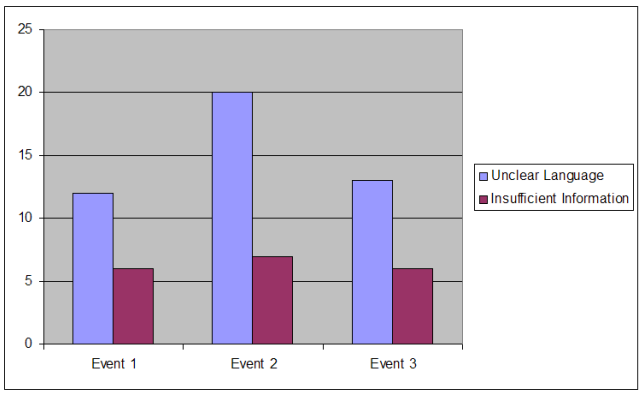


Fig. 3. Reasons for not understanding the event warning message

The last part of stage two was to examine if the participants would notice and respond to the fourth event (color change in the firewall icon in the system tray). While the participant is performing a computer task, we initiated some traffic that made the firewall change it's system tray icon from green to red, which may indicate a risk. Then we asked the participants if they noticed the event. For those who noticed the subtle color change we asked them if they understood the meaning

 ISBN: 978-1-61208-298-1

behind it. Accordingly, we classified the participants into three categories:

- Participants who didn't notice the event (23 out of 30).
- Participants who noticed and understood the event (6 out of 30).
- Participants who noticed the event but didn't understand what was the meaning of the icon alert (1 out of 30).

After stage two, participants were asked to answer questions from group C. The answers to these questions can be Yes, No or Somewhat. Their answers are shown in Table V and Fig. 4.

TABLE V - STAGE 3 RESULTS

Question no.	Yes	Somewhat	No	Total
Q1	7	4	19	30
Q2	16	5	9	30

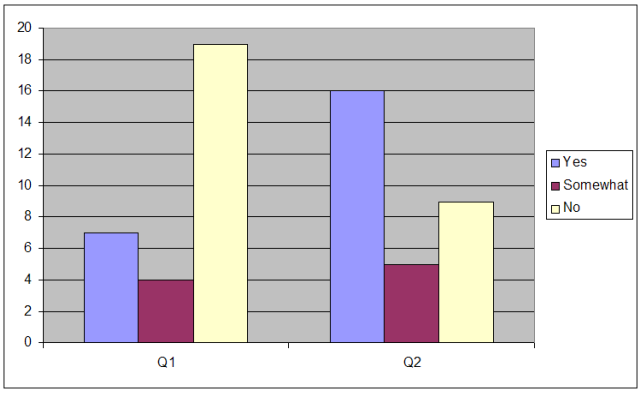


Fig. 4. Stage 3 results

V. ANALYSIS

A common behavior that we noticed is that at the beginning of the experiment participants would carefully examine each event, taking time to read messages in detail. But as the experiment progressed they tend to be quicker and dismissive. We interpreted this as the participant is "getting into" their assigned tasks. The tasks were chosen to be entertaining to create an atmosphere where the participant would be eager to continue on with the task, which is the case when he/she is working on their own computer.

From Table I, it is clear that the participants were not experienced in dealing with firewalls, almost 25% declared that they don't know what a firewall is exactly, and 30% stated that they don't know what a firewall is used for. Also, from their answers to the first group of questions, about 50% said they do not know how to operate a firewall and this was confirmed from our observation as well. We put their general skill level from low to moderate.

Stage two of the study in which participants interact with the firewall is the significant part of the study, we observed how the level of usability of the firewall affected the participants decision making when faced with security concerns. Tables II, III and IV represent the result of this stage.

Table II shows that less than a third of the warning messages (26 out of 90) that occured during the experiment were understood by participants, which means that more than two thirds of the firewall warning alerts were not understood by the participants. This is an alarming indication of serious usability weakness in the way firewalls give feedback to users.

Table IV shows the reason why those warning messages were not understood. When asked, 70% of the participants said the warning messages contained "Unclear Language" while 30% said it was due to "Insufficient Information" in the content of the warning message provided by the firewall.

Even though only 29% of the warning messages were understood by participants, it was interesting to see that 62% (56 out of 90) of the actions taken by participants were correct decisions (see Table III). This means even though some participants did not understand the message, they still made correct action.

In the fourth event of stage 2, only 23% of the participants noticed the change of color from green to red in the firewall system tray icon while 77% did not.

Table V shows that 63% of the study participants do not consider making decisions when prompted by the firewall an easy task. Also, more than 50% of the participants said that the firewall was an obstacle to them while working and that they made decisions that might have been wrong in order to get on with their task.

It was interesting to see how users' behavior changes when we set the firewall to the maximum security settings. This caused the firewall to produce more alerts than usual, resulting in users becoming frustrated after few alerts and getting to the point were they started closing the alert windows without reading them.

Fig. 5. Firewall alert when trying to open a web browser

 ISBN: 978-1-61208-298-1

VI. Discussion

Previous studies have reported usability issues in personal firewalls [4], [1]. Our study provides more detailed insight into the usability problems of personal firewalls. In particular we found that many users can be considered totally ignorant about firewalls, and the question is whether this group of people get any benefit from personal firewalls at all.

An interesting behavior is that in some cases, a participants would think that he/she understands the message content provided by the firewall in a warning and therefore he/she is sure they are making the secure decision. However, when asked to explain the message, it was clear they understood it completely or partially wrong. When we explained it to them, we asked them if this would change their future behavior dealing with this type of warnings many said that it wouldn't. Their reasoning was that they always behave like that and nothing bad happens.

To evaluate the usability of firewalls in our study, we will examine the participants behavior against the security usability principles described in section I.

One of the messages that appeared while participants were working on their tasks is a warning that an application from another machine is trying to access local files using an FTP client, 90% of the participants did not understand the meaning of that message or why it appeared. They described the content of the message to be difficult for them to understand since they didn't have any knowledge or enough knowledge of what an FTP client is. Clearly, this is a violation to principle 2.a. described in section I.

When a participant tries to open a web browser (e.g., Firefox) he would create an event, in this case a pop-up warning (see Fig. 5) that Firefox is trying to access the internet. Although this action was initiated by the participant, 60% of them said they don't understand this message even though the browser name was mentioned in the pop-up. Again, this violates principle 2.a.

Two reasons were given by the participants for not understanding the firewall warning messages, the first was due to unclear language in the message, which represents a violation to principle 1.a, the second reason was that the message does not provide sufficient information, this is clearly violates principle 1.b. The color change in the firewall system tray icon from green to red was not noticed by 77% of participants, this violates principal 2.a.

About 63% of the study participants stated that it was difficult for them to make a decision when prompted by the firewall and more than 50% considered the firewall to be an obstacle while working on the experiment. This represents an obvious violation of principles 1.d and 2.d.

As the complexity of firewalls warning messages increases, the participants make hasty or random decision in order to get their work done. In general, frustrated users usually ignore or bypass sound security measures when faced with tedious and sophisticated security tasks, which in turn make these measures, and the tools that provides them -such as firewalls- ineffective.

Finally, our study gives a strong indication that firewalls suffer from a serious security usability problem which make them insecure. Therefore, it is necessary to improve the usability of firewall to make them a more secure tool.

VII. Conclusion

In this paper, we presented a study on the usability of personal firewall by conducting an experiment that investigated whether users understand the information provided in warning alerts/messages from the firewall and how they would make security decision based on these alerts/messages. This study has shown that feedback to users from the firewall is mostly not understood. In case the general meaning of the messages are understood the users do not see the possible consequences this could have on the security of their system.

Firewalls act as a protective barrier between users (usually not very skilled in IT) and skilled attackers. This reality makes the job of the firewall interface hard when trying to maintain a certain level of simplification without losing technical details while giving feedback to users. In conclusion, personal firewalls typically fail to provide an adequate user interface to users. This seems to be a relatively hard problem to solve, because it would need to include an element of security learning, as well as an improved interface design.

As a final remark it can be noted that the term "firewall" itself might be part of the problem, because it gives wrong mental associations. The term "firewall" indicates that it is impenetrable and can stop all malicious traffic, which is inaccurate. A more accurate term would, e.g., be "Check Point" because it clearly indicates the aspect of checking the traffic. People would also more easily understand the the check point needs specific instructions about what should be allowed to pass and what should be stopped.

References

[1] B. Alfayyadh, J. Ponting, M. AlZomai, and A. Jøsang. Vulnerabilities in Personal Firewalls Caused by Poor Security Usability. In *Proceedings of IEEE International Conference on Information Theory and Information Security.*, 2010.

[2] Online Dictionary. http://www.dictionary.com. Retrieved: June-2013.

[3] P. Gutmann and I. Grigg. Security Usability. *IEEE Security and Privacy*, 3(4):56–58, 2005.

[4] A. Herzog and N. Shahmehri. Usability and Security of Personal Firewalls. In *New Approaches for Security, Privacy and Trust in Complex Environments*. 2007.

[5] A. Jøsang, B. AlFayyadh, T. Grandison, M. AlZomai, and J. McNamara. Security Usability Principles for Vulnerability Analysis and Risk Assessment. In *proceedings of ACSAC 2007 - Annual Computer Security Applications Conference*, Dec 2007.

[6] A. Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, Vol. IX(38):5–38 (January) and 161–191 (February), 1883. Translation available at F. Petitcola's Website: http://www.cl.cam.ac.uk/∼fapp2/kerckhoffs/.

[7] R. Richardson. CSI/FBI Computer Crime and Security Survey. Technical report, Computer Security Institute, San Francisco, USA, San Francisco, USA, 2003.

[8] M. A. Sasse. Computer Security: Anatomy of a Usability Disaster, and a Plan for Recovery. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI2003), (Workshop on Human-Computer Interaction and Security Systems)*, 2003.

 ISBN: 978-1-61208-298-1

[9] M. A. Sasse and I. Flechais. Usable security: What is it? How do we get it? In L.F. Cranor and S. Garfinkel, editors, *Security and Usability: Designing Secure Systems that People Can Use*. O'Reilly, Sebastapol, CA, USA, 2005.

[10] A. Whitten and J.D. Tygar. Usability of Security: A Case Study. Computer Science Technical Report CMU-CS-98-155, Carnegie Mellon University, 1998.

[11] A. Whitten and J.D. Tygar. Why johnny can't encrypt: a usability evaluation of pgp 5.0. In *Proceedings of the 8th USENIX Security Symposium (Security'99)*, 1999.

 ISBN: 978-1-61208-298-1

User Authentication Method with Mobile Phone as Secure Token

Ryu Watanabe and Yutaka Miyake
KDDI R&D Laboratories, Inc.
Ohara 2-1-15 Fujimino Saitama, Japan
Email: ryu@kddilabs.jp and miyake@kddilabs.jp

***Abstract*—In this paper, we propose a user authentication method with mobile phone. In our proposal, a mobile phone with a Subscriber Identity Module (SIM) card is used as security token for user authentication on WEB services. This authentication method named SIM-based authentication provides a secure authentication because of robustness of the SIM card. However, currently, the use of SIM-based authentication is limited to mobile phone. The terminals without a SIM card cannot use this style. In addition, only the mobile operator can use this method because the shared key hidden in the card is not open to service providers. In order to overcome this limitation, we realize pairing between mobile phone and terminals, which do not have SIM cards and the SIM-based authentication method can be applied to the terminals. In addition, the single sign on (SSO) technique is also applied in order to provide SIM-based authentication for service providers.**

***Keywords*—authentication; SIM; identity management.**

I. INTRODUCTION

Recently, Internet services have become more attractive and many users frequently use the services. The services are varied, and online shopping or auction sites are common. User authentication function is important for services that require payments. Many Internet sites still use ID/password (PW) pairs for this purpose. However, malicious attack techniques have recently improved [1][2][11][12], and thus the number of incidents has increased. In order to cope with this problem, some sites apply multifactor authentication methods [3], such as biometrics or One Time Password (OTP). The use of Public Key Infrastructure (PKI) also provides secure authentication. Some Internet banking sites require the random number table for executing money transfer. These methods are practical for malicious attacks. However, such methods usually require dedicated devices or items, for instance, sensors for fingerprints, a one time password generator or random number table. Therefore, more convenient user authentication method is required for Internet service use. On the other hand, in the case of a mobile phone, another authentication method exists. So, a mobile phone uses a subscriber identity module (SIM) for the connection to the base station. (The Universal Integrated Circuit Card (UICC) is the different name of the SIM card. In this paper, we use the word SIM for the card.) The SIM card is removal and can be transferred between different mobile phones. By using SIM cards, mobile carriers can identify users. Using SIM card for user authentication has some good features. The SIM card is one of the tamper resistant modules and it is difficult to duplicate. In addition, users do not have to memorize some secret information such as password. However, currently, use of this authentication method is limited to the connection to the base station for mobile phones or device authentication on Wi-Fi communication. Moreover, only the mobile carrier can apply this authentication method because the secret information inside SIM cards is not open to service providers. Therefore, when the SIM-based authentication method can be used for user authentication of WEB service use, a more secure user authentication method is brought to both users and service providers. In addition, as described previously, this method requires a SIM card. Other terminals, such as a tablet or notebook PC, do not have SIM cards. In this case, if the mobile phone with a SIM card cooperates with mobile terminals without SIM cards, then secure authentication can be utilized on all mobile terminals. In order to cope with these two problems, the authors we proposed the SIM-based authentication method for mobile terminals without SIM cards. We also implemented a proto-system based on the proposal. The authorswe confirmed the functions in an Internet environment. In order to provide the SIM-based authentication for Internet service providers, the single sign on (SSO) technique is used. In addition, for secure usage, pairing between mobile phone with mobile terminal without SIM is confirmed at application level. After this section, we denote related work in section II. In section III, IV and V, we explain our proposal, the implementation based on our proposal, and discussion, respectively. In section VI, the conclusion is drawn.

II. RELATED WORK

A. Authentication method

User authentication methods for Web services are categorized into three basic factors: the knowledge base factor, ownership factor, and inherence factor. The first factor uses something the user knows (e.g., password or Personal Identification Number (PIN)). The second factor uses something the user has (e.g., secure token, software toke, cell phone) [5]. The third factor uses something the user is or does (e.g., fingerprint, deoxyribonucleic acid (DNA) sequence, face) [4]. Because of security problems, the authentication methods that use the last two factors or combination of multiple factors have gradually become widespread.

1) SIM-based authentication: The SIM-based authentication method is one of the authentication methods with the ownership factor. Firstly, this method is used for only the connection between a base station and mobile phone. Then, the method is extended to identify user terminal on Wi-Fi connection as EAP-SIM or EAP-AKA. EAP means extensible

 ISBN: 978-1-61208-298-1

authentication protocol and AKA means authentication and key agreement [6], which are standardized in 3GPP [7] and IETF. EAP protocols are used on user device authentication on Wi-Fi connection establishment. In SIM-based authentication, by checking the response managed with the secret key, the mobile carrier authenticates the user terminal. So, the secret is shared with the authentication server of the mobile carrier.

Currently, the theses authentication methods are limited and not used for user authentication for Web services. For this purpose, another SIM-based authentication method was proposed, which is called Generic Bootstrapping Architecture (GBA). Basic idea is same. It also uses a dedicated shared secret key installed inside the SIM card. The design is based on the AKA. In the GBA method, the user, who wants to use a Web service (called the Network Application Function (NAF)), is authenticated by the Bootstrapping Server Function (BSF), then the BSF issues a dedicated temporary identifier and key pair for the services, and then authentication is executed between the user terminal and the service. The lifetime of the key is limited. Therefore, when it is expired, the user has to restart the sequence. The GBA is standardized; however, it does not use current Internet services because the browser function for the GBA method from the Web browser is not prepared. In addition, not all SIM cards can use the GBA function. Some function is also required for the SIM cards. The Application programmable interface (API) for this purpose is under standardization.

Therefore, in our proposed scheme, we customized the mobile phone and test type card to use the SIM-based authentication function based on the EAP-AKA' authentication protocol, which is derivation of EAP-AKA.

B. Identity Management Technique

The identity management (IdM) technique is used for control of user information. Originally, it was invented for control of user account management on an intranet. Then, the target area of the IdM technique expanded to the Internet. One of the main functions of the IdM technique is SSO, which realizes centralized authentication. A user account control server named identity provider (IDP) authenticates the user then transmits the result to the service server named service provider (SP). The SP provides the user a service depending on the results. Figure II-B shows the concept of SSO. In order to prevent linking, which is a privacy problem, pseudonyms are used for user identification between servers. Therefore, if service servers conspire with each other, the user identifier on both servers cannot be known. For realizing the IdM technique, some specifications and implementations are released from standardization organization. OpenID [9] and SAML [8] are representative of them.

1) OpenID: The OpenID mechanism is a decentralized authentication scheme for the SSO mechanism. OpenID users identify themselves with a URI and XRI. In the OpenID scheme, the identity provider and service provider are referred to as the OpenID provider (OP) and relying party (RP), respectively. The latest version of OpenID is OpenID connect. The OpenID technique is also used on the Internet service site. The combination of SIM-based authentication method and OpenID has been proposed [10]. In this identity management

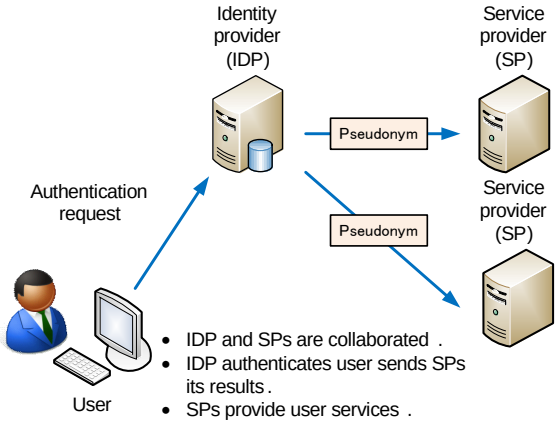


Figure 1. Concept of SSO

system, users and service provider are provided secure and convenient authentication.

2) SAML: SAML (Security Assertion Markup Language) is an XML-based open standard data format produced by OASIS for exchanging authentication and authorization data between an identity provider and a service provider. Based on the SAML, some Web Brower single sign on profiles are defined such as liberty identity web services framework.

III. Our Proposal

A. Requirement

The concept of our proposal is described. A user wants to use some Web service on a mobile terminal, such as tablet or note PC, which do not hold a SIM card inside. It is supposed that such terminals have a wireless local area network (LAN) connection to the Internet. Usually, the service provider authenticates the user by using ID/PW pair. Instead of using the ID/PW pair, in our proposal, mobile phone is used as key device for user authentication. So, the system requires a mobile phone by the user. Under this concept, the SIM-based authentication can be used for any kind of terminals.

However, there are some problems. First, the mobile terminal and mobile phone have to contact each other by some method. Second, service providers cannot use SIM-based authentication directly. In addition, some security mechanism is required between the paired terminal and phone in order to avoid malicious use from another user.

The requirement for the system is summarized below.

1) Connection between mobile terminal and mobile phone
2) Delegated authentication scheme
3) Safety mechanism for avoiding malicious use

B. System Concept

Based on the requirements described above, we designed the prototype. For the first requirement, some connection methods can be used. In order to communicate with each other,

 ISBN: 978-1-61208-298-1

a bidirectional connection is required and a unidirectional one is not appropriate. Therefore, we decided to use Wi-Fi and Bluetooth connections. In addition, we assume this connection is already established. So, the establishment of the communication method between the mobile terminal and mobile phone is out of the scope of this paper.

For the second requirement, we applied the identity management mechanism. So, the SSO mechanism is used. The mobile carrier, who knows the shared secret, behaves as the IDP. And the service providers unite with the mobile carrier and delegate user authentication to it. So, using tThe identity management technique is used better for security reasons to avoid privacy problems such as linkability.

For the third requirement, we applied mutual authentication on the applications for a SIM based-authentication. Currently, the browser API is not prepared for mobile phones. In addition, for our proposal, the mobile terminals without SIM cards and mobile phones have to communicate on the application level. Therefore, we implemented mobile applications for both terminals and phones. These applications also have to associate with each other before executing the SIM-based authentication.

C. Malicious attacks

Here, we denote malicious attacks, which we suppose. The aim of the malicious users is spoofing. They want to be authenticated as legitimate user. However, they do not have legitimate SIM or cannot duplicate the SIM because of the SIM resiliency. In addition, the SIM-based authentication method has a protection against eavesdropping. It uses a challenge and response (CHAP) type authentication and manages its lifetime. Therefore, if a response is eavesdropping, it cannot be used different session. In addition, the SIM-based authentication applies mutual authentication in the message exchange. So, user can notice that the server is faked, even though the attacker prepares a fake server and induce to it by phishing mail.

However, if the attacker can replace the application for our proposal, they can achieve man-in-the-middle type attack because the attacker can put modified messages both user terminal and server via the contaminated application. In order to cope with this problem, in our proposal, the applications share the secrete information. By using this information, the applications both mobile phone and mobile terminal confirm their legitimacy. Therefore, it is required that the secret is pre-shared among all the application and implemented securely. For this purpose, obfuscated codes can be used. However, in current version, this countermeasure against reverse-engineering is not supported and we will apply the feature in next version.

The totalwhole concept of the proposal is shown in Figures 2 and 3. Figure 2 shows the preparation before our proposed scheme and Figure 3 shows the communication among servers, user terminal and user mobile phone, respectively. So, in our proposal, the SIM-based authentication scheme and SSO technique are utilized.

IV. IMPLEMENTATION

A. System Specification

Based on the proposal, we implemented the proto-system. The system specification is summarized on Table I.

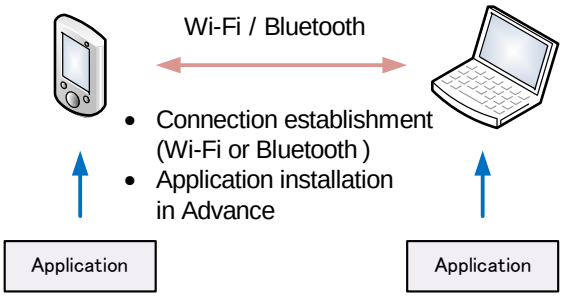


Figure 2. System preparation for our proposal

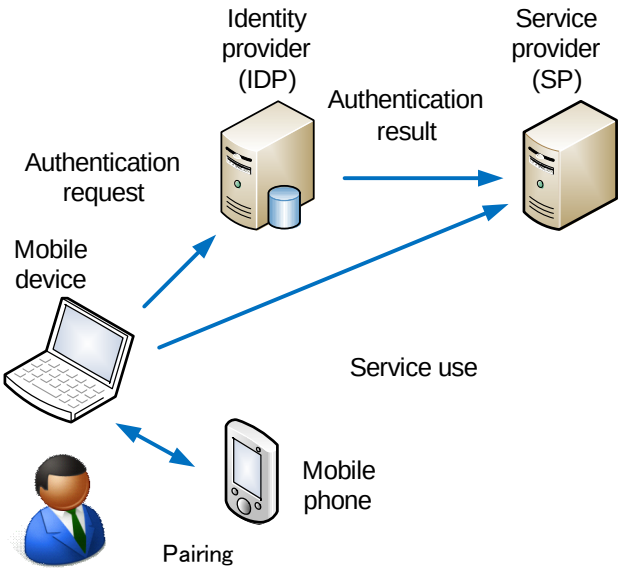


Figure 3. Concept of our proposal

As a mobile phone, we used the Android OS smartphone Nexus One because the kernel of the phone is open and can be modified. For the OpenID platform (IDP and SP servers), the Linux base OS (CentOS) and open module (php OpenID) are used. In addition, Windows OS tablet is used for the user mobile terminal. In our proposal, the EAP-AKA' method is used as SIM based authentication method. As mentioned previously, not all the SIM card is hold the GBA function, so, in our implementation, we selected EAP-AKA based authentication method. Usually, the mobile phone OSs including Android OS do not provide access to SIM card. Therefore, we modified the OS for this purpose. So, the use of the proto-type is limited currently.

In order to limit restrict the mobile terminals, which can contact to the mobile terminal, before the executing SIM-based authentication, named EAP-AKA', the pairing between applications on user mobile terminal and user mobile phone has to be confirmed. The steps for the pairing are summarized below. Before this step, the installation of dedicated applications for both user terminal and phone in advance.

1) At first, user operates the application on mobile phone

Table I. SYSTEM SPECIFICATIONS

Mobile Phone	
Terminal	Nexus One
OS	Android (Customized)
Mobile terminal without SIM	
Windows tablet	Toshiba WT301
OS	Windows 7
OpenID platform	
Server OS	CentOS
OpenID module	php-openid
Application	
Module	java

and change the mode to active in order to accept the request from user terminal. Only the active mode, the application listens to the request from mobile phone. (At this time, the wireless connection between mobile phone and mobile terminal is supposed to be constructed.)

2) Then user sends request from the mobile terminal to the mobile phone.
3) The mobile phone, which catches the request, executes mutual authentication using the secret hidden in the application.
4) If the application on the mutual authentication is successfully achieved on them, then the mobile phone shows accept code (four-six digit number) on the display.
5) User inputs the number to the application on mobile terminal. Then the pairing process is finished.

This process is similar to the paring method of Bluetooth. The purpose is same. Once the pairing is constructed, the user terminal can send authentication request to the mobile phone.

The flow sequence of our proposed scheme is described below. Before this authentication flow, ID association between an IDP and SP has been completed successfully in advance. The figure 4 shows the whole sequence.

1) A user accesses an SP from the browser on the mobile terminal without a SIM card or UIM.
2) The SP redirects the request to IDP.
3) The IDP generates the session ID for the user authentication and sends it to the browser on the user's mobile terminal.
4) The browser launches the application on the mobile terminal and passes the session ID to the application.
5) The application on the mobile terminal searches the mobile phone, which is on standby near the mobile terminals.
6) Connection is confirmed between applications.
7) The application sends an authentication request to the authentication application on the mobile phone, which is found in the previous step.
8) The SIM-based authentication method (EAP-AKA') is executed between the mobile phone and the IDP.
9) The IDP redirects the user authentication result to the SP.
10) The SP generates the session ID for service provisioning.
11) The SP provides the Web service to the user.

The time for the authentication was measured on the prototype. The process time on authetication server is less than 1 second without network delay in ten times average.

V. DISCUSSION

For the pairing between applications on both mobile phone and mobile terminals, the secrete parameter number, which is decided by user is required. If the number parameter is securely protected by user, no one can know it except the legitimate user. Therefore, malicious user cannot execute pairing behind legitimate user's back. In addition, the applications mutually authenticate each other before pairing by using securely embedded secret parameter . When an application cannot validate the other application, the request is denied. So even though a malicious user remakes the application, the modification is detected.

Whole communication between the terminals and the mobile phone IDP and SP are protected by using Secure Socket Layer (SSL) connection, therefore, secrets that is exchanged between them cannot be eavesdropped. Even though, the parameters are stolen by some method parameters included the response has life time and managed with the communication session. So it cannot be reused.

Our proposed system applies the SSO technique for user authentication on SP. Therefore, unwanted information about user is not sent to SP. The important parameter such as telephone number i is not revealed to service providers.So, the IMSI that is one of the important parameter is also kept inside IDP.

VI. CONCLUSION

In this paper, we proposed a SIM-based authentication method for mobile terminals. By using a mobile phone with SIM card as an authentication token, the secure user authentication is realized on mobile terminals without SIM card for WEB service use. The authorswe believe that by using our proposal, more flexible and useful user authentication is brought to both users and service providers.

REFERENCES

[1] A. Vorobiev and J. Han, "Security Attack Ontology for Web Services," Semantics, Knowledge and Grid, 2006. SKG '06. Second International Conference on, 2006, pp. 42-47.

[2] T. S. Chen; F. G. Jeng, and Y. C. Liu, "Hacking Tricks Toward Security on Network Environments," Parallel and Distributed Computing, Applications and Technologies, 2006. PDCAT '06. Seventh International Conference on, 2006, pp. 442-447.

[3] S. R. Basavala, M. Kumar, and, A. Agarrwal, "Authentication: An overview, its types and integration with web and mobile applications," Parallel Distributed and Grid Computing (PDGC), 2012 2nd IEEE International Conference on, 2012 pp. 398-401.

[4] S. Asha and C. Chellappan, "Authentication of e-learners using multimodal biometric technology," Biometrics and Security Technologies, 2008. ISBAST 2008. International Symposium on, 2008, pp. 1-6.

[5] H.K. Lu and A. Ali, "Communication Security between a Computer and a Hardware Token," Systems, 2008. ICONS 08. Third International Conference on 2008, pp. 220-225.

[6] J. Arkko, H. Haverinen, "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)," RFC 4187, IETF.

[7] 3GPP, ⟨http://www.3gpp.org/⟩ 07.07.2013

 ISBN: 978-1-61208-298-1

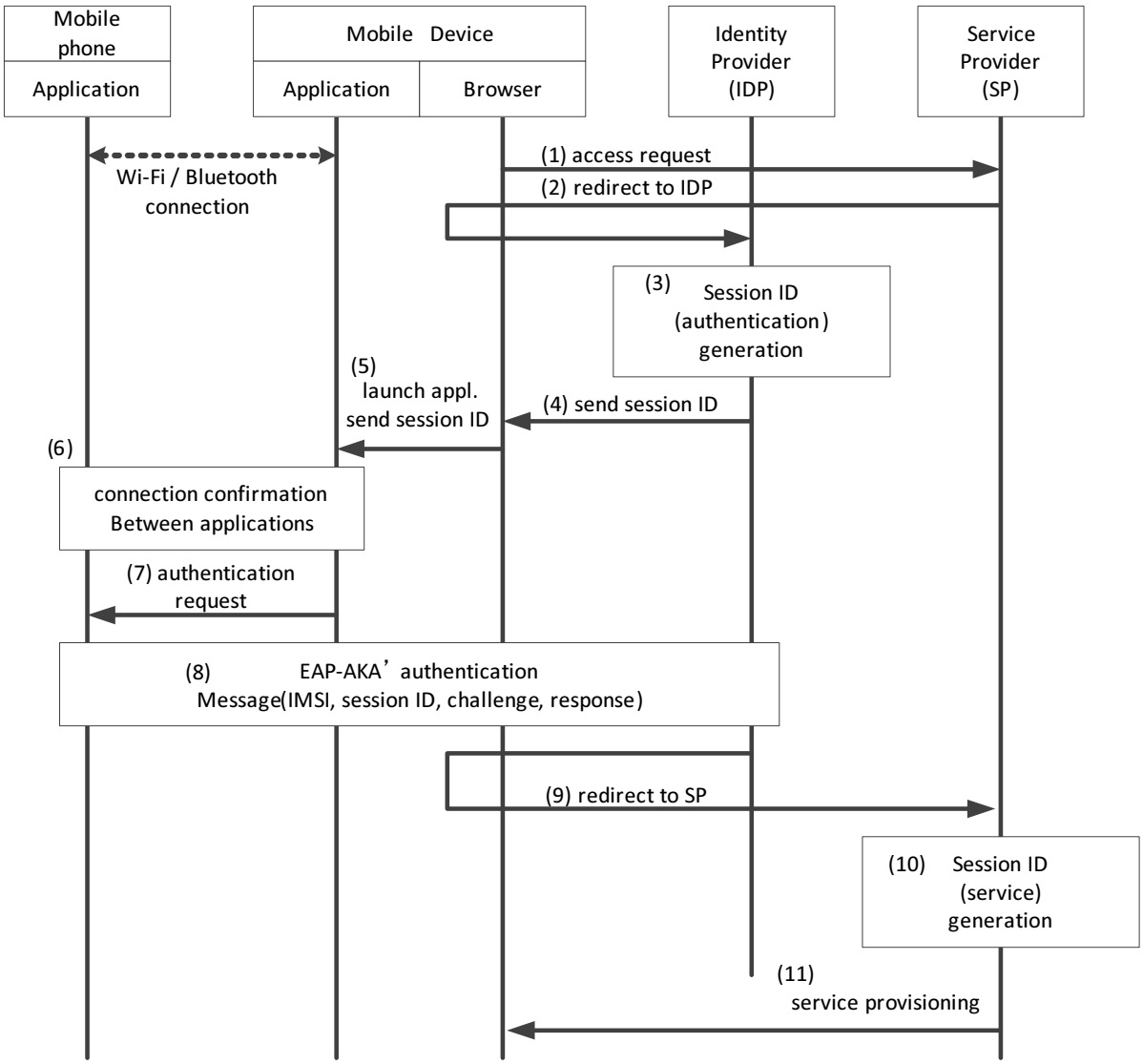


Figure 4. Sequence of our proposed method

[8] OASIS SAML V2.0, ⟨http://www.oasis-open.org/specs/index.php#samlv2.0⟩ 07.07.2013.

[9] OpenID, ⟨http://openid.net/⟩ 07.07.2013

[10] A. S. Ahmed and P. Laud, "Formal Security Analysis of OpenID with GBA Protocol," MobiSec, volume 94 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Springer, 2011, pp. 113-124.

[11] H. M. Sun, Y. H. Chen, and Y. H. Lin, "oPass: A User Authentication Protocol Resistant to Password Stealing and Password Reuse Attacks", Information Forensics and Security, IEEE Transactions on, Vol. 7, 2012, pp. 651-663.

[12] C. Xu and Y. Yang, "Password guessing attack on a key exchange protocol based on ECDLP", Progress in Informatics and Computing (PIC), IEEE International Conference on, Vol. 1, 2010, pp. 449-452.

 ISBN: 978-1-61208-298-1

The All-Seeing Eye: A Massive-Multi-Sensor Zero-Configuration Intrusion Detection System for Web Applications

Christoph Pohl

Munich IT Security Research Group (MuSe)
Department of Computer Science and Mathematics
Munich University of Applied Sciences
Munich, Germany
Email: christoph.pohl0@hm.edu

Hans-Joachim Hof

Munich IT Security Research Group (MuSe)
Department of Computer Science and Mathematics
Munich University of Applied Sciences
Munich, Germany
Email: hof@hm.edu

***Abstract*—Timing attacks are a challenge for current intrusion detection solutions. Timing attacks are dangerous for web applications because they may leak information about side channel vulnerabilities. This paper presents a massive-multi-sensor zero-configuration Intrusion Detection System that is especially good at detecting timing attacks. Unlike current solutions, the proposed Intrusion Detection System uses a huge number of sensors for attack detection. These sensors include sensors automatically inserted into web application or into the frameworks used to build web applications. With this approach the Intrusion Detection System is able to detect sophisticated attacks like timing attacks or other brute-force attacks with increased accuracy. The proposed massive-multi-sensor zero-configuration intrusion detection system does not need specific knowledge about the system to protect, hence it offers zero-configuration capability.**

***Keywords*—intrusion detection, sensor, brute force, timing**

I. INTRODUCTION

Intrusion Detection Systems (IDS) in combination with firewalls are the last defense line in security when protecting web applications. The purpose of an IDS is to alert a human operator or a Intrusion Prevention System that an attack is in preparation or currently taking place. The configuration of an IDS typically involves a configuration that must be adapted for each system to protect. The massive-multi-sensor zero-configuration intrusion detection system for web applications presented in this paper (called All-Seeing Eye in the following) does not need any adaption to the system to protect, hence is very easy to use.

One common challenge for web applications is the detection of timing attacks. A timing attack is an attack, which uses time differences between different actions to gain informations. Intrusion Detection Systems typically use sensors to collect data. In this work, a sensor describes a data source that provides data useful for attack detection. Useful in this context means that the data must be linked to actions of a web application. Data of sensors is analyzed by All-Seeing Eye to detect attacks. Current Intrusion Detection Systems are fairly limited in the number of sensors they use. All-Seeing Eye increases the number of available sensors by injecting code into common web application frameworks or even into web applications to provide additional sensors. A further increase in the number of sensors results from the combination of multiple sensors into one new sensor. Sensors may collect data from any data source available on a web server, or on the network. Sensors include:

- Network sensors like TCP requests, or TCP flags (e.g., SYN or SYN/ACK) per period.
- Hardware sensors like CPU usage, memory usage, or fan speed.
- Kernel sensors like number of file IO handles, number of system calls, and the like.
- Software sensors like log file entries or software hooks.
- System sensors like alerts from other common Intrusion Detections Systems, e.g., Snort [1] .

The rest of this paper is structured as follows: Section II presents related work in intrusion detection using multiple sensors. Section III describes the concept and implementation of the sensors used by All-Seeing Eye. The use of multiple sensors to detect intrusions is described in Section IV. Section V evaluates All-Seeing Eye under different attacks, especially timing attacks. Section VI concludes the paper and gives an outlook on future work.

II. RELATED WORK

Anomaly detection is based on the hypothesis that there are deviations between normal behaviour and behaviour under intrusion [2], [3], [4], [5]. Many techniques have been researched for the detection like network traffic analysis [6], [7], [8], statistical analysis in records [9] or sequence analysis with system calls [10], [11], [12], [13]. A combination of this research with anomaly detection methods based on multiple sensors allows to find yet unknown attacks. Configuring intrusion detecting systems for one distinct system or one distinct vulnerability needs configuration with current solutions. The

 ISBN: 978-1-61208-298-1

solution presented in this paper does not need any configuration.

In [11], [12], it is proved that call chains of system calls show different behavior under normal conditions and under intrusion, hence intrusion detection is possible. However, a normal model must be trained using learning data to detect attacks. In [12], it is shown that normal behaviour produces fingerprintable signatures in system call data. A deviation from these signatures is defined as intrusion. This method is restricted to the usage of system calls and does not use more fine granular sensor data. In [14], a way to detect anomalies with information flow analysis is shown. Profiling techniques are used, injecting small sensors in a running application. They propose a model with clusters of allowed information flows and compare this normal model against actual information flow. Similar models are proposed in [15], [16], [17]. This approach is similar to our approach, but [14] focuses on offline audits for penetration testing. The approach presented in this paper is intended to be used online, hence it does not analyze the whole information flow but focuses on the method call chain, and is therefore more efficient.

In [18], it is shown that vulnerability probing can be detected using multiple sensors, especially sensor that calculate the possibility a resource is called by a user. These sensors are called access frequency based sensors. However, the system presented in [18] needs a lot of information about the system to protect (e.g., patterns describing legitimate resource calls), hence is difficult to deploy in the field. The solution presented in this paper does not need any configuration.

III. Sensors for a Massive Multi-Sensor Zero-Configuration Intrusion Detection System

A sensor describes a data source that provides useful data for attack detection. Useful in this context means that the data must be linked to actions of a web application. Data of sensors is analyzed by the proposed Intrusion Detection System All-Seeing Eye to detect attacks. Sensors are:

- Already available data sources like memory consumption of an application.
- Software sensors inserted into a web application or a web application framework.
- Advanced sensors that combine data from multiple other sensors. Figure 1, shows an example of an advanced sensor: sensor A and sensor B are combined into sensor SumSensor by adding the output of sensor A and sensor B at distinct points in time.

All-Seeing Eye depends on the availability of a large number of sensors that can be used for attack detection. The current implementation of All-Seeing Eye is using the following classes of sensors:

- Alert Sensors: alert sensors have an upper and/or a lower boundary for data measured by this sensor that triggers an alert.
- Sensors with non-string payload:
  - 64 Bit payload: These sensors have a payload. The payload typically consists of measurement values (e.g., time differences).
  - 32 Bit payload: The payload consists of four byte and is typically used for simple state (e.g. the user authentication state, either ”not authenticated” or ”authenticated”).
- String Sensor: the payload of sensors of this class consists of a simple string. This class of sensors is usually used in regular expressions.
- Filter Sensor: a sensor of this class filters the output of an existing sensor, e.g. by evaluating regular expressions.
- Aggregation Sensors: sensors of this class combine output of other sensors into a new sensor. Based on the time of aggregation, two subclasses are used:
  - One-Time Aggregation Sensor: a sensor of this class can combine values of other sensors, e.g. by addition, multiplication or logarithmic scaling.
  - Continuous Aggregation Sensor: sensors of this class combine values of other sensors over a certain period of time, creating an average or quantile.

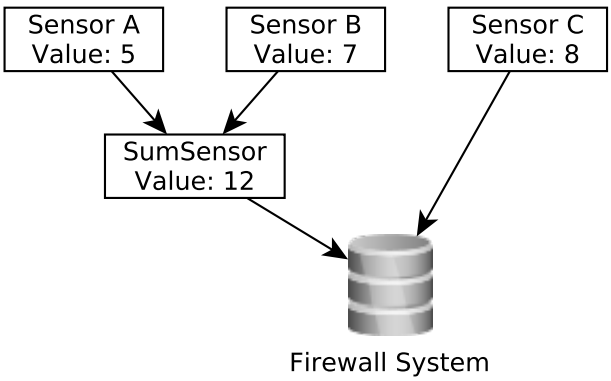


Figure 1: Datasource sensors and aggregation sensors

The current implementation of All-Seeing Eye uses software sensors, but the presented approach can be extended to hardware sensors. However, hardware sensors are not within the scope of this paper. All-Seeing Eye is intended to protect web applications relying on the Java runtime, however, the massive-multi-sensor zero-configuration approach may also be adapted to other execution environments. In the following, it is described how software sensors can be realized.

A. Sensor Implementation

Software sensors are implemented by injecting hooks at the beginning and end of methods of a web application. Hence, hooks are called before and after code execution of a method. With this approach, it is e.g. possible to measure the method execution time for each method used. It is also possible to identify the order of method execution. Hooks are injected directly into Java bytecode. It is not necessary to recompile any Java web application protected by All-Seeing Eye. Deploying All-Seeing Eye is as simple as copying the All-Seeing Eye jar file into the library directory of the web server. It is not

 ISBN: 978-1-61208-298-1

necessary to perform any configuration for the web application that should be protected, hence All-Seeing Eye is called "zero configuration". As injection technology AspectJ [19], with Load Time Weaving [20], is used. For the testbed used for the evaluation presented in this paper the aspect is placed in OpenCMS [21]. OpenCMS is a well known and widely used framework for Content Management. All-Seeing Eye takes care that methods used by the protected web application do not clash with method names used by All-Seeing Eye. Sensor data is written to a log file for further analyses.

A typical software sensor will produce data as followed:

```
timestamp , count , SID , value
```

where $Timestamp$ and $count$ together form a unique ID. $Timestamp$ is a UNIX timestamp of the event. $Count$ is a counter incremented each time adding a SID to a timestamp with at least one SID at the same timestamp. $Values$ is the sensor value. SID is a unique key describing one sensor in this format:

```
package . class . method . id . vid . vvid
```

where $package$ is the package in which the $class$ is located in which the $method$ is situated that was injected for this sensor. Id is used to distinguish overloaded methods from each other. Vid identifies a type of the value (e.g., String), and $vvid$ is an additional sensor for further use (e.g., if two sensors are injected into the same method).

A software sensor is unique with

```
package . class . method . id . vid . vvid
```

One way to minimize the output of sensors (and the number of data to write to the log file) is to produce no output for methods that have an execution time lower than the resolution of the timestamps (1 ms). It is suspected that these methods would not generate any interesting output as these methods are usually helper methods or wrappers.

B. Memory Consumption and Computational Overhead of the Proposed Multi-Sensor Approach

Software sensors have an impact on memory consumption of the protected web application. An additional 30 byte code operation are added to each method call of the web application by All-Seeing Eyes. Hence, the percentage of computational overhead heavily depends on the length of the methods of the web application.

In the testbed using OpenCMS (see Section V), the following experiment was conducted to evaluate the memory and computation overhead:

10,000 requests are sent to a web application using All-Seeing Eye. 100 runs were used, resetting the server after each run. Figure 2 shows the average memory consumption from 100 runs of the experiment compared to the memory consumption from 100 runs of the web application without All-Seeing Eye. It can be seen that a significant overhead of approximately 300 MB is needed to run All-Seeing Eye. This overhead is the result of buffering data before writing to the log files.

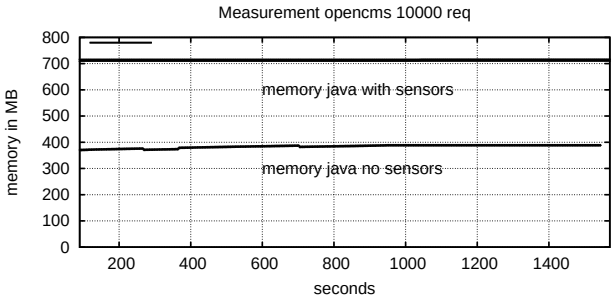


Figure 2: Memory comsumption of All-Seeing Eye

IV. Massive Multi-Sensor Zero-Configuration Intrusion Detection System

This Section describes the design of the proposed massive multi-sensor zero-configuration intrusion detection system All-Seeing Eye. All-Seeing Eye uses the software sensors, described in more detail in the last section, to calculate intrusion metrics. The metrics described in the following are focused on detection of outliers in timeline data values to detect brute force attacks. However, the approach presented in this paper is not limited to this attack class, it can be easily adapted to detect various other attacks. Even attacks on the business logic can be detected as the presented approach uses software sensors embedded in the code of an application. This is out of scope of this paper.

An advantage of All-Seeing Eye is that it allows to detect side channel attacks without knowledge of the web application which is to be protected. In the absence of an attack, there is a high correlation between method calls defined in a method chain. As shown in Section V a single call results in correlated calls (method chain) of other methods. The system under load shows the same correlations. These correlations are further called as fingerprint s. Under attack, however, the system shows a different behavior, hence allows to identify attacks, see Section V for details. All-Seeing Eye does not need a preconfigured or constructed normal model. For this approach the normal model is created from history. At time $t = 0$ it is always assumed that there is no attack, hence status c is always $c! = attack$. If there is no attack, the same fingerprint s should show up in each distinct time period T with the same probability. A deviation from the number of fingerprings (written as $|s|$) in a time period T is defined as possible intrusion. This behaviour is well known, as stated in Section II. The new approach here is the lack of need to define what a similar request is. The normal model is built using a quantile function, where the result is called α. α uses a floating history time period, which is defined as $n \times T$ and $t \in T$ are in state $c! = attack$.The multiplier n defines how much of the history is used. To control the sensitivity of the system, a configuration parameter p is used. In normal model α, a deviation is detected by:

$$c = \begin{cases} attack & \text{if } |s_{currentT}| \geq \alpha \times p \\ !attack & \text{if } |s_{currentT}| < \alpha \times p \end{cases} \quad (1)$$

This calculation is robust against statistical outliers and can be evaluated fast enough for real time calculation, in combination with structures related to sort optimisation. In further

 ISBN: 978-1-61208-298-1

researches these calculations will be done (together with other sensor calculations) with a graphical processing unit.

V. EVALUATION

For the evaluation of the massive multi-sensor zero-configuration intrusion detection system, two typical attacks on web applications are used: timing attacks and vulnerability probing. Especially timing attacks are hard to detect for common intrusion detection systems. For our test environment OpenCMS version 8.5.1 [21] is used as web application to protect. OpenCMS is a well known and widely used framework for Content Management.

A. Evaluation Environment

For the evaluation of All-Seeing Eye, a paravirtualized, openvz solution [22] is used. This approach has the advantage that is is very realistic compared to simulations. The presented hardware settings are the settings of the corresponding virtual machine. Table I lists hardware and software used for the evaluation.

TABLE I: Experimental setup

Hardware(Server)	
CPU	4 Cores (2.1GHZ on hostsystem)
Memory	6 GB Ram
Ethernet	Bridged at 1 GBit Nic
Software(Server)	
Server Version	Apache Tomcat 7.0.28 [23]
JVM	Sun 1.6.0.27-b27

B. Fingerprints of Normal Behaviour

To validate the hypothesis, that requests to the same target have the same fingerprint, the following experiment has been conducted.

First, a baseline is established for all other experiments. To do so, several requests are sent to the server and the server is restarted after each request. No interfering processes are running on this server.

After establishing the baseline, the whole website is crawled in a second step, ensuring that requests are sequential. The crawler is configured to request a single page and all depending images and scripts. To test the software under load in the third step, another crawler requests the server with 20 concurrent users, with a delay of one second between each request/user. Overall there were in average 20 requests per second for different websites. The second and third step have been repeated 100 times. In each run of the experiment, the metrics described above produced unique fingerprints for every requested target. This can be seen in Figure 3. In this figure the fingerprint of the start page and the request to the login page are extracted from the logged data. Under load the signature looks like the picture presented in Figure 4.

The result in Figure 3 shows that there are stable correlations between method calls. For better readability, points that differ less then 3 milliseconds are averaged. The experiments show that it is possible with All-Seeing Eye to identify similar requests using their fingerprint.

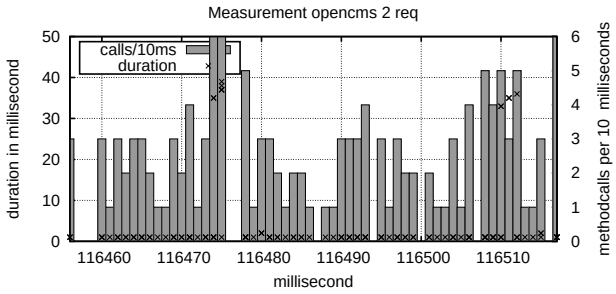


Figure 3: Two fingerprints of different requests

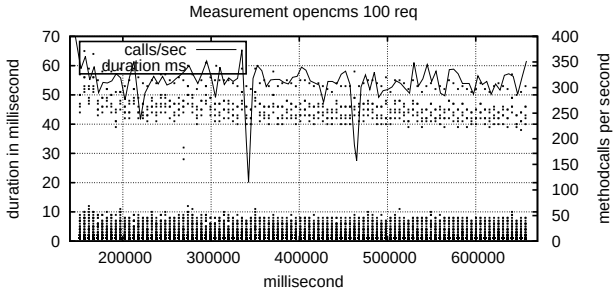


Figure 4: 100 requests on the same page

C. Fingerprints of Information Leakage and Probing for Vulnerabilities

OpenCMS version 8.5.1 has a known information leakage vulnerability, as described in [24]: using the default setting there is no limit for failed logins per time period. Also, a large amount of information is given in error messages, especially the error message ”this username is unknown”, if the given user name does not exist and ”password is wrong”, if the given password is wrong for an existing user allow an attacker to find valid user names, by trying possible user names from a dictionary and using error messages to find out if an user name is valid. This attack can be detected with a statistical analysis to detect the brute force analysis II. To do so, a detection technique needs to identify if a single resource is called many times but with different parameter in the request header. A normal model is needed for allowing patterns to test for deviations of the normal model. This needs deep knowledge of the system to protect and the vulnerability itself. All-Seeing Eye is able to detect this attack (and also other probings using brute force attacks), without this knowledge about system and vulnerability.

To evaluate if All-Seeing Eye can recognize brute-force attacks, an experiment has been conducted where an attacker probes the login page and tries to identify valid user names. The following pattern was used to generate the login requests:

```
http://192.168.2.89:8080/opencms/.../index.html?
    action=login&username=username_1&password=
    passwordnotindb...snippedEnd
//username_1..username_n in dictionary
http://192.168.2.89:8080/opencms/.../index.html?
    action=login&username=username_n&password=
    passwordnotindb...snippedEnd
```

The attacker used a dictionary with 1000 names for the

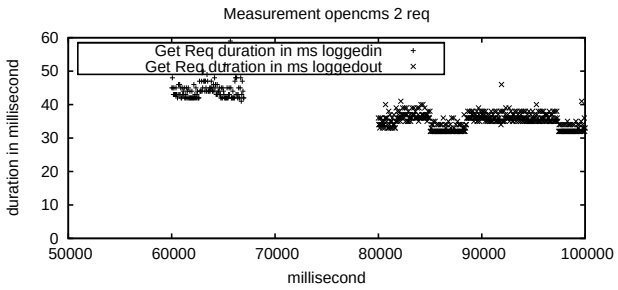


Figure 5: Time difference between logged in users and users not logged in on the start page

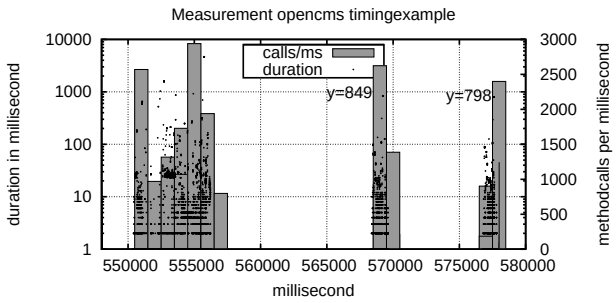


Figure 6: 100 requests on the same page

brute-force attack. To make detection harder, the attacker uses 50 different user agents as well as 20 different IPs. Only one valid username exists in the database.

Figure 4 shows a subset of 100 requests. From the figure it is obvious, that the probing attemps produce many similar fingerprints. It shows a high correlation between different requests, the sensor values and the order different sensors are called in one requests. This order and the values are stable over all requests. Hence, All-Seeing Eyes can easily detect a probing attack even if someone uses different header data. No a-priori knowledge of the system which is to be protected or the vulnerability itself is necessary.

D. Fingerprints of Timing attacks

OpenCMS version 8.5.1 is vulnerable to timing attacks as can be seen in Figure 5. The figure shows the times for loading of the start page for users that are already logged in as well as for users that are not logged in. A significant difference (849 ms to 798 ms) exists.

Using this timing difference an attacker can brute force user names by a dictionary attack. All logged in users can be detected. As with the information leakage and probing attack in Subsection V-C, current intrusion detection solutions need information about the system which is to be protected and the vulnerability to detect this attack.

To test if All-Seeing Eye is able to detect timing attacks without knowledge (zero-Configuration), the following experiment has been conducted: An attacker uses a dictionary of 1000 user names to execute the timing attack. Each request has different header data in the request only in the login name and the password as shown in following listing:

```
//successful login
http://192.168.2.89:8080/opencms/.../index.html?
    action=login&username=admin98&password=admin1...
    snippedEnd
//username not present, pwd not present
http://192.168.2.89:8080/opencms/.../index.html?
    action=login&username=usernamenotpresent&
    password=wrongpwd... snippedEnd
//username present with wrong password
http://192.168.2.89:8080/opencms/.../index.html?
    action=login&username=admin832&password=wrongpwd
    ... snippedEnd
```

Figure 6 shows an examples of a fingerprints of the experiment. The first fingerprint shows a successful login, the second fingerprint shows a login with no present username and third fingerprint shows a login with present username but wrong password. The results clearly show a correlation of the differences in order, time, and amount of method calls for each request. The presence of unusual high amount of similar fingerprints in a distinct time period allows to detect this timing attack. Hence, All-Seeing Eye can identify timing attacks.

VI. CONCLUSION

This paper presented the massive-multi-sensore zero-configuration intrusion detection solution All-Seeing Eye. All-Seeing Eye uses multiple software sensors for attack detection. Software sensors are automatically injected into web applications to protect. No change to deployed web applications is necessary and All-Seeing Eye does not need to be configured for each web application. All-Seeing Eye is zero-configuration. The evaluation showed that brute force attacks, e.g. timing attacks, can be detected by All-Seeing Eye using fingerprints automatically generated. It also shows that it is possible to identify similar requests without knowledge over the system or the definition of patterns. The current version of All-Seeing Eye produces a memory overhead of approximately 300 MB, which will be reduced in future versions using Graphical Processing Units (GPU) for fast calculation, reducing the need to write data to log files. This will also increase the amount of possible sensors, which can be calulated near real time. Further work will target different attacks like Blind SQL-injection, CSRF attacks, XSS attacks, and Command Injection.

ACKNOWLEDGMENT

This work is part of the project “Sichere Entwicklung und Sicherer Betrieb von Webanwendungen“ of the Munich IT Security Research Group funded by the Bayerisches Staatsministerium für Wissenschaft, Forschung und Kunst.

REFERENCES

[1] “Snort :: Home page,” http://www.snort.org/, retrieved 2013-04-11. [Online]. Available: http://www.snort.org/

[2] A. Patcha and J. . Park, “An overview of anomaly detection techniques: Existing solutions and latest technological trends,” Computer Networks, vol. 51, no. 12, pp. 3448–3470, 2007

[3] C. Kruegel and G. Vigna, “Anomaly detection of web-based attacks,” in Proceedings of the 10th ACM conference on Computer and communications security, ser. CCS ’03. New York, NY, USA: ACM, 2003, p. 251261

 ISBN: 978-1-61208-298-1

[4] G. Liepens and H. Vaccaro, “Intrusion detection: Its role and validation,” Computers & Security, vol. 11, no. 4, pp. 347–355, Jul. 1992

[5] D. Denning, “An intrusion-detection model,” IEEE Transactions on Software Engineering, vol. SE-13, no. 2, pp. 222–232, 1987

[6] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft, “Structural analysis of network traffic flows,” in Proceedings of the joint international conference on Measurement and modeling of computer systems, ser. SIGMETRICS ’04/Performance ’04. New York, NY, USA: ACM, 2004, p. 6172

[7] P. Barford, J. Kline, D. Plonka, and A. Ron, “A signal analysis of network traffic anomalies,” in Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment, ser. IMW ’02. New York, NY, USA: ACM, 2002, p. 7182

[8] F. Silveira and C. Diot, “URCA: pulling out anomalies by their root causes,” in 2010 Proceedings IEEE INFOCOM, 2010, pp. 1–9

[9] H. Javitz and A. Valdes, “The SRI IDES statistical anomaly detector,” in 1991 IEEE Computer Society Symposium on Research in Security and Privacy, 1991. Proceedings, 1991, pp. 316–326

[10] W. Lee and S. J. Stolfo, “Data mining approaches for intrusion detection,” in Proceedings of the 7th conference on USENIX Security Symposium - Volume 7, ser. SSYM’98. Berkeley, CA, USA: USENIX Association, 1998, p. 66

[11] S. Hofmeyr, S. Forrest, and A. Somayaji, “Intrusion detection using sequences of system calls,” Journal of computer security, vol. 6, no. 3, pp. 151–180, 1998

[12] S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff, “A sense of self for unix processes,” in 1996 IEEE Symposium on Security and Privacy, 1996. Proceedings, 1996, pp. 120–128

[13] A. Frossi, F. Maggi, G. L. Rizzo, and S. Zanero, “Selecting and improving system call models for anomaly detection,” in Detection of Intrusions and Malware, and Vulnerability Assessment, ser. Lecture Notes in Computer Science, U. Flegel and D. Bruschi, Eds. Springer Berlin Heidelberg, Jan. 2009, no. 5587, pp. 206–223

[14] W. Masri and A. Podgurski, “Application-based anomaly intrusion detection with dynamic information flow analysis,” Computers & Security, vol. 27, no. 56, pp. 176–187, Oct. 2008

[15] L. Feng, X. Guan, S. Guo, Y. Gao, and P. Liu, “Predicting the intrusion intentions by observing system call sequences,” Computers & Security, vol. 23, no. 3, pp. 241–252, May 2004

[16] S. Bhatkar, A. Chaturvedi, and R. Sekar, “Dataflow anomaly detection,” in 2006 IEEE Symposium on Security and Privacy, 2006, pp. 15 pp.–62

[17] F. Qin, C. Wang, Z. Li, H. Kim, Y. Zhou, and Y. Wu, “LIFT: a low-overhead practical information flow tracking system for detecting security attacks,” in 39th Annual IEEE/ACM International Symposium on Microarchitecture, 2006. MICRO-39, 2006, pp. 135–148

[18] C. Kruegel, G. Vigna, and W. Robertson, “A multi-model approach to the detection of web-based attacks,” Computer Networks, vol. 48, no. 5, pp. 717–738, Aug. 2005

[19] “AspectJ aspectj homepage,” http://www.eclipse.org/aspectj, retrieved 2013-04-11. [Online]. Available: http://www.eclipse.org/aspectj

[20] “Load Time Weaving developer guide aspectj,” http://eclipse.org/aspectj/doc/released/devguide/ltw.html, retrieved 2013-04-11. [Online]. Available: http://eclipse.org/aspectj/doc/released/devguide/ltw.html

[21] “OpenCms, opencms homepage,” http://www.opencms.org, retrieved 2013-04-11. [Online]. Available: http://www.opencms.org

[22] “OpenVZ openvz linux containers,” http://openvz.org, retrieved 2013-04-11. [Online]. Available: http://openvz.org

[23] “Apache Tomcat apache tomcat,” http://tomcat.apache.org, retrieved 2013-04-11. [Online]. Available: http://tomcat.apache.org

[24] “Owasp Top Ten,” https://www.owasp.org/index.php/Category: OWASP_Top_Ten_Project, retrieved 2013-04-11. [Online]. Available: https://www.owasp.org/index.php/Category: OWASP_Top_Ten_Project

 ISBN: 978-1-61208-298-1

Incremental Certification of Cloud Services

Maria Krotsiani, George Spanoudakis, Khaled Mahbub
School of Informatics
City University London, UK
{Maria.Krotsiani.1, G.E.Spanoudakis, K.Mahbub}@city.ac.uk

***Abstract*—Cloud is becoming fast a critical infrastructure. However, several recent incidents regarding the security of cloud services clearly demonstrate that security rightly remains one of the major concerns of enterprises and the general public regarding the use of the cloud. Despite advancements of research related to cloud security, we are still not in a position to provide a systematic assessment of cloud security based on real operational evidence. As a step towards addressing this problem, in this paper, we propose a novel approach for certifying the security of cloud services. Our approach is based on the incremental certification of security properties for different types of cloud services, including IaaS, PaaS and SaaS services, based on operational evidence from the provision of such services gathered through continuous monitoring. An initial implementation of this approach is presented.**

Keywords–cloud services; security certification; continuous monitoring

I. INTRODUCTION

Cloud technology offers the opportunity for utilising computational capabilities offered as computation, data, storage, and network cloud services upon demand without owning the resources that realise and offer them [26]. The use of cloud services is spreading fast, formulating a market that is expected to reach a value of $66bn by 2016 [29]. Despite their fast spread, the use of cloud services has been associated with several security problems. These include breaches of integrity, confidentiality and privacy of customer data on clouds [6][9][18], spamming, wrapping and cross-site scripting attacks [6], various forms of Denial-of-Service (DoS) attacks resulting in reduced application and data availability [9][24], and Authentication, Authorization and Accounting (AAA) vulnerabilities [9][18]. Thus, the use of the cloud has created significant concerns regarding the security of the data and services offered through it [6][9][18]. Frequent reports of cloud security incidents spanning across major providers indicate that these are valid concerns [14].

The provision and assessment of cloud service security is difficult and not well supported by the existing state of the art. Researchers have made significant progress delivering methods and tools for access control and identity management on clouds [1], secure storage protocols (e.g., proof-of-retrievability [3], proof-of-storage protocols [31]), encryption and key management [17], and secure virtualization [12]. Despite this work, the security of cloud services is far from perfect. This is due to several vulnerabilities of cloud service provision that are related to the possibility of breaches of integrity, confidentiality and privacy due to multi-tenancy of services; interference between security mechanisms operating at different layers of cloud services (infrastructure, platform and software services); interference between security and cloud virtualization/optimisation mechanisms (e.g., spreading of DoS attacks due to load balancing in cloud federations [24]); and subverting administrative functions of the cloud infrastructure.

The risk arising from these vulnerabilities is increased by dependencies between services at all the layers of the cloud stack (i.e., IaaS, PaaS and SaaS services), which may also evolve dynamically (e.g., changing VMs, configurations of platform services, or deployed software services). These dependencies and their dynamic changes make it difficult to introduce appropriate pre-deployment and operation controls for assessing and guaranteeing the security [9][24]. Furthermore, the exact provision of cloud services is frequently opaque, making it difficult to assess cloud security through audit mechanisms.

Under these circumstances, the security and trustworthiness of cloud services require continuous and transparent assessment. Certification has a long history as a mechanism for verifying properties of software systems and increasing trust in them, mainly due to the liability that it entails for the authority that issues the certificates. However, existing certification methods, such as those based on the Common Criteria model [10], focus mainly on systems with a stable structure that operate under stable operational conditions and, therefore, they are not suitable for the cloud. Recent research focuses on certification of service-based systems, whose structure can change dynamically [22]. However, it still ignores the deployment infrastructure and platform services that are involved in the provision of software services in a cloud. Also, SOA certification research [22][23] is centered on certificates produced through pre-deployment testing and/or formal analysis of services without incorporating real and continuous cloud service monitoring data.

In this paper, we propose a novel cloud service certification approach that addresses this gap. Our approach is based on the creation of incremental certificates for security properties of cloud services. In such certificates, the evidence required for assessing and verifying security properties is acquired through continuous monitoring of the operation of cloud services. Hence, the evidential basis for the assessment of properties can cover contextual conditions that might not be possible to envisage, test or simulate through other forms of assessment (e.g., testing and static

 ISBN: 978-1-61208-298-1

analysis) before deploying a cloud service. Continuous monitoring can capture contextual conditions in cloud service provision as, for example, changes in the population of co-tenant services, the deployed virtualization and optimisation strategies and mechanisms, and network and middleware configurations in a cloud, which are difficult to take into account in static forms of assessment. It can also capture and adapt to the migration of SaaS cloud services within cloud federations, providing support for the adaptation of monitoring infrastructures when this happens.

The rest of this paper is organized as follows. Section II overviews our approach. Section III introduces the certification models that drive the incremental certification process. Section IV details the certification process, gives examples of how it is carried out, and overviews the tool support for our approach. Section V overviews related work, and finally, Section VI provides concluding remarks and directions for future work.

II. OVERVIEW OF INCREMENTAL CERTIFICATION

A. Certificates and certification models

The core model of certificates that can be issued in our approach is shown in Figure 1. As the model shows, a certificate asserts a *security property* regarding an *asset* of a cloud service (e.g., a specific service operation, a set of service operations, data managed by the service). Certificates are produced based on a *certification model* that is defined (or endorsed) by a *certification authority*, and are signed by this authority. The certification model is based on an *assessment scheme* determining what evidence should be taken into account for assessing a property. The assessment scheme also determines how frequently the evidence should be checked and when the accumulated evidence will be sufficient for issuing the certificate through an *evidence aggregation scheme*.

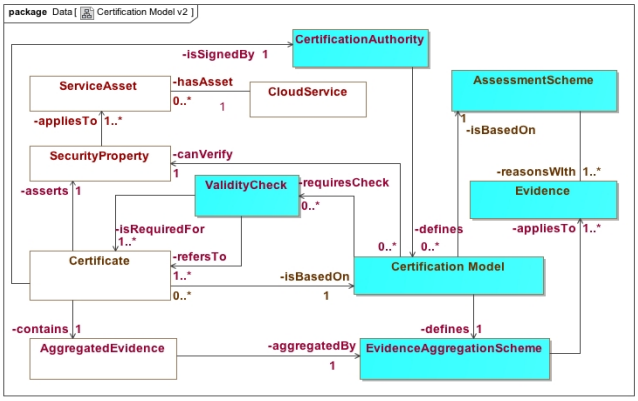


Figure 1: Model Based Certificates

The certification model defines also *validity checks* that should be executed before issuing a certificate. Such checks may, for example, require that the monitoring infrastructure, which is used to gather the operational evidence for the certificate C, has itself a certificate C' confirming the integrity and non-repudiation of the monitoring data that it captures and provides. Also, C' had to be valid for the entire period over which the particular monitoring infrastructure has been used to gather evidence for C.

When the evidence gathered for a certificate type is sufficient for verifying the security property related to it (as determined by the certification model), a certificate that is an instance of this type could be issued. However, even after they are issued, certificates can be subject to changes in the operational conditions of the cloud service that they are associated with. The possible updates and other key changes in the life cycle of incremental certificates are described by a generic life cycle model that we discuss next.

B. Lifecycle of incremental certificates

The life cycle of incremental certificates is described by the UML statechart diagram of Figure 2. The first state in the certificate's lifecycle is *Activated*. In this state, a certificate type is activated with a unique ID; a reference to the certification model that will be used for its incremental assessment and for issuing and updating certificates that are instances of it; a reference to the asset(s) of the cloud service that it refers to; and a digital signature of its issuer. The signature confirms that certificates, which are instances of the particular type, may be potentially issued for the referenced cloud service, based on the specific certification model and the evidence that needs to be collected.

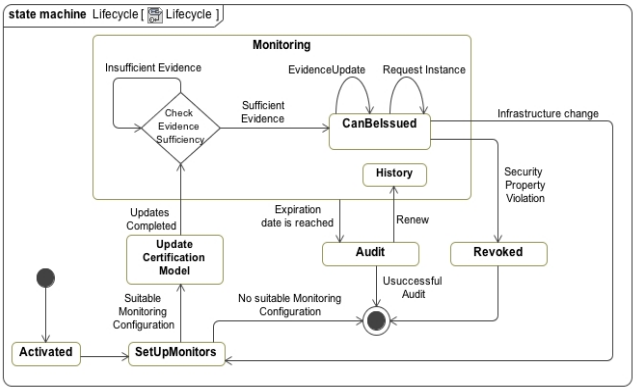


Figure 2: Life cycle of incremental certificates

After being activated, a certificate type enters the *SetUpMonitors* state. At this state, a monitoring infrastructure that will be used to acquire the operational evidence required for the assessment of the certificate is set up. If a suitable monitoring infrastructure for the type can be identified and set up, the certificate type moves to *UpdateCertificationModel* state. At this state, a concrete operational specification of the security property of the certificate type is generated for the particular infrastructure. This specification will enable the monitoring infrastructure to determine the monitoring events (evidence) required for the assessment of the property and how to use them in order to assess the property. When these updates are completed and recorded in the certification model, the certificate type moves to *Monitoring* state. Note, however, that if no suitable monitoring configuration is found, the certificate type will cease to exist.

Whilst at the monitoring state, the evidence required for the assessment of the certificate type is continually gathered

 ISBN: 978-1-61208-298-1

by the monitoring infrastructure. When the accumulated evidence becomes sufficient for confirming the validity of the security property of the type, the certificate type gets to the sub state *CanBeIssued*. This state indicates that certificates of the particular type can be issued by the certification platform. Whilst a certificate type is at the state *CanBeIssued*, its monitoring continues and, at specific checkpoints defined by the certification model, any additional operational evidence that is acquired for the type by the monitoring infrastructure is recorded in an aggregated format within the certification infrastructure (see the self-transition *EvidenceUpdate*).

When a certificate type gets to the state *CanBeIssued*, any agent interested in it (whether an application or a human actor) can request the generation of a certificate that is an instance of the particular type from the certification infrastructure by using the certificate type's unique ID. Upon such a request, the certification infrastructure will check if the extra validity conditions for the certificate type (if any) are satisfied and, if they are, it will generate a certificate instance and return it back to the requester. The generated instance will be identified by the combination of the ID of the certificate type and its own unique ID. The certificate instance will also record the expiration date of its type (as this date will apply to it as well). These actions are executed during the transition *RequestInstance* in Figure 2. The issued instance of the certificate type will incorporate the aggregated evidence until the last checkpoint when monitoring data were retrieved from the monitoring infrastructure and aggregated for the certificate type. It will also contain the timestamp of the earliest and latest available evidence for the security property that it asserts, in order for its users to know the exact period covered by the evidence.

A certificate type that is at the state *CanBeIssued* may also be revoked. This will happen if the monitoring infrastructure identifies new monitoring data contradicting the evidence gathered so far and indicating that the relevant security property has been compromised. The occurrence of such evidence is signified by the transition *Security Property Violation* to the state *Revoked*. When a certificate type gets to the state *Revoked*, all the previously generated instances of it will be revoked. Also, the certificate type will be flagged as revoked in the certification infrastructure in order to prevent the generation of new instances of it, and its monitoring will be terminated.

In line with traditional models of certification, a certificate type has an expiration date. When this date is reached, the certificate type will move to the state *Audit*. This state signifies an audit of the certification model and the evidence gathered for the certificate type. The certification authority, which defined the certification model, will carry this out in order to ascertain that it may continue to use the certification model for producing certificates. If the audit is successful, the certificate type can be renewed: the transition *Renewed* brings the certificate back to the state where it was prior to reaching its expiration date and moving to the state *Audit*. If the audit is unsuccessful, the specific certificate type will be terminated. In this case, a notification should also be sent to owners of any issued instance certificates of the type to notify them that the type no longer exists but that the instances of it that have been issued will remain valid until their expiration date.

Following the expiration of a certificate instance, its agent can request from the certification infrastructure to provide a renewed instance of the same certificate type. The infrastructure will be able to do so only if the certificate type at this stage is at the state *CanBeIssued*.

The life cycle mode also reflects reactions to changes related to the cloud infrastructure where the certified service has been deployed, or to the monitoring infrastructure that is used to generate the operational evidence for the certificate. Such changes may require a change in the monitoring infrastructure that is used to gather the operational evidence underpinning the certificates of the given type and the concrete operational specification of the security property of the certificate type that drives the operation of the monitoring infrastructure. The evidence, however, that has been held so far in the certification infrastructure regarding the property may (depending on the certification model) remain relevant and should be maintained by the certification structure so that to be used when issuing new instances of the given type.

Changes in the cloud infrastructure and/or the monitoring infrastructure will trigger the transition of the certificate type to the *SetUpMonitors* state to check whether after the changes, the cloud deployment infrastructure of the service provides the monitoring capabilities required for continuing the monitoring of the specific certificate type. If successful, the certificate will move to the *UpdateCertificationModel* during which the certification model will be updated as described previously. When this completes successfully, the certificate will transit back to the *evidence* accumulation state inside *Monitoring*. If, however, the security property cannot be monitored after the changes, the certificate type will cease to exist.

C. Incremental Certification Infrastructure

The generation of incremental certificates is supported by a proof-of-concept certification infrastructure (CeRTiN). The architecture of CeRTiN is shown in Figure 3.

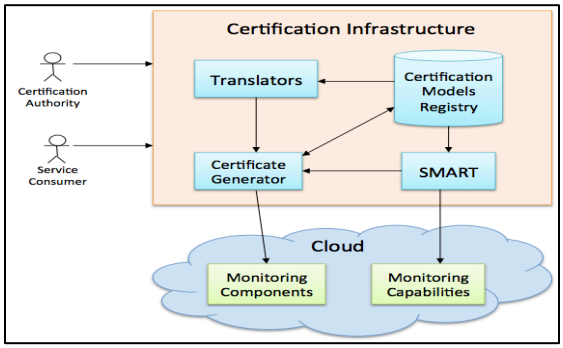


Figure 3: Certification Infrastructure

CeRTiN consists of a *Certificate Generator (CG)*, a *service monitorability reporting tool (SMART)*, a set of *translators*, and a *certification models registry*. The certificate generator has the responsibility for initiating and

managing the process of creating incremental certificates following the life cycle model introduced in Sect. II.B, upon requests for certification authorities. The operation of the certificate generator is driven by certification models, maintained as part of a model registry. To realise the functionality underpinning the state *SetUpMonitors* and *Monitoring* in the life cycle model, the generator interacts with *SMaRT* and *Translators*, and *Monitoring components* on cloud platforms, respectively. *SMaRT* has responsibility for checking (i) whether the cloud on which the cloud service to be certified is deployed has the monitors required for the incremental certification of the service, and (ii) for assembling an appropriate monitoring platform. To do so, *SMaRT* has access to the certification model to be applied and descriptions of the monitoring capabilities available in cloud platforms. *Translators* have the responsibility for translating the security property to be verified by certificates into an operational monitoring specification for the monitoring platform assembled by SMaRT. Finally, the monitoring components on cloud platforms have the responsibility for providing the raw monitoring evidence from cloud monitoring components.

III. SPECIFICATION OF CERTIFICATION MODELS

As discussed in Sect. II, the generation of incremental certificates is driven by certification models. In this section, we define the language for specifying such models as an XML schema. The top-level structure of this schema is shown in Figure 4 and the elements at this level of the schema are discussed below.

A. Certification model elements

CASignature: The element *CASignature* represents the digital signature of the certification authority that has defined/advocated the certification model.

AbstractSecurityProperty: This element defines the security property that is to be certified by the particular certification model. The security property must be defined in an abstract form that is independent from the language used by the monitoring infrastructure that will be used for gathering operational evidence for the property. This is because the monitoring infrastructure that will be used for different certificates of the model may vary across different cloud platforms. Also, the monitoring infrastructure may change during the lifecycle of the certificate type when, for example, the service or the constituent services and components that are the subject to incremental certification migrate across different clouds (e.g., within a cloud federation). During the monitoring infrastructure set up state in the certificate type life cycle model, the abstract property is translated automatically to the concrete property that will then be used to drive the operational monitoring (see below).

In the current implementation of our approach, the abstract security property is expressed using a subset of the language for specifying Service Level Agreements (SLAs) that was introduced by the SLA@SOI project, known as SLA* model [19]. In particular, we are using the part of the language that enables the definition of guaranteed terms in an SLA. SLA* has been chosen for two reasons. The first is that it defines several build-in security properties such as integrity, non-repudiation, availability and forms of confidentiality as "standard" guarantee terms. The second reason is that it provides the syntactic and semantic means for customizing the definitions of built-in properties and/or defining new properties.

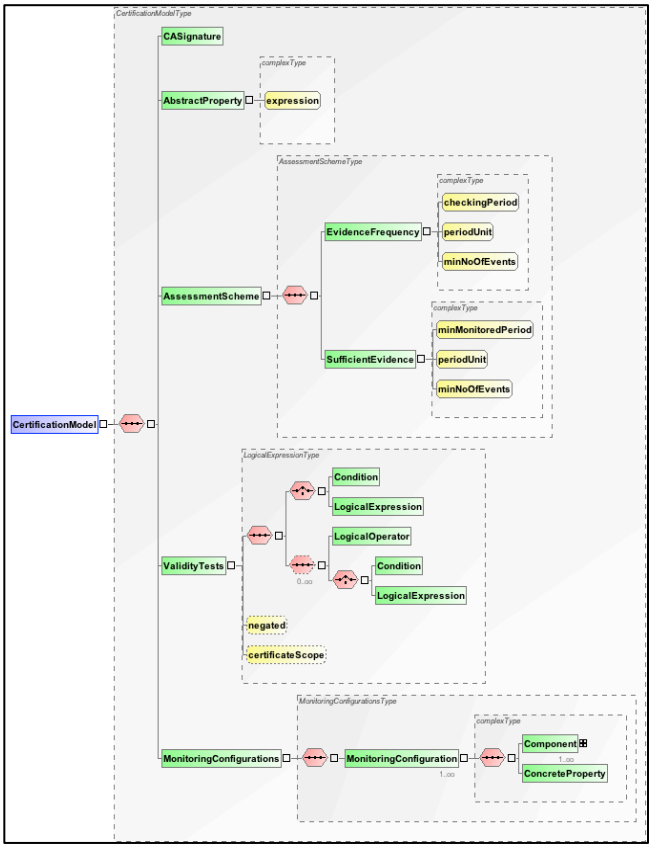


Figure 4: Certification model schema (top level)

The certification model schema provides two options for specifying abstract security properties in SLA*: (1) to specify the property as a string, according to the BNF syntax of SLA*, or (2) to use the XML schema defined for SLA*. A full account of SLA* is beyond the scope of this paper and can be found in [19]. In Figure 5, however, we show an example of specifying the availability of a service as defined in SLA*.

AssessmentScheme: This element defines conditions regarding the assessment of the evidence that should be satisfied, in addition to the adherence of the cloud service to the security property, for the certificate to become issuable. These conditions relate to the frequency and sufficiency of evidence collection, and are specified by the following sub-elements of an assessment scheme:

- *EvidenceFrequency* – This element defines the *checkingPeriod*, which states how often the events should be checked, and/or the *minNoOfEvents*, which declares the minimum number of events that should occur in a specific period of time.

 ISBN: 978-1-61208-298-1

- *SufficientEvidence* – This element defines sufficiency conditions regarding the extent of collected evidence for issuing a certificate. Such conditions are specified by the sub-elements of *SufficientEvidence*, namely:
  - *minMonitoredPeriod* that states the minimum time needed for monitoring,
  - *minNoOfEvents* that states the minimum number of events (evidence) that need to be monitored.

Validity tests: A certification model may, in addition to the assessment scheme, define extra validity tests as preconditions for issuing a certificate of a given type. These tests may relate, for example, to conditions regarding the cloud where the service is deployed (e.g., requiring that the cloud offers full isolation of virtual machines) or the adherence of other services that this service may depend on to standards (e.g., requiring that a storage service, which is used by a SaaS service implements correctly a proof-of-retrievability protocol [3]) or the monitoring infrastructure itself (e.g., requiring the integrity of the transmission of monitoring events and results inside the infrastructure and to external clients of it). Such conditions are specified by the element *validityTests* in the certification model schema. Our approach to the specification of validity tests assumes that any related components will have other certificates confirming their adherence to the required conditions and therefore the validity tests are expressed as logical conditions against the contents of such certificates.

MonitoringConfigurations: This element specifies the list of the monitoring configurations that have been used to collect the evidence for generating certificates. Each monitoring configuration includes:

- A list of *components* of the monitoring environment. These components can be of two types: (1) *sensors,* which are components capable of capturing and transmitting primitive monitoring events, and (2) *reasoners,* which are components capable of analyzing events and checking whether monitoring conditions are satisfied (aka *monitors*). The *start* and *end* of each time period during which it has been used for collecting evidence for the specific certification model
- *ConcreteSecurityProperty* – This element provides the concrete operational specification of the security property that is to be certified by the model, expressed in the language accepted by the reasoner(s) of the particular monitoring configuration. The concrete security property is generated automatically from the abstract security property once a monitoring configuration is selected as we discuss in Sect. IV.C.

B. Example of Certification Model for Availability Property

Figure 5 presents an example of a certification model for the availability of a service, i.e., the ratio of the period of time during which a service cannot respond to calls of its operations without producing an error (i.e., it is unavailable), over the total period of time during which the service is deployed.

As shown in the figure, the value of the *expression* attribute in the *AbstractSecurityElement* is set to *availability* (i.e., a standard term of SLA* with the meaning defined above). The element *AssessmentScheme* of the model specifies that monitoring should be performed every 24 hours for minimum of 1000 events in this period (see the attribute values of *EvidenceFrequency* element) and monitoring should be performed for at least 30 days for minimum 100000 events in this period in order to issue a certificate (see the attribute values of *SufficientEvidence* element). The *ValidityTest* element specifies a precondition that should be met to issue a certificate. The precondition in this example specifies that the *integrity* of the transmission of monitoring events and results should be maintained. Finally, in the *MonitoringConfiguration* element, it is specified the monitoring environment that will be used to monitor the events and collect the evidence.

```
<ns1:CertificationModel  xmlns:xsi='http://www.w3.org/2001/XMLSchema-
instance'
  xmlns:ns3='http://slasoi.org/monitoring/citymonitor/xmlrule'
  xmlns:ns2='http://assert4soa.eu/schema/Assert_SQL'
  xmlns:ns1='http://www.cumulus.org/certificate/model'
  xsi:schemaLocation='http://www.cumulus.org/certificate/model
CertificationModel-v2.xsd'>
   <CASignature></CASignature>
   <AbstractSecurityProperty
      expression="http://www.slaatsoi.org/commonTerms#availability"/>
   <AssessmentScheme>
      <EvidenceFrequency checkingPeriod="24" periodUnit="hours"
        minNoOfEvents ="1000"></EvidenceFrequency>
      <SufficientEvidence minMonitoredPeriod="30" periodUnit="days"
           minNoOfEvents="100000"></SufficientEvidence>
   </AssessmentScheme>
   <ValidityTests negated="false" certificateScope="SINGLE">
      <ns2:Condition negated="false" relation="EQUAL-TO">
         <ns2:Operand1>
            <ns2:AssertOperand facetName="Assert" facetType="Assert">
              //ASSERTCore/SecurityProperty/@PropertyAbstractCategory
            </ns2:AssertOperand>
         </ns2:Operand1>
         <ns2:Operand2>
            <ns2:Constant type="STRING">
              http://www.assert4soa.eu/ontology/security/security#Integrity
            </ns2:Constant>
         </ns2:Operand2>
      </ns2:Condition>
   </ValidityTests>
   <MonitoringConfigurations>
      <MonitoringConfiguration>
         <Component type="REASONER">
            <EndPoint>http://localhost:8888/...</EndPoint>
         </Component>
         … … … …
      </MonitoringConfiguration>
   </MonitoringConfigurations>
</ns1:CertificationModel>
```

Figure 5: Example Certification Model

IV. CERTIFICATION PROCESS

In the following, we give an example demonstrating the certification process based on the availability property specified in the certification model of Figure 5.

A. Initiation of certification process

The certification process starts when a service provider makes a request to a Certification Authority (CA), in order to certify a security property for a service. The CA may use an existing Certification Model (CM) if it is suitable for the property and the service, create a new model for it, or reject the request. If a CM is identified, CA submits it to the Certificate Generator CG). CG has then to select and

configure a monitoring infrastructure for starting the incremental certification process.

B. Selecting and configuring the infrastructure

Assuming that the request for certification is about the availability of a service S, the CM of Figure 5 could be selected and used for the certification of S. After identifying it, CG calls SMART (see Figure 3) to find if there is a monitoring infrastructure available in the cloud platform where the service is deployed that could monitor availability.

To do so, SMART parses the security property in the certificate model and generates an Abstract Syntax Tree (AST) of the property based on the BNF grammar of the security property language. Then, it searches through the registry of the monitoring capabilities of the cloud infrastructure to check if there are suitable sensors for providing the events and analysis functions required for monitoring the availability property.

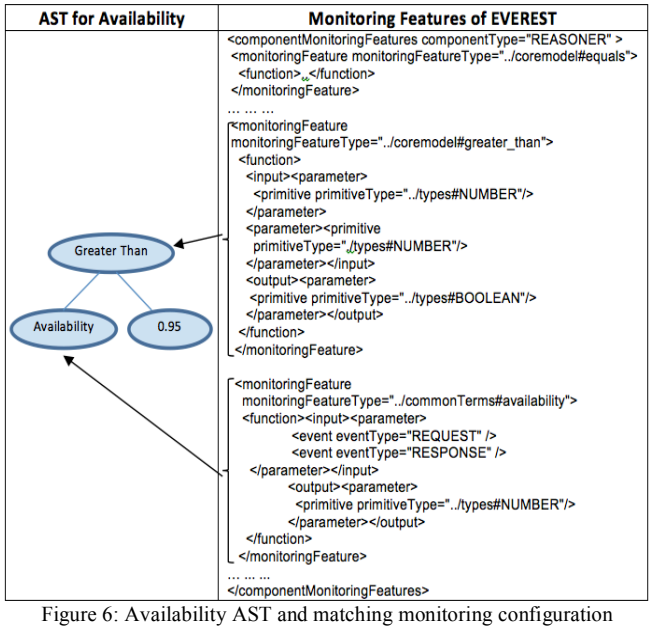


Figure 6: Availability AST and matching monitoring configuration

The capabilities of the monitoring components that may be available in the infrastructure are described according to the monitoring capability model introduced in [16]. According to this model, a monitoring component in a cloud infrastructure is described by its unique identifier within the infrastructure (uuid), its type (i.e., SENSOR or REASONER depending on whether the component can capture and transmit events or can analyse them to check if a monitoring condition is satisfied, respectively), and a list of *MonitoringFeatures*. The latter describe either *basic features* such as event capturing and transmitting operations (e.g., service operation requests and responses) or more complex computational *functions* (e.g., computation of average time between service calls, CPU load etc.).

SMART uses the AST for the property to find monitoring components, which have monitoring features matching each node of the tree. More specifically, a non-leaf node of AST would have the form *(Parent: Operator (OP), Children: left-hand side (LHS), right-hand side (RHS))* and each of OP, LHS and RHS needs to be matched with a monitoring component with appropriate features. A full match of the AST with monitors constitutes a candidate configuration. For these, SMART also checks if they satisfy any validity conditions for monitors that may have been defined in the certification model, and maintains only the configurations that are compliant. If there is more than one such configuration, it also performs a selection. Figure 6 shows the AST for availability and an example of a matching monitoring configuration.

C. Deriving the monitoring specification

Following, the selection of a monitoring configuration, CG translates the abstract security property of the certification model into the monitoring language accepted by the REASONER component in the configuration.

In the current implementation of CeRTiN, we are using EVEREST (EVEnt REaSoning Toolkit [15]) to perform the monitoring required during the certification process. EVEREST is an open-source monitoring framework developed by the last two authors of this paper to support the monitoring of service-based systems. EVEREST supports the monitoring of properties expressed in *EC-Assertion*, a first order temporal logic language based on Event Calculus.

EC-Assertion specifies monitorable properties in terms of *events* and *fluents*. An event is something that occurs at a specific instance of time and has instantaneous duration. Fluent represent system states and are initiated and terminated by events. The basic predicates used by EC-Assertion for expressing events and fluents, and their meanings are summarized in Table I.

TABLE I: EC-ASSERTION PREDICATES

Predicate	Meaning
Happens(e,t∈ℜ(t^L,t^U))	An event *e* of instantaneous durations occurs at some time point *t* within the time range *ℜ(t^L,t^U)* (ℜ(t^L,t^U)=[t^L,t^U]).
HoldsAt(f,t)	A state (aka *fluent*) *f* holds at time *t*. This is a derived predicate that is true if the *f* has been initiated by some event at some time point t' before t and has not been terminated by any other event within the range [t',t].
Initiates(e,f,t)	Fluent *f* is initiated by an event *e* at time *t*
Terminates(e,f,t)	Fluent *f* is initiated by an event *e* at time *t*
Initially(f)	Fluent *f* holds at the start of system operation.
<rel>(x,y), <rel>::= = | < | > | ≤ | ≥ | ≠	The relation <rel> holds between the x and y.

Based on these predicates, EC-Assertion expresses monitorable properties as *monitoring rules* of the form *body ⇒ head*. The meaning of a monitoring rule is that if its body evaluates to *True*, its head must also evaluate to *True*. EC-Assertion also uses *monitoring assumptions*, which have the same form as rules but their meaning is that when their body evaluates to *True*, their head can be deduced. Thus assumptions are used to deduce and/or record information about the state of the system during monitoring.

Operational monitoring specifications in EC-Assertion are produced from the specification of an abstract security property in a certification model by transforming the AST of the property into EC-Assertion formulas. The translation process is based on the use of predefined parametric

 ISBN: 978-1-61208-298-1

monitoring templates for the standard terms of the SLA* model. These templates are retrieved and instantiated during the translation process when the relevant standard term of SLA* is encountered in a node of the AST generated for a security property. The details of this translation process are beyond the scope of this paper and can be found in [20].

Table II shows the parametric template used for the standard *Availability* property in SLA*. As discussed in Sect. III.B, service availability is defined as the ratio of the period during which a service is unavailable over the total period of monitoring a service.

TABLE II: AVAILABILITY TEMPLATE (EC-ASSERTION)

⟨Availability⟩$_{def}$==

A0.Availability.<CaseId>: Initially(LastServiceMonitoringPeriod(<_SrvId>, systemTime())

A1.Availability.<CaseId>:
Initially(UnavailablePeriods(<_SrvId> ,_PN, _P[]))

A2.Availability.<CaseId>:
Happens(e(_id1, _Snd, <_SrvId>, Call(_O), <_SrvId>), t1, [t1,t1]) ∧
¬**Happens**(e(_id2, <_SrvId>, _Snd, Response(_O), <_SrvId>), t2, [t1,t1+<D>]) ∧ ¬∃ _PN, _ST,: **HoldsAt**(Unavailable(_PN, <_SrvId>, _ST), t1)) ∧
∃ _PN, _P[]: **HoldsAt**(UnavailablePeriods(<_SrvId>, _PN, _P[]), t1)) ⇒
Initiates(e(_id1, _Snd, <_SrvId>, Call(_O), <_SrvId>), Unavailable(_PN+1, <_SrvId>, t1), t1) ∧ **Terminates(**e(_id1, _Snd, <_SrvId>, Call(_O), <_SrvId>), UnavailablePeriods(<_SrvId>, _PN, _P[]), t1) ∧
Initiates(e(_id1,_Snd,<_SrvId>,Call(_O),<_SrvId>),UnavailablePeriods(<_SrvId>,_PN+1,_P[]), t1)

A3.Availability.<CaseId>:
Happens(e(_id1, _Snd, <_SrvId>, Call(_O), <_SrvId>), t1, [t1,t1]) ∧
Happens(e(_id2, <_SrvId>, _Snd, Response(_O), <_SrvId>), t2, [t1,t1+<D>]) ∧∃ _PNum, _ST: **HoldsAt**(Unavailable(_PN, <_SrvId>, _ST), t1) ⇒
Terminates(e(_id1, _Snd, <_SrvId>, Call(_O), <_SrvId>), Unavailable(_PN, <_SrvId>, _ST), t1+1)

A4.Availability.<CaseId>:
Happens(e(_id1, _Snd, <_SrvId>, Call(_O), <_SrvId>), t1, [t1,t1]) ∧
Happens(e(_id2, <_SrvId>, _Snd, Response(_O), <_SrvId>), t2, [t1,t1+<D>]) ∧ ∃ _PN, _ST,: **HoldsAt**(Unavailable(_PN, <_SrvId>, _ST), t1)) ∧
∃ _PN, _P[]: **HoldsAt**(UnavailablePeriods(<_SrvId>, _PN, _P[]), t2)) ⇒
Terminates(e(_id1, _Snd, <_SrvId>, Call(_O), <_SrvId>), UnavailablePeriods(<_SrvId>, _PN, _P[]), t2) ∧
Initiates(e(_id1, _Snd, <_SrvId>, Call(_O), <_SrvId>), UnavailablePeriods(<_SrvId>, _PN, append(_P[], t1 – ST)), t2)

R.Availability.<CaseId>:
Happens(e(_id1, _Snd, <_SrvId>, Call(_O), <_SrvId>), t1, [t1,t1]) ∧
Happens(e(_id2, <_SrvId>, _Snd, Response(_O), <_SrvId>), t2, [t1,t1+<D>]) ∧∃ _PN, _ST, _P []: **HoldsAt**(Unavailable(_PN, <_SrvId>, _ST), t1)) ∧
HoldsAt(UnavailablePeriods(<_SrvId>, _PN, _P[]), t2) ∧
HoldsAt(LastServiceMonitoringPeriod(<_SrvId>, _lmsTime), t2)) ⇒
sum(_P[]) / (t2 – _lmsTime) > K

The parametric availability template uses three fluents specific to availability computation. The fluent Unavailable keeps track of unavailability. The fluent has three parameters: (i) _PN that counts the number of unavailable periods for a monitored service; (ii) _SrvId that records the unique ID of the monitored service, and (iii) _ST that records the time point when the service becomes unavailable. To keep track of unavailable periods the template uses the fluent UnavailablePeriods. This fluent has also three parameters: (i) _SrvId that records the unique ID of the monitored service; (ii) _PN that records the count of unavailable periods of the monitored service; and (iii) _P[] that records duration of each unavailable period of the monitored service. The third fluent in the template, i.e., LastMonitoringPeriod, is used to record the starting time point of the monitoring session. This fluent has two parameters. The first parameter (i.e. <_SrvId>) records the unique ID of the monitored service, and the second parameter (i.e. systemTime()) signifies a system call executed by the monitor to obtain the current time of the system where the monitoring service is running.

The assumptions A0 and A1 initiate the *LastMonitoringPeriod* and *UnavailablePeriods* fluents respectively for first time. A2 starts new period of unavailability when a non-served event occurs and increases the number of unavailability periods. A3 terminates a current period of unavailability for a service. The assumption A4 records the length of a terminated period of unavailability. Finally, the rule *R* checks if the availability of a service is greater than *K*.

In the template, *<CaseId>* refers to the unique id of the certificate to be generated. It should be noted that EVEREST would receive primitive call and response events from the sensor that has been selected by SMaRT. Therefore, a call to the monitored service occurred at *t* is considered as served if a corresponding response occurs within a predefined time range between t and t+d. The value of *d* is denoted as *<D>* in the templates. During the translation process concrete values of *<_SrvId>, <CaseId>* and *<D>* are chosen according to a predefined set of criteria.

D. Monitoring process & reaction to changes

During the monitoring state in the lifecycle model of Figure 2, EVEREST analyses the events received from sensors and detects violations of the monitoring rule that provides the concrete operational specification of the security property of the certification model. EVEREST attempts to match events sent to it from sensors with the monitoring rules and assumptions. When the *body* of a monitoring rule is fully matched (instantiated) with events, it will expect to receive events matching the rule's *head* or find them in its internal event database according to the designated time constraints. If such events are not found/received the rule is treated as violated. Otherwise, it is treated as satisfied. The same process is used for assumptions, except that when the *body* of an assumption becomes fully instantiated (i.e., True), the predicates in its *head* are derived and recorded in the EVEREST's database. Instances of monitoring rules instantiated by the events that match them constitute the monitoring results, which are sent to CG (see [15] for a detailed description).

In the case of the availability property example, EVEREST would report to CG instances of *R* where *sum(_P[]) / (t2 – _lmsTime) > K* (property satisfaction cases) as well as cases where *sum(_P[]) / (t2 – _lmsTime)* $\leqslant K$ (property violations). *_P[]* in these results will provide all the durations of unavailability periods and the events instantiating rule R will provide the concrete evidence demonstrating satisfaction or violation of the property.

 ISBN: 978-1-61208-298-1

E. Certificate provision

When a request for a certificate regarding the availability of a the service is made to CeRTiN, if the certificate type under monitoring is in the state *CanBeIssued*, CG will extract the monitoring results from its own database and generate the certificate. Prior to generating it however, it will also check if any extra validity tests required by the certification model are satisfied.

F. Tool Support

An early proof-of-concept implementation of CeRTiN has been developed using two components of the open source monitoring framework developed by the SLA@SOI project, namely SMART and EVEREST. These two components play the roles of monitoring configuration selection tool and monitors in CeRTiN. Both of these tools have been implemented in Java based on Eclipse Modeling Framework (EMF). The tool is shown in Figure 7.

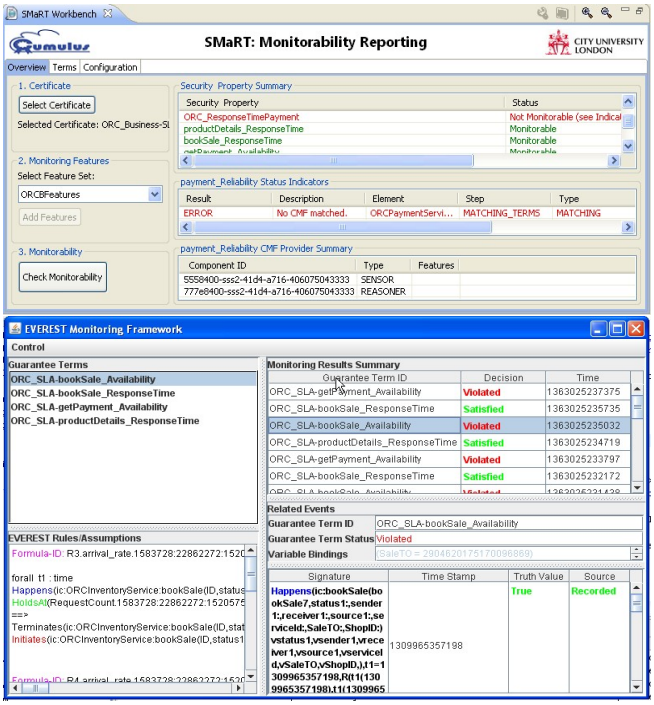


Figure 7: CeRTiN prototype

The upper part of Figure 7 illustrates SMART. As shown in the figure, the left hand panel of the tool allows the user to select the certification model to be applied for a given service and the monitoring features (i.e., capabilities) that are available in a cloud infrastructure.

The lower part of Figure 7 shows monitoring results produced by EVEREST. The left top panel of EVEREST lists the concrete security property that has been produced from the abstract security property that is to be monitored. The bottom left panel shows the monitoring specification of the abstract property (concrete security property) in EC–Assertion. The top right panel of the tool shows the summary of the monitoring results and bottom right panel shows the detail description of each monitoring result.

V. RELATED WORK

Our approach is related to research strands in three different areas, namely: software and software services certification, cloud security and cloud monitoring.

Existing approaches in the field of security certification have focused on concrete software components and provide human readable certificates. Thus, they cannot support service-based scenarios that require machine-readable certificates and could support dynamic service selection and composition [11].

Unlike traditional approaches, the FP7 Project ASSERT4SOA [22] has been focusing on formal and test-based certification of services and has developed a framework for representing and using machine-readable service security certificates, known as ASSERTS, in service discovery and composition [21][22]. But overall research on the certification of cloud services and applications is still in an early stage. Some work has been done to predict the potential benefits of integrating certification schemes within cloud infrastructures [2][18], without, however, offering a concrete solution to this problem. Grobauer et al. [2], examine potential vulnerabilities in cloud computing, and acknowledge that the existence of such vulnerabilities is the lack of certification schemes and security metrics for cloud. Heiser and Nicolett [18] evaluate the cloud security risks and propose sharing IT risks with any externally provided service. A test based cloud certification approach has been proposed in [23] focusing on minimising test generation and execution activities in certifying cloud services. Test based certificates of cloud services could be combined with monitoring based certification, so as to produce extended hybrid certificates for the properties of interest.

More work has focused on auditing cloud security. The Cloud Controls Matrix (CCM) of the Cloud Security Alliance (CSA) [5], for example, contains a comprehensive set of controls to assess the information security assurance in clouds and maps controls to existing frameworks such as PCI DSS [30], COBIT [7]. CCM is currently being developed through the Open Certification Framework (OCM) [28] into a 3rd party certification program. Moreover, CSA has published the Cloud Audit protocol [4], which provides an automated query interface to cloud services for audit. Our approach is compliant with CSA's frameworks and we are investigating the potential integration of CeRTiN into OCM.

Cloud monitoring has been supported by several monitoring systems, including commercial software, open source systems, and research prototypes. Most of these systems (e.g., [25][27]) focus on performance monitoring without checking security properties as such. Cloud security monitoring is supported by positioning agents at different key points of a cloud infrastructure to detect incidents [13][14]. Other systems (e.g., DeSVi [32]) support the detection of SLA violations. None of these systems, however, supports security monitoring directly and customers can only use them to establish performance baselines, which, if deviated substantially, could indicate some incident conditions. EVEREST [15], the monitoring

 ISBN: 978-1-61208-298-1

framework that underpins CeRTiN, is in exception as it provides direct support for security monitoring.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have introduced a certification approach for the security of cloud services that is based on incremental assessment of security properties based on continuous monitoring, and have described the basic models and mechanisms for realising it.

The use of evidence coming from continuous monitoring provides coverage of contextual conditions that might not be possible to envisage, test or simulate through other forms of assessment of security properties (e.g., testing and static analysis) before deploying a cloud service. Therefore, our approach provides an advantage over test and static analysis approaches. We should note, however, that our work is part of a broader research programme undertaken by the EU F7 project CUMULUS, which aims to develop certification schemes that will enable the development of hybrid certification schemes, incorporating multiple types of evidence, including testing, static analysis and monitoring. Hence, we also aim to investigate how to integrate the certification models described in this paper with test and static analysis based models.

Furthermore, we are looking into the further development of some elements of our approach, notably the development of mechanisms for filtering low level monitoring data and aggregating them into compound forms of evidence prior to including them in certificates. This will be necessary in order to produce scalable and auditable certificates. We are also planning an evaluation activity based on inputs from certification authorities and industrial stakeholders that participate in CUMULUS.

ACKNOWLEDGMENT

This work has been partially funded by the EU F7 project CUMULUS [8] (grant no 318580).

REFERENCES

[1] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, “Security and Cloud Computing: InterCloud Identity Management Infrastructure”, In 19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, 2010, pp. 263-265.

[2] B. Grobauer, T. Walloschek, and E. Stocker, "Understanding Cloud Computing Vulnerabilities," IEEE Security & Privacy, vol. 9, 2011, pp. 50-57.

[3] C. Cachin, I. Keider, and A. Shraer, “Trusting The Cloud”, IBM Research, Zurich Research laboratory, 2009.

[4] Cloud Security Alliance, Cloud Audit, Available from: https://cloudsecurityalliance.org/research/cloudaudit/

[5] Cloud Security Alliance, Cloud Controls Matrix, Available from: https://cloudsecurityalliance.org/research/ccm/

[6] Cloud Security Alliance, Security Guidance for Critical Areas of Focus in Cloud Computing vol. 2, available from: http://www.cloudsecurityalliance.org/guidance/csaguide.v2.1.pdf [retrieved: June, 2013].

[7] COBIT, http://www.isaca.org

[8] CUMULUS project, http://www.cumulus-project.eu/

[9] D. Catteddu and G. Hogben, “Cloud Computing: Benefits, Risks and Recommendations for Information Security.”, European Network and Information Security Agency (ENISA), 2009.

[10] D. S. Herrmann, “Using the Common Criteria for IT security evaluation”, Auerbach Publications, 2002.

[11] E. Damiani, and A. Mana, “Toward WS-Certificate”, In Proc. of the ACM Workshop on Secure Web Services (SWS 2009), 2009, pp. 1-2.

[12] F Lombardi and R Di Pietro, “Secure virtualization for cloud computing”, J. Netw. Comput. Appl. 34, 4, July 2011, pp. 1113-1122.

[13] F. Doelitzscher, C. Reich, M. H. Knahl, and N. L. Clarke, "Incident detection for cloud environments", Proc. of the Third International Conference on Emerging Network Intelligence,2011, pp. 100-105.

[14] F. Doelitzscher, C. Reich, M. Knahl, A. Passfall, and N. Clarke, "An agent based business aware incident detection system for cloud environments", Journal of Cloud Computing: Advances, Systems and Applications vol. 1:9, 2012.

[15] G. Spanoudakis, C. Kloukinas, and K. Mahbub, “The SERENITY Runtime Monitoring Framework”, In Security and Dependability for Ambient Intelligence, Springer, 2009, pp. 213-238.

[16] H. Foster and G. Spanoudakis, “Advanced Service Monitoring Configurations with SLA Decomposition and Selection”, In Proc. of 26th Annual ACM Symposium on Applied Computing, 2011, pp. 1582-1589.

[17] H. Li, Y. Dai, and B. Yang, "Identity-Based Cryptography for Cloud Security", IACR Cryptology ePrint Archive, 2011, pp.169-169.

[18] J. Heiser and M. Nicolett, “Assessing the Security Risks of Cloud Computing”, Gartner technical report, June 2008.

[19] K. Kearney, F. Torreli, and C. Kotsokalis, “SLA*: An Abstract Syntax for Service Level Agreements”, In Proc. Of 10th IEEE/ACM International Conference on Grid Computing, 2010, pp. 217-224.

[20] K. Mahbub, G. Spanoudakis, and T. Tsigkritis, "Translation of SLAs into Monitoring Specifications", In Service Level Agreements for Cloud Computing, R. Yahyapour, P. Weider (Eds), Springer-verlag, 2011.

[21] L. Pino, G. Spanoudakis, “Constructing Secure Service Compositions with patterns” 2012 IEEE 8th World Congress on Services, 2012, pp. 184-191.

[22] M. Anisetti et al., “ASSERT4SOA: Toward Security Certification of Service-Oriented Applications”, OTM 2010 Woekshops, 2010, pp. 38-40.

[23] M. Anisetti, C. A. Ardagna, and E. Damiani, “A Low-Cost Security Certification Scheme for Evolving Services”, In Proc. Of IEEE 19th International Conference on Web Services, 2012, pp. 122-129.

[24] M. Jensen, J. Schwenk, N. Gruschka, and L. L. Iacono, “On Technical Security Issues in Cloud Computing”, In Proc. of the IEEE International Conference on Cloud Computing, 2009, pp. 109-116.

[25] M. L. Massie, B. N. Chun , and D. E. Culler, “The ganglia distributed monitoring system: design, implementation, and experience”, Parallel Computing, vol. 30, 2004, pp. 817–840.

[26] Microsoft, The Economics of Cloud for the EU Public Sector, Paper, 2010:http://www.microsoft.eu/Portals/0/Document/EU_Public_Sector_Cloud_Economics_A4.pdf [retrieved: June, 2013].

[27] Nagios, 2011. http://www.nagios.org/

[28] Open Certification Framework, https://cloudsecurityalliance.org/research/ocf/

[29] Ovum, http://www.computing.co.uk/ctg/news/2113323/ovum-public-cloud-services-market-explode

[30] PCI Security Standards Council, PCI DSS Quick Reference Guide: Understanding the Payment Card Industry Data Security Standard v2, https://www.pcisecuritystandards.org/documents/PCI%20SSC%20Quick%20Reference%20Guide.pdf [retrieved: June, 2013]

[31] S. Kamara and K. Lauter, “Cryptographic cloud storage”, 14th International Conference on Financial cryptograpy and data security, 2010, pp. 136-149.

[32] V. C. Emeakaroha, R.N. Calheiros, M.A.S. Netto, I. Brandic, and C.A.F. De Rose, “DeSVi: An Architecture for Detecting SLA Violations in Cloud Computing Infrastructures” 2nd International ICST Conference on Cloud Computing, 2010.

 ISBN: 978-1-61208-298-1

Detecting Social-Network Bots Based on Multiscale Behavioral Analysis

Francisco Brito, Ivo Petiz
Paulo Salvador, António Nogueira
Instituto de Telecomunicações, Aveiro, Portugal
DETI, University of Aveiro, Portugal
e-mails: franciscobrito@ua.pt, petiz@live.ua.pt,
salvador@ua.pt, nogueira@ua.pt

Eduardo Rocha
Instituto de Telecomunicações, Aveiro, Portugal
Faculty of Computer Science, HTWK Leipzig,
Germany
e-mail: eduardo.rocha@imn.htwk-leipzig.de

***Abstract*—Social network services have become one of the dominant human communication and interaction paradigms. However, the emergence of highly stealth attacks perpetrated by bots in social-networks lead to an increasing need for efficient detection methodologies. The bots objectives can be as varied as those of traditional human criminality by acting as agents of multiple scams. Bots may operate as independent entities that create fake (extremely convincing) profiles or hijack the profile of a real person using his infected computer. Detecting social networks bots may be extremely difficult by using human common sense or automated algorithms that evaluate social relations. However, bots are not able to fake the characteristic human behavior interactions over time. The pseudo-periodicity mixed with random and sometimes chaotic actions characteristic of human behavior is still very difficult to emulate/simulate. Nevertheless, this human uniqueness is very easy to differentiate from other behavioral patterns. As so, novel behavior analysis and identification methodologies are necessary for an accurate detection of social network bots. In this work, we propose a new paradigm that, by jointly analyzing the multiple scales of users' interactions within a social network, can accurately discriminate the characteristic behaviors of humans and bots within a social network. Consequently, different behavior patterns can be built for the different social network bot classes and typical humans interactions, enabling the accurate detection of one of most recent stealth Internet threats.**

Keywords - Facebook user behavior, Human social-networking behavior, social-network bots, bot detection, Facebook interactions model.

I. INTRODUCTION

Together with the growing predominance of social networking services in human communication, a set of scam attacks perpetrated by bots [1], [2] have emerged. We are currently witnessing a major increasing in cybercrime attacks against individuals targeting their personal data and financial assets [3], [4]. Most of the current Internet malware threats disseminate themselves using social engineering and, mainly, using social networks [5], since social networking provides an open field for illicit activities [6]. Social networking sites are always improving their security but this is a constant race behind the leading criminals [7]. Existing botnets [8], [9] can use social networking services to spread themselves, but more importantly, can use social networks to impersonate the owner of the controlled machine in order to obtain valuable personal information or force the person to interact with unwanted individuals or services. One of the better documented examples of social networking services abuse for malicious purposes was Koobface [10], [11]. Being currently the largest social networking service, Facebook is the main vector of attack via social networking services [12]. Current techniques to detect bots within a social network rely on automated algorithms that evaluate social relations. Based on graph-theory techniques, they try to detect unnatural relations in social networks [13], [9]. Another technique used to detect bot activity measures mouse movements and keystrokes produced while interacting in the generation of online contents. In [14] this class of behavioral analysis was applied in blogging activities, but it can also be easily applied to social networks interfaces. The main downside of this approach is that it must rely on software loaded on the client browser, which can be difficult to implement and certainly impossible to generalize to all users due to confidentiality constrains. A viable solution should only rely on ubiquitous statistics that do not compromise the users privacy, namely counting the number of social interactions per time interval (e.g., number of posts, number of likes, number of photo uploads).

It is extremely difficult to program a bot to replicate the characteristic human behavior of social interactions over time. Humans actions have an inherent pseudo-periodicity mixed with random (and sometimes chaotic) actions which are almost impossible to emulate/simulate. Nevertheless, this human uniqueness is very easy to differentiate from other behavioral patterns. Therefore, in this paper, we propose a new methodology that, by jointly analyzing the multiple scales of the users' interactions within a social network, can accurately discriminate the characteristic behaviors of humans and bots within a social network. Consequently, different behavior signatures can be use to accurately detect bots acting with a social network.

The proposed methodology applies the concept of multiscalling analysis based on scalograms to the statistical processes that describe the interaction of a user within the social network. Scalograms reveal much information about the nature of non-stationary processes that was previously hidden, so they are applied to a lot of different scientific areas: diagnosis of special events in structural behavior during earthquake excitation, ground motion analysis, transient building response to wind storms, analysis of bridge response due to vortex shedding, among others [15].

The remaining part of this paper is organized as follows: Section II presents some important background on multiscaling analysis; Section III presents the characteristic behaviors

 ISBN: 978-1-61208-298-1

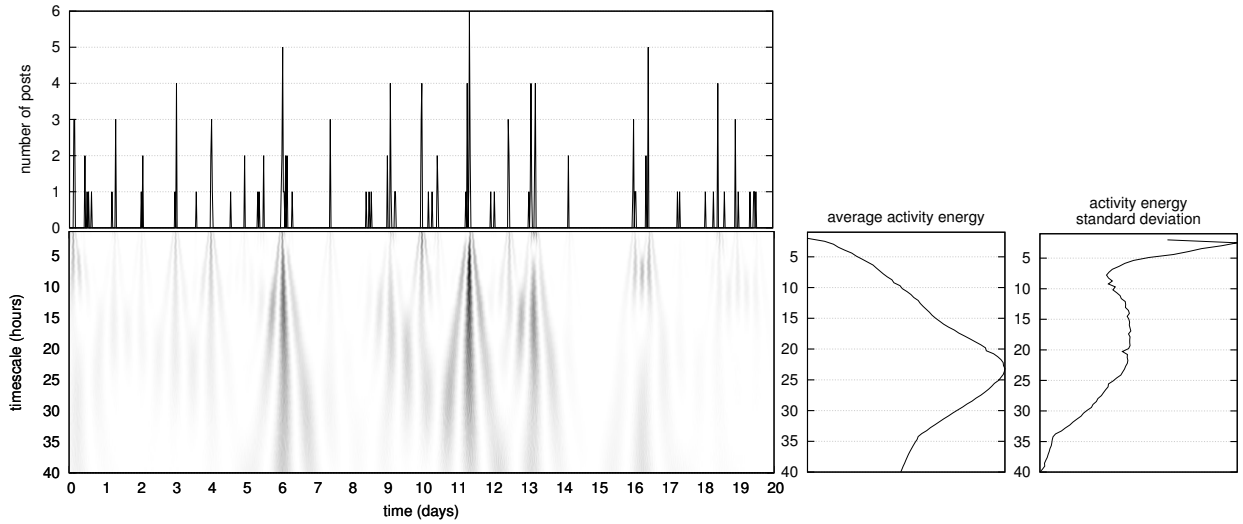


Figure 1. Multiscale analysis example: (top-left) number of Facebook posts in 20-minutes intervals, (bottom-left) scalogram / user activity energy per timescale over time, (bottom-middle) average user activity energy over time and (bottom-right) activity energy standard deviation over time.

of human and bots within a social network and how they differ; Section IV presents the results of a proof-of-concept of the proposed detection methodologies and, finally, Section V presents some brief conclusions about the presented detection methodologies.

II. Multiscaling analysis

The main purposes of a multiscaling analysis is to identify the most important time-scales of (pseudo-periodicity) activity and quantify the constancy of that pseudo-periodicity. In order to achieve that objective, it is necessary to quantify the activity over time for multiple timescales.

Wavelets are mathematical functions that are used to divide a given signal into its different timescales components. Wavelets enable the analysis of each one of the signal components in an appropriate scale. Starting with a mother wavelet $\psi(t)$, a family $\psi_{\tau,s}(t)$ of "wavelet daughters" can be obtained by simply scaling and translating $\psi(t)$:

$$\psi_{\tau,s}(t) = \frac{1}{\sqrt{|s|}}\psi(\frac{t-\tau}{s}) \tag{1}$$

where s is a scaling or dilation factor that controls the width of the wavelet (factor $\frac{1}{\sqrt{|s|}}$ is introduced to guarantee the energy preservation, $\|\psi_{\tau,s}\| = |\psi|$) and τ is a translation parameter controlling the time location of the wavelet. Scaling a wavelet simply means stretching it (if $|s| > 1$) or compressing it (if $|s| < 1$), while translating it simply means shifting its position in time.

Given a signal $x(t)$, its Continuous Wavelet Transform (CWT) with respect to the wavelet ψ is a function of time (τ) and scale (s), $W_{x;\psi}(\tau, s)$, obtained by projecting $x(t)$ onto the wavelet family $\{\psi_{\tau,s}\}$:

$$W_{x;\psi}(\tau, s) = \int_{+\infty}^{-\infty} x(t)\frac{1}{\sqrt{|s|}}\psi(\frac{t-\tau}{s})dt \tag{2}$$

By analogy with the terminology used in the Fourier case, the energy components of the signal are given by the square of the CWT components of the signal and the (local) Wavelet Power Spectrum (sometimes called Scalogram or Wavelet Periodogram) is defined as the normalized energy over time and scales:

$$E_x(\tau, s) = 100\frac{|W_{x;\psi}(\tau, s)|^2}{\sum_{\tau'}\sum_{s'}|W_{x;\psi}(\tau', s')|^2} \tag{3}$$

An example of a scalogram can be observed in Figure 1 (bottom-left).

III. Social-Networking Behavior Characterization

A. Single User Behavior Inference

Within the context of this paper, signal $x(t)$ is a counting process that quantifies the number of social network interactions (i.e., number of posts, number of likes and number of posted photos) in a time-interval (e.g., 30 minutes, as used in section IV) over time. An example of a counting process (Facebook posts) can be observed in Figure 1 (top-left). In order to characterize the user multiscale behaviour over time, it is possible to estimate (i) the Average Activity Energy of signal $x(t)$, $\mu E_x(s)$, by averaging the normalized energy of the signal over time for all timescales (see (4)) and (ii) the Activity Energy Standard Deviation of signal $x(t)$, $\sigma E_x(s)$, by calculating the standard deviation of the normalized energy of the signal over time for all timescales (see (5)):

$$\mu E_x(s) = \frac{1}{N}\sum_{i=1}^{N} E_x(\tau_i, s), \forall s \tag{4}$$

$$\sigma E_x(s) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(E_x(\tau_i, s) - \mu E_x(s)\right)^2}, \forall s \tag{5}$$

An example of these metrics can be observed in Figure 1, in the bottom middle and right plots, respectively.

 ISBN: 978-1-61208-298-1

B. User Group Behavior Inference

In order to characterize the behavior of a group of users, it is necessary to define metrics that can quantify their behavior as a group. Assuming a group size of U users and assuming that $x_u(t)$ represents a counting process that describes the activity of user u in the social network, we can quantify the mean and variance values of the (i) group average activity energy (see (6) and (7)) and (ii) group activity energy standard deviation (see (8) and (9)), for all timescales and users within the group:

$$\overline{\mu E(s)} = \frac{1}{U}\sum_{u=1}^{U} \mu E_{x_u}(s), \forall s \quad (6)$$

$$VAR(\mu E(s)) = \frac{1}{U-1}\sum_{u=1}^{U} \left(\mu E_{x_u}(s) - \overline{\mu E(s)}\right)^2, \forall s \quad (7)$$

$$\overline{\sigma E(s)} = \frac{1}{U}\sum_{u=1}^{U} \sigma E_{x_u}(s), \forall s \quad (8)$$

$$VAR(\sigma E(s)) = \frac{1}{U-1}\sum_{u=1}^{U} \left(\sigma E_{x_u}(s) - \overline{\sigma E(s)}\right)^2, \forall s \quad (9)$$

IV. Results

The proposed methodology was applied to two different data-sets. The first data-set comprises Facebook posted between January 1st, 2007 and December 31, 2008 by a group of 160 users, which individually posted more than 300 posts in this two year time frame. The total number of recorded Facebook posts is 72893. This 2007-2008 data-set was extracted from the data presented by Viswanath *et al.* [16] and is freely available online. The second data-set was extracted from a group of Facebook friends of this paper authors using the Facebook Graph API [17]. Facebook Graph API is a low level HTTP-based API used to query data from Facebook's Social Graph. The data queried using Facebook Graph API is returned in JSON format and can be easily post-processed in order to extract its relevant statistics. This data-set contains all social networking activities of 140 users between January 1st, 2011 and December 31, 2012. Only users with more than 300 interactions with the social network in the two years time frame were considered. The total number of Facebook interactions in this data-set are 167171 Facebook posts, 90882 likes and 48755 photo uploads. The second data-set is hereafter referred as 2011-2012 data-set.

Based on the timestamps of each social network interaction, time processes were extracted from the data-sets by counting the number of interactions in 20-minute time intervals. For the 2007-2008 data-set only Facebook posts were considered due the the limited information of the original data. However, for the 2011-2012 data-set we considered separately the Facebook posts, likes and photo uploads.

A. Facebook User Behavior Evolution

We have applied the proposed multiscale user activity characterization to Facebook posts time processes extracted from both data-sets. The obtained results are depicted in Figure 2.

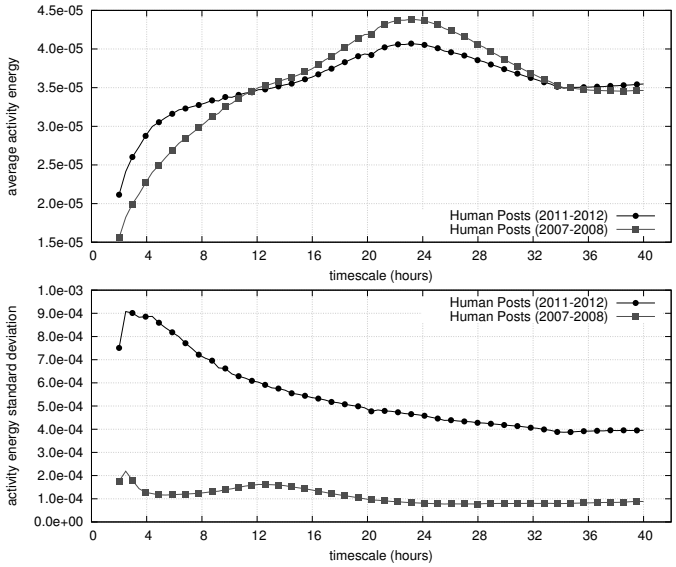


Figure 2. Evolution of multiscale characteristics of human posts on Facebook from 2007-2008 to 2011-2012: (top) average energy activity over time for all scales and (bottom) standard deviation of energy activity over time for all scales.

The average energy activity reveals that pseudo-periodicity with a 24 hours period is predominant for both data-sets. This fact reveals that humans behave and interact with social networks in 24 hour cycles, although humans also spread their pseudo-periodic intervals between activity over a wide range of timescales. Another fact that can be observed is that nowadays users tend to have a more spread usage when compared to the older data-set; this reveals that human users have evolved to a more frequent Facebook interactivity, so their profiles are not so deeply shaped according to the 24-hour cycle. This evolution on the characteristics of the human activities is more visible when analyzing the activity energy standard deviation, which is much higher on the newer data-set. This also show that human users have a very variable behavior in terms of intervals between activities on Facebook.

B. Human and Bot Differentiation

In order to present a proof-of-concept for our methodology as a social network bot detection tool, we created two different bots that emulate Facebook posts, likes and photo uploads according to different behavioral profiles, namely, one periodic bot and one exponential bot. A periodic bot interacts with Facebook in exact intervals (in this paper, we have considered a periodicity of 24 hours) and an exponential bot interacts with Facebook in exponential distributed intervals (in this paper, we considered an average interval of 24 hours).

We have applied the proposed multiscale user activity characterization to human users of 2011-2012 data-set and to the emulated bots. Figures 3, 4 and 5 depict the multiscale characteristics (with 98% confidence intervals) of human and bot Facebook users when making posts, likes and uploading photos, respectively. The results show that the multiscale behaviors are similar when observing the Facebook posts, likes and photo uploads. From the obtained results it is also possible

 ISBN: 978-1-61208-298-1

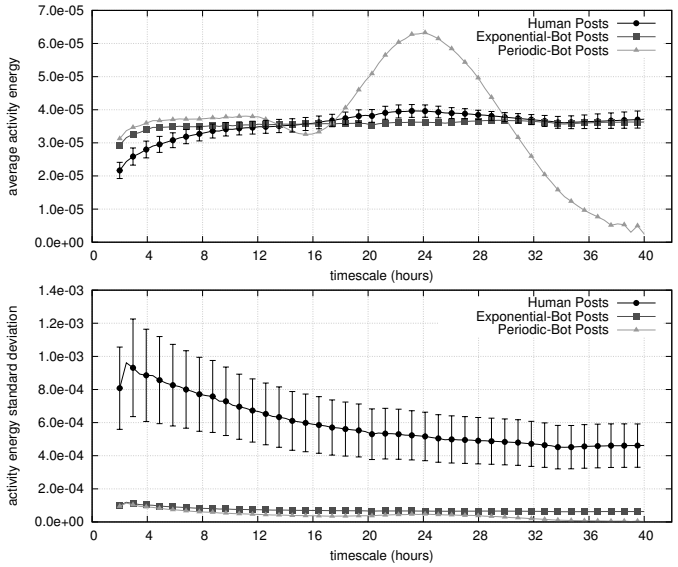


Figure 3. Multiscale characteristics of human and bot posts on Facebook in 2011-2012 dataset: (top) average energy activity over time for all scales and (bottom) standard deviation of energy activity over time for all scales, with 98% confidence intervals.

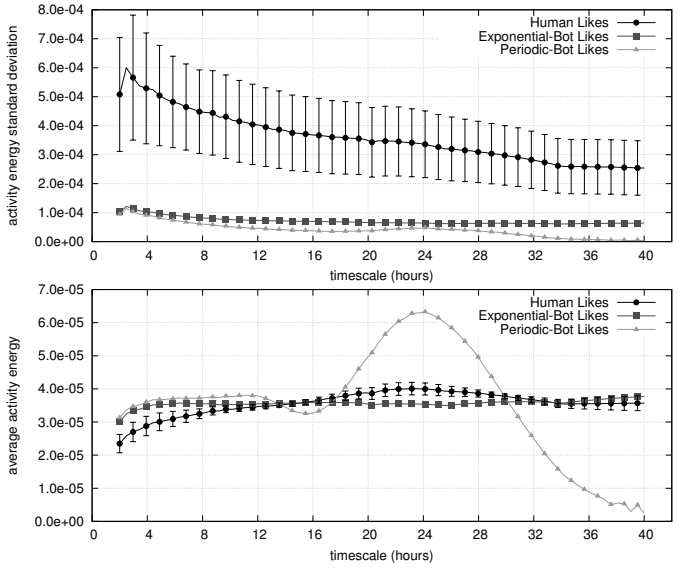


Figure 4. Multiscale characteristics of human and bot likes on Facebook in 2011-2012 dataset: (top) average energy activity over time for all scales and (bottom) standard deviation of energy activity over time for all scales, with 98% confidence intervals.

to observe that bots and humans have distinct multiscale behaviors. Periodic bots have their average activity energy centered on a 24-hour timescale and have low energy variation. Exponential bots and human users have a similar average activity energy distribution over the timescales, however, human users still have slighter higher energy around the 24-hour timescale. However, the variation of activity energy over time is much higher in human users.

The multiscale characteristics of human activities in social networks have a pseudo-periodicity of 24-hour, however, the human analysis reveals an inherent chaotic and unpredictable behavior shown by the much higher variation of activity energy

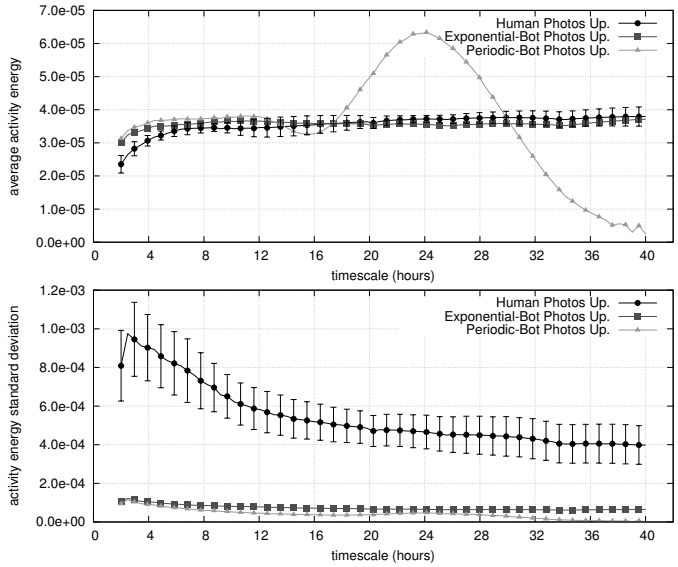


Figure 5. Multiscale characteristics of human and bot photo uploads on Facebook in 2011-2012 dataset: (top) average energy activity over time for all scales and (bottom) standard deviation of energy activity over time for all scales, with 98% confidence intervals.

over time. This inherent chaos of the behavior is extremely difficult to emulate by bots and can be used to differentiate human from bot users within a social network.

V. Conclusion

In this paper, a novel paradigm was proposed to perform the joint analysis of multiple scales of users' interactions within a social network. The presented methodology allows an accurate discrimination between human and bot behaviors within social networks. The results obtained reveal that multiscale behavior signatures can be built for different social network bot classes and typical human interactions, which will enable the development of accurate tools for the detection of social networks bots.

References

[1] S. Sengupta, "Bots Raise Their Heads Again on Facebook," The New York Times - Bits Blog, Jul. 2012.

[2] E. Gamma, "Your Facebook Friends May Be Evil Bots," InfoWorld, Apr. 2013.

[3] E. Kraemer-Mbula, P. Tang, and H. Rush, "The cybercrime ecosystem: Online innovation in the shadows?" Technological Forecasting and Social Change, vol. 80, no. 3, Mar. 2013, pp. 541–555.

[4] W. Kim, O.-R. Jeong, C. Kim, and J. So, "The dark side of the internet: Attacks, costs and responses," Information Systems, Special Issue on WISE 2009 - Web Information Systems Engineering., vol. 36, no. 3, 2011, pp. 675–705.

[5] S. Abraham and I. Chengalur-Smith, "An overview of social engineering malware: Trends, tactics, and implications," Technology in Society, vol. 32, no. 3, Aug. 2010, pp. 183–196.

[6] G. R. Weir, F. Toolan, and D. Smeed, "The threats of social networking: Old wine in new bottles?" Information Security Technical Report, vol. 16, no. 2, May 2011, pp. 38–43.

[7] P. Jagnere, "Vulnerabilities in social networking sites," in 2nd IEEE International Conference on Parallel Distributed and Grid Computing (PDGC 2012), Dec. 2012, pp. 463–468.

[8] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles, "Botnets: A survey," Computer Networks, Fev. 2013, pp. 378–403 .

 ISBN: 978-1-61208-298-1

[9] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, “Design and analysis of a social botnet,” Computer Networks, Fev. 2012, pp. 556–578.

[10] R. Fergusonnam, “Back to the future,” Network Security, vol. 2010, no. 1, Jan. 2010, pp. 4–7.

[11] K. Thomas and D. Nicol, “The koobface botnet and the rise of social malware,” in 5th International Conference on Malicious and Unwanted Software (MALWARE 2010), Oct. 2010, pp. 63–70.

[12] D. Bradbury, “Spreading fear on facebook,” Network Security, vol. 2012, no. 10, Oct. 2012, pp. 15–17.

[13] G. Yan, “Peri-watchdog: Hunting for hidden botnets in the periphery of online social networks,” Computer Networks, vol. 52, no. 2, Fev. 2012, pp. 540–555.

[14] Z. Chu, S. Gianvecchio, A. Koehl, H. Wang, and S. Jajodia, “Blog or block: Detecting blog bots through behavioral biometrics,” Computer Networks, vol. 57, no. 3, Fev. 2013, pp. 634–646.

[15] K. Gurley and A. Kareem, “Applications of wavelet transforms in earthquake, wind, and ocean engineering,” Engineering Structures, no. 21, 1999, pp. 149–167.

[16] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, “On the evolution of user interaction in facebook,” in Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN’09), Aug. 2009, pp. 37–42.

[17] “Graph API - Facebook Developers,” http://developers.facebook.com/docs/reference/api/, 2013, [Online; accessed April-2013].

 ISBN: 978-1-61208-298-1

Policy-Aware Provisioning of Cloud Applications

Uwe Breitenbücher, Tobias Binz, Oliver Kopp, Frank Leymann, Matthias Wieland
Institute of Architecture of Application Systems
University of Stuttgart, Stuttgart, Germany
{breitenbuecher, lastname}@iaas.uni-stuttgart.de

***Abstract*—The automated provisioning of complex composite Cloud applications is a major issue and of vital importance in Cloud computing. It is key to enable Cloud properties such as pay-as-you-go pricing, on-demand self-service, and elasticity. The functional aspects of provisioning such as instantiating virtual machines or installing software components are covered by several technologies on different technical levels: some are targeted to a pretty high level such as Amazon's Cloud Formation, some deal with deep technical issues based on scripts such as Chef or Puppet. However, the currently available solutions are tightly coupled to individual technologies without being able to consider non-functional security requirements in a non-proprietary and interoperable way. In this paper, we present a concept and framework extension enabling the integration of heterogeneous provisioning technologies under compliance with non-functional aspects defined by policies.**

Keywords*—*Cloud Applications; Provisioning; Security; Policies

I. INTRODUCTION

The increasing use of IT in almost every part of enterprises leads to a higher management effort in terms of development, deployment, delivery, and maintenance of applications. For enterprises, IT management becomes a challenge as each new technology increases the degree of complexity while operator errors account for the largest fraction of failures [1]. These issues have been tackled by outsourcing IT to external providers and automating the management of IT—both enabled by Cloud computing [2]. The modular component-based architectures that are the consequence of using Cloud services allow to benefit from Cloud properties such as elasticity, scalability, and high availability without the need to have technical insight [3]. Unfortunately, the necessary balance between functional possibilities and non-functional security issues has been often skewed toward the first. Cloud services are typically easy to use on their own but hard to configure and extend in terms of non-functional aspects that are not covered natively by the offering. Creating applications that integrate different heterogeneous components which are hosted on or interact with Cloud services while fulfilling non-functional security requirements can quickly degenerate to a serious problem, especially if the technical insight is missing. Even the initial provisioning of applications can be a difficult challenge if non-functional requirements of different domains with focus on heterogeneous technologies have to be fulfilled.

In this paper, we present a concept and extend an existing Management Framework [4] to tackle these issues. The extension enables the fully automated provisioning of composite Cloud applications complying with non-functional security requirements from different domains which are expressed by *Provisioning Policies*. We introduce *Policy-Aware Management Planlets* that allow the implementation of policy-aware provisioning logic in a reusable way independently from individual applications. The presented framework enables Cloud providers as well as application developers to specify security requirements for the provisioning without the need to have the deep technical management knowledge needed in other approaches. In addition, the concept allows security experts of different domains to work together in a collaborative way. We evaluate the approach in terms of performance, complexity, economics, feasibility, extensibility, and the implementation of a prototype. The remainder of this paper is structured as follows. In Section II, we explain fundamentals and motivate our approach by a scenario presented in Section III. In Section IV, we introduce Provisioning Policies. Section V describes the Management Framework that is extended by our approach, which is presented in Section VI. Section VII evaluates the approach and Section VIII reviews related work. We conclude this paper and give an outlook on future work in Section IX.

II. FUNDAMENTALS

In this section, we explain two fundamental concepts providing the basis for the remainder of the paper: (i) Application Topologies and (ii) Management Plans.

A. Application Topologies

An application topology is a directed, possibly cyclic graph describing the structure of an application. It consists of nodes, which represent the different components of an application such as Virtual Machines (VM) or Java applications, and the relations among them, which are the edges between the nodes. Nodes and relations are called *elements* of the topology and have a type attribute defining its semantics. Each element may have a set of arbitrary properties used to describe the element in detail. Figure 1 shows an example, which describes a PHP-based Web shop application that stores data in a database backend. We use the visual notation *Vino4TOSCA* [5] to depict topologies graphically. Following this notation, the type of an element is enclosed by parentheses whereas its name is undecorated text. The application's infrastructure is provided by Amazon's public Cloud in the form of two virtual machines of type *AmazonEC2VM*. Thereon, operating systems are installed of type *Windows7* and *UbuntuLinux*. A PHP runtime of type *ApachePHPWebServer* hosts the application of type *PHP*. This application connects to a database of type *MySQL* which is hosted on the Linux operating system of the right stack. We modeled all relations as *hostedOn* relation and left out properties to simplify the diagram. Our approach employs application topologies to describe the structure of applications to be provisioned and to attach policies to the contained elements.

 ISBN: 978-1-61208-298-1

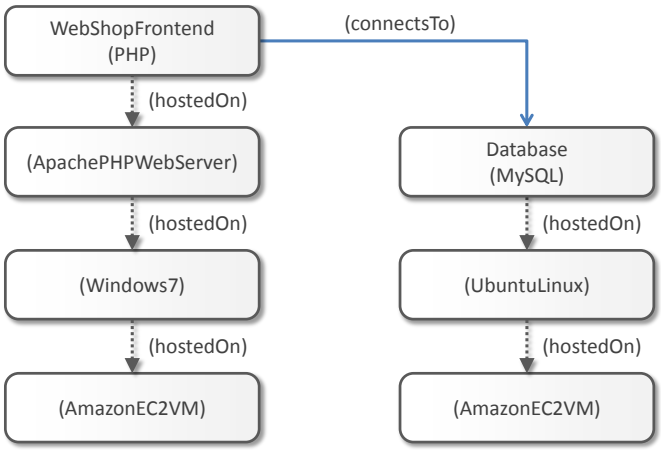


Figure 1. Example of a PHP-based application topology.

B. Management Plans

Management Plans are executable workflows used to automate the management of applications such as the provisioning of applications. They enable a much more robust and reliable way to manage applications than the manual or script-based management through technologies on a deep technical level. They inherit features from workflow technology such as recoverability, compensation, and fault handling mechanisms. One workflow language to implement these plans is the Business Process Execution Language (BPEL), which is, for example, used for the provisioning of applications [6]. Management Plans are often created manually by the application developers themselves. However, this is a difficult and time-consuming task and, as plans are typically coupled tightly to single applications, of limited value: Plans are mostly sensitive to structural differences and, therefore, hardly reusable for the management of other applications. In this paper, we generate so-called *Provisioning Plans*, which are a subclass of Management Plans, to fully automate the provisioning of applications based on their topologies.

III. Motivating Scenario

The following scenario is used to motivate the approach presented in this paper. Let us assume a company wants to host the PHP-based Web shop application described in the previous section on Amazon's public Cloud (Figure 1). Depending on the importance of that application for the company, there may be different non-functional security requirements. For example, the Web server hosting the functional logic typically comes up with default credentials, i. e., username and password are both "admin". If this is ignored, the Web server is open for attacks and may be misused. Thus, we need a mechanism to ensure that the strength of these credentials is considered during the deployment. The second example deals with the products data stored in the database. If the company financially depends on this Web shop, data loss caused by a failure may ruin the business. Thus, a frequent data backup of this database is needed. The last example deals with the geographic location of the company's customer data, which is also stored in the database. Legal aspects may require that this data has to remain in the European Union (EU). Thus, the virtual machine must be hosted on a physical server located in one of its states.

IV. Policies for Provisioning of Applications

In this section, we introduce *Provisioning Policies* which are used by our approach to express non-functional security requirements on the provisioning of applications. Provisioning Policies are a subclass of Management Policies which are employed in systems and network management. We describe Management Policies in detail and refine them into Provisioning Policies afterwards to clearly define the kinds of policies that are supported by the presented approach.

A. Management Policies

Management Policies are a well-known concept and common in research as well as in industry. They are derived from management goals and used to influence the management of applications, resources, and IT in general based on non-functional aspects such as security, performance, or cost requirements. They provide a (semi-) formal concept used to capture, structure, and enforce the objectives [7]. A lot of work on policies exists dealing with classifications, methodologies, and applications. To classify and identify the policies covered by our approach, we use the hierarchy of Wies [7] that classifies policies based on the level on which they influence the management. The hierarchy was developed based on criteria such as policy life-time, how they are triggered and performed, and the type of its targets. Wies differentiates between four classes: (i) Corporate/High-Level Policies, (ii) Task Oriented Policies, (iii) Functional Policies, and (iv) Low-Level Policies. Corporate Policies are directly derived from corporate goals and embody *strategic business management* rather than technical management aspects. The other three classes embody *technology oriented management* in terms of applying management tools, using management functions, and direct operation on the managed objects. Our approach covers the technology oriented management in terms of security. The most important requirement of security policies is to ensure strict adherence. The system must prevent that the security requirements defined by a policy get violated. This is vital as many types of policies cause actions that cannot be undone if once violated. For example, if a Data Location Policy defines that the application data must not leave a certain region (legal rights), i. e., the physical servers must be located in that region, and the data gets distributed over the world by a public Cloud, then the policy is violated and it is impossible to undo this violation. Violating such policies may result in high penalties.

B. Provisioning Policies

Provisioning Policies are a subclass of Management Policies and responsible for ensuring non-functional requirements during the provisioning of applications. During our research on application provisioning, we identified three different kinds of policies that must be considered: (i) Configuring Policy, (ii) Guarding Policy, and (iii) Extending Policy. We explain these three kinds in the following and give examples based on the motivating scenario introduced in Section III. Our approach supports primarily these three kinds of policies. A *Configuring Policy* configures the provisioning of components or relations. For example, a *Data Location Policy* attached to a virtual machine with *region* value "EU" configures the provisioning in a way that the virtual machine is hosted on a server located in the European Union. A *Guarding Policy* guards the provisioning

 ISBN: 978-1-61208-298-1

of components or relations, i. e., it supervises the instantiation in terms of certain specified values or thresholds. For example, a *Safe Credentials Policy* ensures that the strength of login credentials, i. e., username and password, is strong enough. A *Extending Policy* extends the provisioning in terms of structure, i. e., it may add new components or relations which are not contained in the original application topology to be provisioned. For example, a *Frequent Data Backup Policy* for a database causes the installation of an additional software component on the operating system which backups specified database tables in a certain time interval to a certain location.

V. Used Management Framework

In this section, we explain the Management Framework that gets extended to support processing non-functional security policies. We presented this framework in a former paper [4], which describes all details about the framework, Management Planlets, and Management Annotations.

The main functionality of the framework is managing running applications by generating Management Plans that are executed to perform the desired changes. Therefore, a Plan Generator gets a so-called *Desired Application State Model* as input which defines the desired state the application shall have after executing the management task. This model is an application topology which reflects the desired state: Components may have to be added or removed, relations must be established, or updates must be installed, to mention a few examples. To define the low-level management tasks that have to be performed to achieve the desired state, we use so-called *Management Annotations*, which are described in the next section. Based on this topology, a Management Plan is generated that brings the application from the current state to this desired state, i. e., performs the management tasks. This generation orchestrates so-called *Management Planlets* that provide small management tasks such as installing or removing components or establishing a database connection. Details are explained in Subsection V-A. All available planlets are stored in a repository which is used by the plan generator to find appropriate planlets to generate the plan. For example, if such a management task defines that a database of type *MySQL* must be installed on an existing operating system of type *UbuntuLinux* and data needs to be imported afterwards, the plan generator uses a planlet that installs the database and another planlet to import the data. Thus, planlets enable separating different functional low-level tasks in order to be orchestrated to provide a higher level of management tasks.

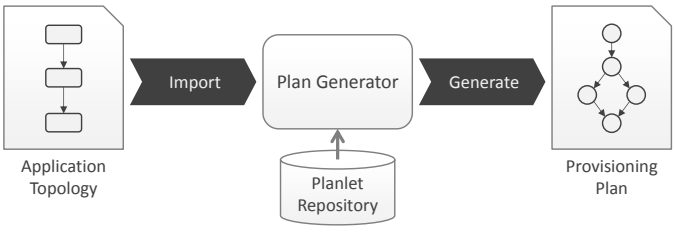


Figure 2. Management Framework for generating Provisioning Plans.

The framework also supports generation of provisioning plans, i. e., plans that provision a complex composite Cloud application on each level: from infrastructure to software components with all required relations among them. This is shown in Figure 2: The plan generator gets an application topology as input and generates the corresponding Provisioning Plan by orchestrating several planlets. In addition, the whole application to be provisioned is packaged into a so-called *Application Package* that contains all artifacts needed for provisioning such as installables. Planlets may be also contained in this package. This enables the developer to implement own custom planlets that are taken into account by the plan generator. Thus, the concept of planlets enables even the developers of the application themselves to define and influence the provisioning logic. However, the framework currently does not support the consideration of non-functional requirements. Therefore, one contribution of this paper is making the framework policy-aware to tackle this issue. In the next subsections, we provide more details about the framework to help understanding our extension presented in Section 6.

A. Orchestration of Management Planlets

Management Planlets are small workflows implementing reusable low-level management tasks such as creating a virtual machine on Amazon EC2, installing a Tomcat on an operating system, or establishing a connection between an application's frontend and its database backend. Planlets are intended to be orchestrated by so-called Management Plans, which provide a higher level of management tasks such as the provisioning of a whole application consisting of multiple interconnected nodes. Thus, they serve as generic building blocks for the generation of Management Plans. In general, planlets are used to manage a running application, but they can be used for the initial provisioning as shown in [4], too. Management Planlets express their functionality through an annotated application topology fragment. This fragment contains a small application topology consisting of typed nodes and relations which may be annotated with so-called *Management Annotations*. These annotations express small low-level management tasks that are performed by the planlet on the respective nodes or relations. The types of components and relations and the annotations are used by the plan generator to find suitable planlets for the desired tasks. A more detailed description of these annotations is given in Section V-B. A single planlet may perform multiple of these small management tasks in order to implement a more sophisticated task, e. g., a single planlet may install a database on an operating system and imports data afterwards. Planlets often need to express preconditions which must be fulfilled to make the planlet applicable, e. g., the previous mentioned planlet which installs the database on an operating system and imports data afterwards requires the operating system to be already running. These preconditions are also expressed through the topology fragment: all therein contained elements, i. e., all nodes and relations, and their properties which have no Management Annotations are treated as precondition. Thus, the operating system would be contained in the topology fragment without any annotation but with state-property value "Instantiated" as shown in Figure 4. This indicates that the operating system must be running. The state-property may be set by another planlet which provisioned the virtual machine and the operating system before. Thus, the ordering and selection of planlets in the overall Management Plan is calculated based on these properties and Management Annotations. During the plan generation, a virtual representation of the current state of the application gets

 ISBN: 978-1-61208-298-1

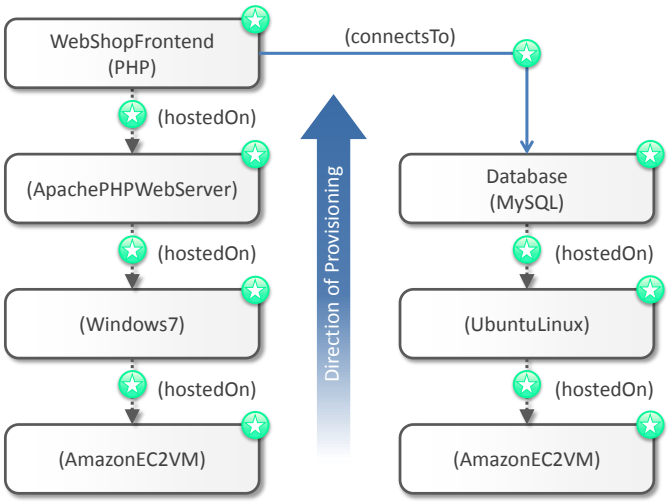


Figure 3. Annotated application topology to be provisioned.

transferred towards the goal state, i. e., all elements are created. In each state, all planlets whose preconditions match the current virtual state are called *Candidate Planlets*. These planlets are eligible to be applied. The plan generator then decides which planlet to choose to transfer the application into the next state. Planlets may also communicate with each other via properties, e. g., one planlet writes the IP-address and credentials of a Web server to the corresponding node which are used by another planlet to deploy applications on it. In addition, planlets may have input parameters used to get information they need from the user. These parameters are exposed to the input message of the overall generated Management Plan. For example, if a planlet needs Amazon credentials to acquire a virtual machine, it defines them in its local input message and the system exposes them to the global input message of the overall generated plan. Therefore, the plan generator adds activities transferring the data internally to the planlet. In the next section, we explain the used Management Annotations in detail.

B. Management Annotations

Management Annotations express low-level management tasks which have to be performed on the nodes and relations they are attached to. They are subdivided into two disjoint classes: *Structural Management Annotations* and *Domain-Specific Management Annotations*. The first class contains annotations that structurally change the application in terms of creating or destroying nodes or relations. Thus, there is a *Create-Annotation* and a *Destroy-Annotation*. For plain provisioning of applications only the Create-Annotation is used and annotated to all elements in the topology model as shown in Figure 3. The green circle with the star inside represents the Create-Annotation. These annotations tell the system that the corresponding nodes and relations shall be created. Figure 4 shows a planlet fragment that may be used to provision a part of this topology: The shown planlet creates a node of type *MySQL* and a relation of type *hostedOn* to an already existing node of type *Linux*, i. e., it installs a MySQL database on a Linux operating system. The precondition of the planlet is that the operating system is already running which is expressed by its state-property set to “Instantiated”. The MySQL node to be created also has this state-property but with a Create-Annotation, which means that the planlet sets this property to the specified value “Instantiated”. This property may be a precondition for another planlet which connects an application to this database or imports initial data. Thus, the installation planlet has to be executed before them. Application components connected by *hostedOn* relations are typically instantiated bottom up, i. e., the first planlets matching the topology are those creating the infrastructure and middleware components. After creating the basis, other planlets are applicable that have the existence of those components as precondition.

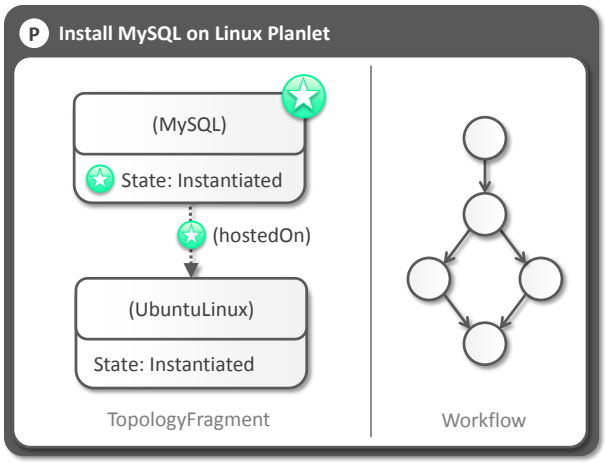


Figure 4. Planlet installing a database on a running operating system.

The second class is used to express domain-specific management tasks such as importing data into a database. Management Annotations are identified by a unique id, which is used for matchmaking. Thus, this enables defining any additional management task needed to configure the overall provisioning.

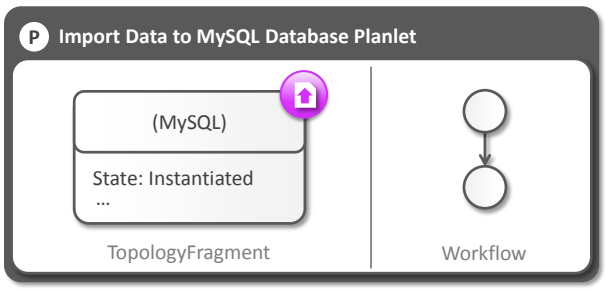


Figure 5. Planlet doing a database import.

Figure 5 shows a planlet that imports data to a database. This is expressed by the domain-specific *ImportData-Annotation* attached to the MySQL node, which is depicted by the purple cycle. This planlet must not be executed before the database is instantiated because of the state-property precondition. Therefore, considering the two example planlets shown in Figure 4 and Figure 5, the planlet that installs the database has to be executed before the planlet doing the import. Based on this concept, planlets get selected and ordered by the plan generator. Of course, the second planlet may have much more property preconditions such as database credentials and endpoint information. We ignore these to simplify the figures.

 ISBN: 978-1-61208-298-1

The concept allows distributing logic across several planlets which do not need to know each other. Each planlet implements a small functionality and can be used in combination with other planlets to achieve an overall goal. The approach also enables separating concerns in terms of management domains: Database experts are able to implement knowledge about databases into planlets without the need to know anything about the PHP applications connecting to databases. All in all, this provides a holistic and collaborative framework for managing applications.

VI. Policy-aware Provisioning of Complex Composite Cloud Applications

In this section, we present the two major contributions of this paper. We propose (i) a concept for defining and processing the Provisioning Policies introduced in Section IV and (ii) show how the Management Framework presented in Section V is extended to support policy-aware provisioning of applications.

The general concept is based on attaching policies to elements in application topologies and elements in planlet fragments which are matched during plan generation to compare *non-functional requirements* of the topology and *non-functional capabilities* offered by planlets. Therefore, we introduce *Policy-aware Management Planlets* that consider policies. This concept allows a fine grained definition of non-functional requirements and capabilities targeted directly to individual elements while preserving the functional description. In contrast to many existing bidirectional policy processing approaches, we use a strict one-way requirement-driven perspective for policies: policies attached to an application topology define requirements whereas policies attached to the fragment of a planlet define the planlet's capabilities. Planlets cannot express non-functional requirements and topologies cannot express capabilities. Thus, the planlet's policies may be ignored if they are not required.

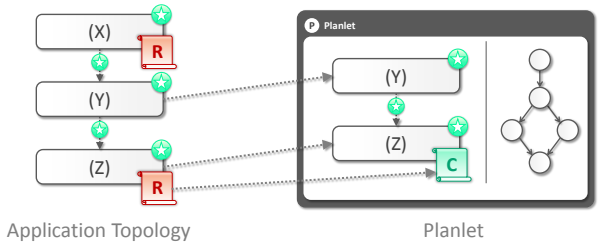


Figure 6. General concept of the presented approach.

Figure 6 explains the concept visually: On the left, it depicts an application topology consisting of three components connected by hostedOn relations that have to be provisioned. The node of type X and Z have a policy attached, which defines the non-functional security requirements that have to be fulfilled during the provisioning. On the right, there is a Policy-aware Management Planlet that provisions nodes of type Z and Y connected by a hostedOn relation. The policy attached to the node of type Z expresses the non-functional capabilities that may be provided by the planlet for that provisioning. During plan generation, the policies are compared and checked for compatibility. If the planlet fulfills all policies attached to elements in the topology that are covered by its topology fragment, the planlet is applicable. In the next section, we introduce the technical structure of Provisioning Policies.

A. Structure and Properties of Provisioning Policies

In this section, we introduce the format for Provisioning Policies to bind non-functional security requirements and capabilities to the tasks performed by planlets. A Provisioning Policy has a mandatory unique *id* within the topology it is contained in and an optional *type* defining the semantics of the policy, e. g., a policy of type *Safe Credentials Policy* ensures that username and password of a component to be provisioned are strong enough to resist attacks. The semantics of a policy type have to be well-defined and documented, i. e., application topology modelers and planlet developers must be aware of its meaning and how to use and implement it. There are a lot of existing policy languages in research and industry such as WS-Policy, Ponder, or KAOS [8]. Our approach supports their integration through an optional *content* field and an optional *language* attribute: The content field enables to fill in any policy- or language-specific information whereas the language attribute defines the used language. A processing mode attribute defines how the policy has to be fulfilled, e. g., whether it is sufficient to compare only the types of topology and planlet policy or if the content of the policy needs to be analyzed. That is the reason why the type and language attributes are optional: if only the type need to be compared, the language attribute is unnecessary. This is explained in detail in Section VI-D. The processing mode is used only by policies attached to elements in the topology as only they impose requirements. Each policy has an attribute *optional* that defines if the processing of the policy is mandatory. Topologies may use optional policies to express security requirements which are "nice to have" but not necessarily required. Planlets may use optional policies to enable configuration and reusability by offering additional non-functional capabilities that are fulfilled only if explicitly required. To target policies exactly to the affected management tasks, each policy in the topology may define the management tasks that have to consider the policies. Therefore, Provisioning Policies have a *MustProcess* list that may contain Management Annotations. All planlets implementing one of these tasks must consider the policy. If this list is empty, all planlets must consider the policy. To add exceptions, each policy has a *NeedNotProcess* list containing the Management Annotations that do not have to consider the policy. This concept allows separating concerns and targeting policies exactly to the affected tasks: Only planlets executing the Management Annotations defined in the MustProcess list must consider the policy while all other planlets do not have to care.

Figure 7 shows Data Location Policies attached to the application used in the motivating scenario. It depicts two Data Location Policies which are attached to the virtual machine and database node. Both are defined in the same language and must be processed by a type specific plugin. The difference lies in the MustProcess lists: The virtual machine policy must be considered only for its creation, which is depicted by the Create-Annotation in the MustProcess list. The database policy must be considered only by planlets that handle data, e. g., import or export data. This is expressed by the domain-specific *DataHandling-Annotation* depicted as blue circle with white paper inside. This differentiation makes sense as the policies express requirements on different tasks. As the location of the physical servers the virtual machine is hosted on determines also the geographic location of the database and, thus, of the data itself, the VM has to be located in the region the data has

 ISBN: 978-1-61208-298-1

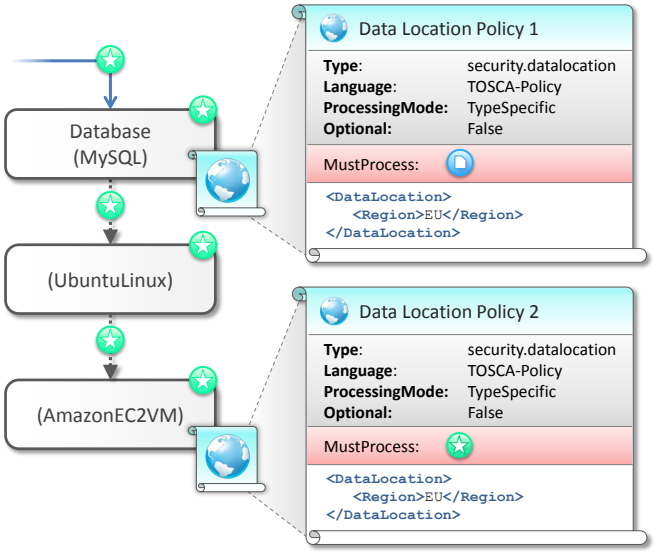


Figure 7. Two Data Location Policies attached to an application topology.

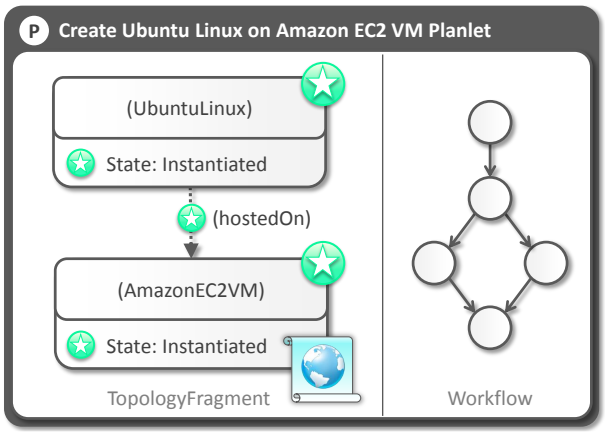


Figure 8. Planlet that instantiates a virtual machine with Linux operating system fulfilling a Data Location Policy.

to remain. Thus, the planlet instantiating the VM must consider this policy, e. g., as shown by the planlet depicted in Figure 8. In contrast, the location of the database needs not to be considered by the planlet installing it on the operating system. Therefore, the planlet shown in Figure 4 can be reused although it does not define any non-functional location information at all. However, handling data, e. g., export data, needs special considerations on the database layer because the data has to remain in the European Union, too. Thus, this concept allows a fine-grained definition of requirements on different levels.

B. Extending Management Annotations

Management Annotations are atomic entities that define either structural or domain-specific self-contained management tasks as explained in Section V-B. This is not sufficient for working with policies as the atomicity allows no abstract definition of those tasks. For example, the Data Location Policy attached to the MySQL database as shown in Figure 7 needs to be processed by all planlets having Management Annotations that deal with data, e. g., planlets exporting data must ensure that they do not store the backup at locations violating the policy. As the complete set of annotations may be unknown in advance, we need a mechanism to classify annotations of certain kinds of tasks. In particular, there are Management Annotations of type Import Data or Export Data as shown in Figure 5 that need to fulfill the Data Location Policy, i. e., data to be imported or exported must not be stored on servers outside the European Union. Listing all these annotations in the MustProcess list of the policy is not sufficient as mentioned before. Thus, we extend the concept by introducing inheritance: The data import and data export Management Annotations inherit all properties from a superclass annotation of type *DataHandling*. For example, the Data Location Policy in Figure 7 specifies that all Management Annotations having this superclass must process the policy. This extension allows defining abstract tasks which can be bound to policies in a generic way. Thus, if the framework processes a policy having this annotation, it ensures that all planlets handling data take this Data Location Policy into account. To achieve flexibility, we also allow multi-inheritance.

C. Plan Generator Extension

The plan generator of the framework tries to find planlets that can be orchestrated to provision the application. During this generation, a set of candidate planlets is calculated for each state and the planner decides which of the candidate planlets is applied next—as introduced in Section V-A. This calculation is based on compatibility: A planlet is applicable if each element in the planlet's fragment can be mapped to a compatible element in the topology. This means, that all preconditions of the planlet are fulfilled and that the management tasks which are implemented by the planlet and expressed in the form of Management Annotations are also specified in the topology. Details about this compatibility check can be found in [4]. We extend this compatibility check by taking policies into account: For each element in the topology that has an attached policy, the policy needs to be processed as defined by the processing mode attribute. How to deal with this processing mode attribute during matchmaking is explained in the next section.

D. Policy Processing Modes and Matchmaking

Each policy specifies a processing mode that defines how the policy has to be checked during the matchmaking of topology and planlets. We introduce three processing modes: (i) Type Equality, (ii) Language Specific, (iii) Type Specific. The mode defines the *minimum criterion* that must be met to fulfill the policy. Thus, the modes are ordered: from weak (Type Equality) to strong (Type Specific). Every stronger criterion outvotes the weaker criteria, i. e., if a policy defines processing mode *Type Equality*, which cannot be fulfilled by any planlet but there is a planlet fulfilling the *Language Specific* processing mode, the policy is fulfilled. Thus, if a criterion cannot be met, the system tries the next stronger criterion automatically. If it does not matter which criterion has to be met, this is equal to set processing mode to *Type Equality*.

1) Type Equality: This processing mode defines that only the types of two policies must be equal to fulfill the policy attached to an element in the topology. That means, that for each policy attached to an element in the topology there must be a policy of the same type attached to the corresponding element in the planlet's fragment. This processing mode is sufficient for policies that can express all their requirements and needs

 ISBN: 978-1-61208-298-1

by a well-defined keyword used as type, for example, a *No Connection To Internet Policy* attached to a virtual machine node is expressive enough to define the requirement.

2) Language Specific: Language specific processing means that the policy must be processed by a dedicated framework plugin that is responsible for the used language. For example, if a policy is written in WS-Policy, there must be a corresponding WS-Policy plugin. All the language-specific logic is implemented by the plugins themselves, i. e., in the case of WS-Policy, operations such as intersection or normalization are up to the language plugin. The language plugin gets a reference to the policy to be checked, the whole topology, the candidate planlet's fragment, and a mapping of elements as input. The mapping defines which elements in the topology would correspond to which elements in the planlet's fragment if the policy can be fulfilled by the planlet. The plugins are free to interpret their policy language in any way. For example, if a certain language defines a key-value format for defining policy requirements, the plugin is allowed to compare these requirements with properties of the corresponding fragment node. If requirements and properties are compatible, the policy is fulfilled. Thus, there is no explicit need that a matching policy exists in the fragment at all.

3) Type Specific: This processing mode is the most specific one and bound to policy language and policy type, i. e., if there is a policy of type *Data Location Policy* written in WS-Policy, there must be a plugin registered for exactly that combination. Otherwise, the policy cannot be fulfilled. If such a plugin exists, the processing is equal to language specific: The plugin gets the same kind of information as input and decides if the planlet fulfills the policy's requirements. This processing mode enables a very specific processing of policies as the mode is bound to the policy type directly. For example, if a Data Location Policy with region "EU" (cf. Figure 7) is attached to a MySQL node that should be created and there are no planlets available in the system that have a compatible policy attached, the plugin may analyze the stack the MySQL database shall be hosted on and recognizes that the virtual machine below runs on Amazon's EC2 with region-property set to "EU". In this case, the policy would be fulfilled by the simple MySQL-Creation Planlet shown in Figure 4 that does not know anything about policies at all. This kind of processing enables complex logic which can be only known by a very specific type plugin.

E. Language and Type plugins

The processing mode attribute of a policy decides if a language or type specific plugin has to assess whether the policy can be fulfilled by a candidate planlet or not. Plugins may need to pass information about the matchmaking to the candidate planlet if it fulfills the policy's requirement, e. g., to configure it. Therefore, each plugin may return an XML-document and a list containing the policy ids of the planlet which have to be fulfilled as result that is passed to the planlet via its input message from the calling provisioning plan. This enables configuring if optional policies provided by the planlet have to be fulfilled, for example. This document is also linked with the id of the fragment's policy. Thus, the planlet is able to retrieve the policy language- or type-specific information.

F. Framework Architecture

In this section, we describe how the presented approach is implemented in the used Management Framework. Figure 9 shows the simplified architecture of the framework with the new integrated policy extension (gray background).

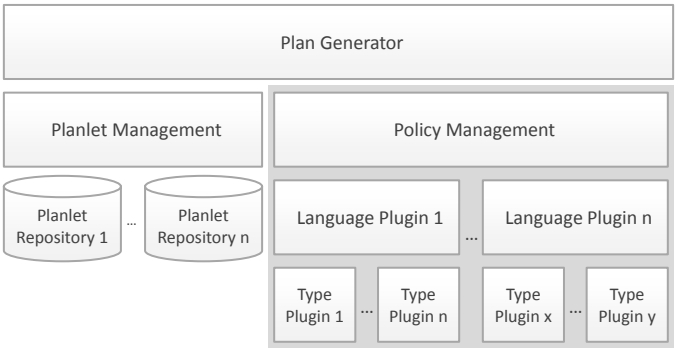


Figure 9. Extended framework architecture.

The basic architecture of the Management Framework substantially consists of a *Plan Generator* that uses a *Planlet Management* to retrieve the planlets and their descriptions stored in *Planlet Repositories*. The Plan Generator has a planlet orchestrator inside, which is responsible for scheduling planlets in the right order. We extend this orchestrator by the integration of a new component called *Policy Management* that is responsible for policy matchmaking and invoking *Language Plugins* and *Type Plugins*. Each planlet that is analyzed by the orchestrator gets now additionally checked if it fulfills the attached policies by a simple call to this new API. The integration is straight forward as the basic architecture of the Management Framework was built in a modular way.

G. Implementing Policy-aware Planlets—Lessons learned

In this section, we describe our experiences from the implementation of policy-aware planlets. A planlet providing additional non-functional capabilities expressed in the form of attached policies on elements contained in its fragment has to ensure that the semantics of the policies are only fulfilled if explicitly needed. This is important as the policy matchmaking is directed: only policies of the application topology are considered by the plan generator, not the policies of the planlet. Thus, if a planlet provides an extending policy, e. g., a Frequent Data Backup Policy, which exports data frequently from a database, this additional functionality should be installed only if needed. If the planlet is able to offer both modes, with and without fulfilling the policy, the planlet should declare this policy as optional. Therefore, the planlets get the mapping of elements in the topology to the elements in its fragment as input. Based on this mapping, the planlet is capable of recognizing if a policy is optional.

In many cases, the extension of planlets to policy-aware planlets is possible by only adding additional activities to the original planlet workflow—especially for Guarding and Extending Policies: The Frequent Data Backup Policy and the Safe Credentials Policy can be implemented by adding activities that install the additional software or check the chosen credentials. The actual process needs not to be modified. In

 ISBN: 978-1-61208-298-1

contrast to this, Configuring Policies often need to adapt the original process: for example, the Data Location Policy may influence the provisioning of a virtual machine. Thus, the activity that creates this VM must be modified.

VII. EVALUATION

In this section, we evaluate the presented approach. We prove the (i) feasibility, (ii) economics, (iii) analyze performance and complexity, (iv) describe our prototypical implementation, and describe (v) how the approach can be extended.

A. Feasibility

To prove the approach's feasibility, we evaluated the framework in terms of the three kinds of Provisioning Policies discussed in Section IV. We implemented planlets fulfilling the policy examples which were used throughout the paper. To prove the feasibility of Configuring Policies, we implemented the planlet described in Figure 8 that creates a virtual machine with an Ubuntu Linux operating system on Amazon EC2 complying with a Data Location Policy which defines that all data must remain in the European Union. The planlet extracts the region from the policy and uses this value as *region-attribute* for creating the VM using the Amazon Web Services API. This configures the provisioning in a way that the VM is hosted on physical servers located in states of the European Union. To prove the feasibility of Guarding Policies, we defined a Safe Credentials Policy attached to an Apache PHP Web server to ensure that username and password are strong enough. We implemented a planlet that provisions this Web server on Windows 7 complying with this policy by generating a safe password on its own. To prove the feasibility of Extending Policies, we implemented a Frequent Data Backup Policy which is attached to a MySQL database node. The corresponding planlet executes an additional bash script via cron job that frequently backups the data as MySQL dump to an external storage. This script execution may be seen as additional node hosted on the operating system. Thus, it extends the application structurally in order to fulfill a non-functional requirement. For all policy definitions, we used the properties-based policy language shown in the TOSCA specification [9].

The approach also enables defining complex non-functional security requirements that occur in real enterprise systems. This is enabled by the individual content field of Provisioning Policies. The field allows specifying any information about the policy language- or type, e. g., complex system configuration options and tuning parameters. As planlets that match such a policy are built to process exactly the information stored in the content field, the tight coupling of policies to the planlets processing them enables the implementation of any policy language and type. Thus, the corresponding planlets may deal with any individual policy-specific semantics or syntax. For example, if a security policy of type X specifies a set of complex system configuration files that must be taken into account during the provisioning of a certain component, the planlets complying with this policy type X expect these files and know how to process them. This enables to integrate expert knowledge about individual domains through defining own policy types and the corresponding planlets that deal with them.

B. Economics

The economic goal of our approach is to lower operating cost of provisioning. It is obvious that automating IT operations in order to reduce manual effort leads to a cost reduction in many cases. However, Brown and Hellerstein [10] analyzed the automation of operational processes and how this influences costs. They found out that three issues must be considered which counteract this reduction by causing additional effort: (i) Deploying and maintaining the automation environment, (ii) structured inputs must be created to use automation infrastructures, and (iii) potential errors in automated processes must be detected and recovered which is considerably more complicated than for manual processes. The presented approach tackles these issues. Planlets are reusable building blocks for the generation of provisioning plans. They are developed by expert users of several domains and provided to communities similar to DevOps. Therefore, free accessible planlet repositories enable continuous maintenance without the need for individual effort. Of course, maintaining local management infrastructure and the development of own custom planlets for special tasks causes additional effort, but this is a general problem that cannot be solved generically. The second issue of upfront-costs for creating structured input is reduced to a minimum as we provide tools for the modeling of application topologies and policies. The third issue of occurring errors is tackled implicitly by the workflow technology: every planlet defines its own error and compensation handling. Thus, errors are handled either locally by planlets themselves or by the generated provisioning plan, which triggers the compensation of all executed planlets to undo all operations for errors that cannot be handled.

C. Performance and Complexity

The performance of the approach is of vital importance as the generation of provisioning plans must be possible within a few seconds to obtain Cloud properties such as scalability or on-demand self-service. The employed Management Framework presented in Section V uses a partial order planning algorithm [11] for the generation of Management Plans [4]. As described by Bylander [12], the complexity of planning varies from polynomial to PSPACE-complete, depending on the preconditions, effects, and goals. The Management Framework tackles this issue by introducing restrictions on the design of planlets: it is forbidden to use multiple planlets providing the same or overlapping functionality but having different effects. For example, it is forbidden to have two planlets installing a MySQL database on an Ubuntu Linux operating system that differ in the set properties. This reduces the number of nondeterministic choices that have to be made during plan generation in terms of selecting planlets: if a Management Annotation can be processed by multiple different planlets, it does not matter which one is chosen because they have the same effects and may differ only in their preconditions. Thus, planlets having preconditions on the effects of these planlets are not affected by the plan generator's choice. This decreases the complexity to polynomial time [11]. Extending the framework by policies must follow this restriction: If a policy-aware planlet implements a functionality that is provided already by an existing planlet, the existing planlet has to be merged with the new policy-aware planlet. The new planlet must also support the original functionality which is trivial in most cases as policies only deal with non-functional requirements but do not change

 ISBN: 978-1-61208-298-1

the original functionality. The only difference is additional effort as calling plugins might be necessary. In worst case, each policy in a topology must be processed by a plugin. As the number *n* of used policies within a topology is constant, the extension has no influence on the complexity. We evaluated the performance of the policy-aware framework with the following setting: Based on a set of 25 fully-implemented real planlets, we developed different application topologies of different size - from small (5 nodes, 4 hostedOn relations) to large (100 nodes, more than 100 relations). We added 1000 randomly generated test planlets without workflow implementation to simulate a real environment. Thus, the plan generator has to consider 1025 planlets during plan generation. The generation of provisioning plans for all test application topologies (small to large) required only a few milliseconds. Serialization to BPEL and deployment on the used process engine (WSO2 Business Process Server 2.1.2) increased this by approx. 7 seconds. Thus, the time for generating plans is negligible in comparison to the time-consuming serialization to BPEL and the deployment.

D. Prototype

To validate the concept technically, we implemented the approach on the basis of the Management Framework prototype presented in Section V. This prototype is implemented in Java and uses OSGi in order to provide a flexible and dynamic plugin system. Planlets are implemented in the Business Process Execution Language (BPEL) whereas topologies and planlet fragments are implemented using an internal data model similar to the structure of TOSCA [9]. We extended this topology model with the possibility to attach policies to nodes and relations. The policies are provided as XML files following the simple properties-based policy language used in this paper. We use declarative OSGi services to build the plugin system for language- and type-plugins, as described in Section VI-F. To prove the technical feasibility of the conceptually evaluated policies described in Section VI we implemented these policies and others by extending already existing planlets. The successful implementation of this prototype extension proves that the presented approach works in reality.

E. Extensibility

As there are many different existing policy types and languages, the presented approach must support extensibility. The used Management Framework (cf. Section V) supports creating own custom planlets that implement Management Annotations for any conceivable management task. As the approach presented in this paper relies on this concept, it is possible to implement new policy types the same way. The plugin-based architecture for language and type plugins complements the planlet-based policy extension: If a new policy type needs a dedicated type plugin for advanced processing, the architecture allows installing new plugins that handle these types. In addition, the architecture enables the integration of any existing policy language as well as the development of own languages. We successfully validated this criterion based on the integration of WS-Policy. As WS-Policy has its own type system in the form of assertions, the type attribute of the Provisioning Policy is not needed. In addition, the created plugin retrieves the information about domain-specific processing of assertions by extracting the policy-specific content field which defines this kind of information.

VIII. Related Work

There are several works focusing on the automated provisioning of Cloud applications. In this section, we describe the most related ones and compare them to our approach. The work of Eilam et al. [13] focuses on deployment of applications by orchestrating low-level operation logic similarly to planlets by so-called *automation signatures*. El Maghraoui et al. [14] present a similar approach that orchestrates provisioning operations provided by existing provisioning platforms and is, thus, much more restricted than using planlets, which are able to integrate any technology and system. Both works do not consider non-functional requirements at all—especially not in the form of explicitly attached policies, which are able to define explicitly the tasks that must consider the policy. In contrast to both works, planlets and Management Annotations allow the application developer to bind policies to abstract tasks which are executed during provisioning. Thus, Policy-aware Management Planlets introduce an additional layer of abstraction in terms of defining and processing non-functional requirements.

Mietzner and Leymann [15] present an architecture for a generic provisioning infrastructure based on Web services and workflow technology that can be used by application providers to define provisioning flows for applications. These flows invoke so-called provisioning services that provision a certain component or resource. Policies can be used by the provisioning flow to select the specified provisioning services based on non-functional properties of the resource to be provisioned, e. g., availability of the provisioned resource. The general idea of implementing provisioning services is similar to planlets. However, planlets allow a much more fine grained differentiation between provisioning tasks, e. g., the provisioning of a database and the following initial data import are done by different planlets. Thus, policies can be bound more specifically to tasks and allow, therefore, a more precise definition of non-functional security requirements.

The Composite Application Framework (Cafe) [16] is an approach to describe configurable composite service-oriented Cloud applications that can be automatically provisioned across different providers. It allows expressing non-functional requirements in WS-Policy that can be matched to properties of resources in an environment. However, these policies are restricted to the selection of services and lack mechanisms to configure, guard, or extend application provisioning as enabled by our approach.

The CHAMPS System [17] focuses on Change Management to modify IT systems and resources by processing so-called *Requests For Change (RFC)* such as installation, upgrade, or configuration requests. After receiving an RFC, the system assesses the impact of the RFC on components and generates a so-called Task Graph which is afterwards used to generate an executable plan. The system can be used for initial provisioning of composite applications, too. Although the system's plan generator considers policies and SLAs, the work does not describe how the executed tasks have to process the artifacts during provisioning.

Kirschnick et al. [18] present a system architecture and framework that enables the provisioning of Cloud applications based on virtual infrastructure whereon the application components get deployed. However, the framework does not

 ISBN: 978-1-61208-298-1

support non-functional requirements, is tightly coupled to virtual machines, and lacks integrating any XaaS offerings. Thus, the system is not able to provision Cloud applications that consist of several XaaS offerings complying with non-functional security requirements.

The DevOps community provides tooling to automate the provisioning of Cloud applications. To mention the most important, Chef and Puppet are script-based frameworks used to automate the installation and configuration of software artifacts in distributed systems. The DevOps community also provides additional tooling such as Marionette Collective, ControlTier, and Capistrano used to improve the orchestration capabilities on a higher level. The frameworks are extensible in terms of adding new installation, configuration, and—in general—provisioning functionalities. This enables to integrate provisioning logic that considers non-functional requirements. However, all these frameworks focus on a deep technical level of provisioning and do not provide a means to express and integrate non-functional security requirements on such a high level as enabled by Management Annotations and planlets. The reusability in terms of provisioning different applications, the interoperability between script-based and non-script-based technologies as needed to build complex composite Cloud applications, and the holistic integration of different policy languages in order to achieve broad acceptance is not supported yet.

The Topology and Orchestration Specification for Cloud Applications (TOSCA) is an upcoming standard to describe composite Cloud applications and their management [9]. It tackles current challenges in Cloud computing such as portability and interoperability of Cloud applications, prevention of vendor lock-in, and the automated management of applications including provisioning [19]. TOSCA provides a XML-based format for describing Cloud applications as application topologies and enables the management of applications through Management Plans that capture management knowledge in an executable way. TOSCA provides a similar mechanism to attach policies to nodes and relations in topologies but, however, only provides a means to integrate policies into the topology but lacks a detailed description of their processing.

IX. Conclusion and Future Work

In this paper, we presented an approach that enables fully automated provisioning of complex composite Cloud applications in compliance with non-functional security requirements. Based on this approach, we extended an already existing Management Framework to support policy-aware provisioning. We introduced a new format for Provisioning Policies that are considered by Policy-aware Management Planlets during provisioning. In addition, the extended framework allows Cloud providers as well as application developers to implement their own policy-aware provisioning logic in a flexible and reusable manner independently from individual applications. The paper evaluates the presented approach in terms of performance, feasibility, and extensibility. In addition, we implemented a prototype that serves as a proof of concept of the presented conceptual work. In future work, we will extend this concept by a policy-aware preprocessing of topologies in order to increase the reusability of planlets to a higher level. Afterwards, we plan to apply the presented concept of policy-aware planlets also to runtime management of applications and extend the approach to support policies of other domains such as Green IT.

Acknowledgment

This work was partially funded by the BMWi project CloudCycle (01MD11023).

References

[1] D. Oppenheimer, A. Ganapathi, and D. A. Patterson, “Why do internet services fail, and what can be done about it?” in USITS, June 2003.

[2] F. Leymann, “Cloud Computing: The Next Revolution in IT,” in Proc. 52th Photogrammetric Week, September 2009, pp. 3–12.

[3] M. Armbrust et al., “Above the Clouds: A Berkeley View of Cloud Computing,” University of California, Berkeley, Tech. Rep., 2009.

[4] U. Breitenbücher, T. Binz, O. Kopp, and F. Leymann, “Pattern-based runtime management of composite cloud applications,” in CLOSER, Mai 2013, pp. 475–482.

[5] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, and D. Schumm, “Vino4TOSCA: A visual notation for application topologies based on tosca,” in CoopIS, September 2012, pp. 416–424.

[6] A. Keller and R. Badonnel, “Automating the provisioning of application services with the BPEL4WS workflow language,” in DSOM, November 2004, pp. 15–27.

[7] R. Wies, “Using a classification of management policies for policy specification and policy transformation,” in IFIP/IEEE IM, June 1995, pp. 44–56.

[8] W. Han and C. Lei, “Survey paper: A survey on policy languages in network and security management,” Comput. Netw., vol. 56, no. 1, January 2012, pp. 477–489.

[9] OASIS, Topology and Orchestration Specification for Cloud Applications Version 1.0, May 2013. [Online]. Available: http://docs.oasis-open.org/tosca/TOSCA/v1.0/cs02/TOSCA-v1.0-cs02.pdf

[10] A. B. Brown and J. L. Hellerstein, “Reducing the cost of it operations: is automation always the answer?” in HOTOS, June 2005, pp. 12–12.

[11] D. S. Weld, “An introduction to least commitment planning,” AI Magazine, vol. 15, no. 4, Winter 1994, pp. 27–61.

[12] T. Bylander, “Complexity results for planning,” in IJCAI, August 1991, pp. 274–279.

[13] T. Eilam, M. Elder, A. Konstantinou, and E. Snible, “Pattern-based composite application deployment,” in IFIP/IEEE International Symposium on Integrated Network Management, May 2011, pp. 217–224.

[14] K. El Maghraoui, A. Meghranjani, T. Eilam, M. Kalantar, and A. V. Konstantinou, “Model driven provisioning: bridging the gap between declarative object models and procedural provisioning tools,” in Middleware, November 2006, pp. 404–423.

[15] R. Mietzner and F. Leymann, “Towards provisioning the cloud: On the usage of multi-granularity flows and services to realize a unified provisioning infrastructure for saas applications,” in SERVICES, July 2008, pp. 3–10.

[16] R. Mietzner, T. Unger, and F. Leymann, “Cafe: A Generic Configurable Customizable Composite Cloud Application Framework,” in CoopIS, November 2009, pp. 357–364.

[17] A. Keller, J. L. Hellerstein, J. L. Wolf, K. L. Wu, and V. Krishnan, “The CHAMPS system: change management with planning and scheduling.” in NOMS, April 2004, pp. 395–408.

[18] J. Kirschnick, J. M. A. Calero, L. Wilcock, and N. Edwards, “Toward an architecture for the automated provisioning of cloud services,” Comm. Mag., vol. 48, no. 12, December 2010, pp. 124–131.

[19] T. Binz, G. Breiter, F. Leymann, and T. Spatzier, “Portable Cloud Services Using TOSCA,” IEEE Internet Computing, vol. 16, no. 03, May 2012, pp. 80–85.

 ISBN: 978-1-61208-298-1

Fighting Spam by Breaking the Economy of Advertising by Unsolicited Emails

Alexander Schmidtke, Hans-Joachim Hof
Munich IT Security Research Group (MuSe), Department of Computer Science and Mathematics
Munich University of Applied Sciences
Munich, Germany
e-mail: AlexanderSchmidtke@gmx.de, hof@hm.edu

***Abstract*—Unsolicited email (spam) is still a problem for users of the email service. Even though current email anti-spam solutions filter most spam emails, some still are delivered to the inbox of users. A special class of spam emails advertises websites, e.g., online dating sites or online pharmacies. The success rate of this kind of advertising is rather low, however, as sending an email does only involve minimal costs, even a very low success rate results in enough revenue such that this kind of advertising pays off. The anti-spam approach presented in this paper aims on increasing the costs for websites that are advertised by spam emails and on lowering the revenues from spam emails. Costs can be increased for a website by increasing traffic. Revenues can be decreased by making the website slow responding as some business gets lost. To increase costs and decreased revenues a decentralized peer-to-peer coordination mechanism is used to have mail clients to agree on a start date and time for an anti-spam campaign. During a campaign, all clients that received spam emails advertising a website send an opt-out request to this website. A huge number of opt-out requests results in increased traffic to this website and will likely result in a slower responsibility of the website. The coordination mechanism presented in this paper is based on a peer-to-peer mechanisms and a so-called paranoid trust model to avoid manipulation by spammers. A prototype implementation for the Thunderbird email client exist. The anti-spam approach presented in this paper breaks the economy of spam, hence, makes advertising by unsolicited emails unattractive.**

***Keywords*—spam; unsolicited email; peer-to-peer; security (key words)**

I. INTRODUCTION

According to [3], spam accounts for 64.1% of all emails. For this paper, spam refers to an unwanted email and spammer refers to the person sending this unwanted email. According to [1] spam is used for the follwing purposes: advertise services or products (e.g., pharmaceutical drugs, dating sites, etc.), distribute malware, scam

Anti-spam solutions targeting servers used by spammers (blacklists) are of limited use, as nowadays more than 80% of spam messages are sent with the help of botnets [2]. Hence, no central server is the source of spam but thousands of regular clients using thousands of legitimate email providers. Email filtering based on the content of spam emails [14] is an effective method to filter spam mails, however it can not be guaranteed that legitimate emails are not classified as spam. Also, spammers usually optimize their spam emails such that they pass Bayesian filters. In spam mails advertising a website, the URL is hard to obfuscate, because a user must have an easy way to go to the advertised website for the spam mail to be successful. According to [3], spam advertising sex sites, dating sites or sites selling pharmaceutical drugs make up for 86.52% of all spam. Hence, an anti-spam approach targeting this class of spam is highly relevant. The approach presented in this paper targets on websites advertised by spam. It aims on both increasing the cost to maintain the website as decreasing the revenue of the website. In [4], this is described as one way to stop spam as the success rate of spam is very low. Increasing the costs for the advertised website or decreasing the revenue of the website may lead to spam not being profitable any more. The approach presented in this paper increases the costs and decreases the revenue:

- Cost increase: clients send opt-out request to websites advertised by spam. These opt-out requests result in additional traffic to the website. If the owner of a website pays for traffic to his site, costs are increased to maintain the website.
- Revenue decrease: clients coordinate to send opt-out requests at nearly the same time, resulting in the website becoming slow in response. The website may loose some business because customers do not want to wait for the page to load. Sometimes spam websites are hosted on compromised hosts. Increasing the traffic to these hosts results in a higher probability that the administrator of the compromised host notices the infection of the host because legitimate services run slower than usual. It is very likely that the administrator then takes down the unwanted website, reducing the success rate for this website to 0%.

Multiple clients that received spam send opt-out requests in a so-called campaign. A campaign is defined by one target URL as well as a start date and time. Opt-out requests are sent after the start date and time. Clients participating in a campaign are called comrades. To participate in a campaign, a client must have received a spam mail advertising a website that leads to the URL of the campaign. Hence, it is ensured that clients only send opt-out requests for spam mails they really received. The result of a campaign is much traffic on a site as well as a slow responsiveness of the site. A coordination algorithm based on a peer-to-peer network is used to let comrades agree on a start date and time for their campaign. The coordination algorithm uses a so-called paranoid trust model to avoid manipulations by spammers. As long as the peer-to-peer network implements a Distributed Hash Table (DHT) offering

 ISBN: 978-1-61208-298-1

a PUT method to store information in the DHT and a PULL method to receive information from the DHT, any DHT can be used by the coordination algorithm. It is advised to use an existing DHT resilient to Distributed Denial of Service attacks, e.g., a file sharing peer-to-peer network with a huge user base like eMule [15] or other resilient peer-to-peer networks like NeighborhoodWatch [11].

It should be noted that the approach sketched may result in a Distributed Denial of Service attack on websites advertised by spam. It is an open legal question if sending an opt-out request using the proposed system is legal. This question can only be answered individually for each country and is out of scope of this technical paper. However, the existence of other opt-out anti-spam solutions (see Section II for a thorough discussion on related work) indicates, that countries exist where using the anti-spam solution proposed in this paper is legal.

Another problem are false positives: a spammer could try to abuse the proposed anti-spam solution by sending spam mails for an innocent website, e.g., to discredit the website of a competitor. A careful examination of emails is necessary to avoid false positives. Giving the user the possibility to check the website the anti-spam solution identified as potential target of a campaign may help to avoid false positives.

The rest of this paper is structured as follows: Section II discusses related work. Section III gives an overview of the system architecture. Section IV presents the distributed coordination mechanism of the proposed anti-spam solution. Section V introduced a paranoid trust model used for the presented approach. Section VI presents a prototype implementation for Thunderbird. Section VII evaluates the anti-spam approach presented in this paper under different attacks. Section VIII concludes the paper and gives an outlook on future work.

II. Related Work

This section presents related work on fighting spam by increasing the cost of spam and related work on using a DHT to fight spam.

Hashcash [8] [9] is used to target Denial of Service attacks. Hashcash is a challenge-response approach. In [8] [9], hashcash is used to increase the cost of sending email. To do so, when an email clients delivers an email to a mail server, the email server does not accept the email at once. Instead, the mail server provides a challenge to the mail client. The challenge requires some computational effort to be solved. The solution of the challenge is sent together with the mail. The mail server only accepts mails that include a solved challenge. Hence, the solved challenge can be seen as a virtual stamp. With this system, sending a single mail is still fast, but sending thousands of messages is significantly slowed, resulting in a lower spam per time rate increasing the cost of spam. However, there are some disadvantages:

- Hashcash helps email providers to throttle down the number of mails its users can sent. This helps an email provider to avoid abuse of its service. However, spammers often have an email provider of their own or directly deliver the message. Also, service for users of the email provider is limited.
- Hashcash only avoids sending a huge bulk of emails. According to [2], more than 80% of spam is sent with the help of a botnet. Hence, each zombie of a botnet has a much smaller bulk of emails to sent. Hashcash is not very effective to avoid spam sent by a botnet.

Some anti-spam approaches already use a DHT to fight spam, e.g., NeighborhoodWatch [11], a blacklist of IP addresses of known spammers. Although NeighborhoodWatch prevents Distributed Denial of Service attacks on the blacklist, it does not increase costs for spammers, hence, does not target the business model of spammers. However, the DHT used by NeighborhoodWatch can be used for the approach presented in this paper.

The startup company Blue Frog [12] offered a software that let email users send coordinated opt-out requests for spam they received similar to the approach presented in this paper. Blue Frog was very successful and forced six of the top ten bulk email groups to cooperate with them to remove users of the Blue Frog software from their email lists [12]. Hence, Blue Frog showed that a coordinated opt-out approach to fight spam can be very successful. Lycos [13] offered a similar approach with its ”Make Love not Spam” campaign. However, both Lycos and Blue Frog used a central coordination for the period of time the clients should send opt-out requests. This central coordination is vulnerable to attacks. In the case of Blue Frog, the central coordination mechanism was brought down by a massive Denial of Service attack. Another disadvantage of Blue Frog was the way users submitted received spam mails: users were asked to forward received spam mails. This resulted in users being blacklisted by mail providers that scan outgoing mails for spam mails. The approach presented in this paper follows the idea of coordinated opt-out requests. The success of Blue Frog and Lycos showed that the traffic generated by sending coordinated opt-out requests has a significant impact on the business model of spammers and websites advertised by spam. However, the approach presented in this paper tries to avoids the disadvantages of the solutions of Blue Frog and Lycos: no central coordination is used and users do not have to forward spam mails.

III. System Design

This section gives an overview of the proposed anti-spam solution. The following sections focus on one part of the system: the Campaign Coordinator. The Campaign Coordinator is the main difference to existing anti-spam solutions as discussed in Section II.

The proposed anti-spam solution has four components: Spam Classifier, Target Evaluator, Campaign Coordinator, and Opt-Out Module

The *Spam Classifier* decides which emails are spam and which are not. One possible implementation includes a user interaction that allows a user to express which emails are spam from his point of view. Involving the user for spam classification has the benefit, that only the user can decide what ”unwanted” mail means for him. Also, involving the user may help to avoid false positives. If not enough users decide that an email is spam, no campaign will take place. The Spam Classifier passes spam mails to the Target Evaluator. The *Target Evaluator* extracts one or more URLs from a spam

 ISBN: 978-1-61208-298-1

email. As a user must have a possibility to go to the website advertised by a spam mail, it is assumed that the identification of the URL of the website is possible. The Target Evaluator outputs URLs of the advertised website. A careful evaluation of the mail content is necessary because of the following reasons:

- Spam mails may use redirector services like TinyURL [16] for advertised website. The URLs of the redirection service may even be different for each spam mail. While getting a redirection URL already increases the cost of sending spam, this may be efficient to avoid spam campaigns because all URLs are different. To avoid this attack, the Target Evaluator looks for the use of redirection services.
- URLs in spam mails may contain parameters that identify emails of users. Hence, going to this URL may verify the email address of a receiver, potentially resulting in getting even more spam. Also, parameters of a URL may be different for each user. Hence, the Target Evaluator needs to strip parameters from URLs.
- Some email providers add footers to outgoing mail advertising their service. This footers usually include the URL of the mail server providers website, e.g., "This mail was sent by www.gmx.de". The Target Evaluator has a whitelist of URLs not to attack to avoids that such legitimate URLs become targets of anti-spam campaigns.
- An attacker could try to abuse the anti-spam solution by sending URLs of innocent websites in spam mails to provoke an attack on the innocent website.

It may be a good idea to involve the user in identifying the URLs to avoid false positives. The user decides on the websites to attack. The Target Evaluator is out of scope of this paper. The Target Evaluator passes one ore more URLs to the Campaign Coordinator.

The *Campaign Coordinator* is the heart of the anti-spam approach presented in this paper. The Campaign Coordinator uses a peer-to-peer network to identify other clients that also received a spam mail advertising the same URL. All clients that received this spam mail decide on a date and time when to start a campaign against the website with the given URL. The Campaign Coordinator is described in more detail in IV.

The *Opt-Out Module* is invoked at the date and time when a campaign starts. The module starts to send opt-out requests to the website. Please note that the Opt-Out Module does not depend on the availability of an opt-out link in the spam mail. One possible implementation of the attack module scans the website at the given URL for images and sends the opt-out request as a parameter in a GET request to all images. This opt-out request will show up in the log of the web server. Sending opt-out requests in this way can not even be stopped by using a captcha on the website to avoid automated opt-out requests. The opt-out requests result in traffic to the site, and the coordinated start of the campaign may result in a slower response of the website due to many requests. The Opt-Out Module is out of scope of the paper, successful opt-out anti-spam solutions like those discussed in Section II show that enough traffic can be generated to have an impact on the business model of websites advertised by spam.

Campaign Table	
www.buyviagra.com	01/09/13-11:30:00MEZ
www.buyviagra.com	03/09/13-08:00:00MEZ
www.buyviagra.com	03/09/13-01:00:00MEZ
www.date4u.com	30/08/13-04:31:00MEZ

Fig. 1. Example for a campaign table holding two campaigns.

Campaign Comrades Table	
01/09/13-11:30:00MEZwww.buyviagra.com	Public Key Client A
01/09/13-11:30:00MEZwww.buyviagra.com	Public Key Client B
03/09/13-08:00:00MEZwww.buyviagra.com	Public Key Client B
03/09/13-01:00:00MEZwww.buyviagra.com	Public Key Client C
30/08/13-04:31:00MEZwww.date4u.com	Public Key Client D

Fig. 2. Example for a campaign comrades table for Figure 1.

The rest of this paper focuses on the central component of the system: the Campaign Coordinator. The Campaign Coordinator is the main difference to existing anti-spam solutions as discussed in Section II.

IV. Design of the Campaign Coordinator

The Campaign Coordinator is the heart of the anti-spam approach presented in this paper. A campaign is identified by exactly one URL. If the Target Evaluator extracts more than one URL from a spam mail, it invokes the Campaign Coordinator once for each URL.

The Campaign Coordinator uses a DHT like Kademlia [5] or Chord [6]. Only PUT and GET options of these DHTs are used, hence, it is not necessary to start a new DHT but an existing DHT can be used. Many file sharing systems are based on DHTs. One of those could be used for the approach presented here. Reusing an existing DHT has the benefit of using a large peer-to-peer network that is not as vulnerable to attacks (e.g. Sybil attack) as a small network. Reusing an existing network may also make it easier to join the network [7].

Campaign Coordinators access three different tables stored in the DHT:

 ISBN: 978-1-61208-298-1

Inbox	
Public Key Client A	$E_{PK\ Client\ A}$(message 1 to A)
Public Key Client A	$E_{PK\ Client\ A}$(message 2 to A)
Public Key Client B	$E_{PK\ Client\ B}$(message to B)
Public Key Client C	$E_{PK\ Client\ C}$(message to C)

Fig. 3. Example for an inbox.

Campaign Table: the Campaign Table lists available campaigns. A campaign is identified by the URL of the website that is the target of this campaign. The database key to the Campaign Table is the URL, the hash value of the URL is used to access the DHT. The entries in the campaign table list one or more start times and start dates for this campaign. Figure 1 gives an example of a Campaign Table: one of the campaigns has multiple start dates and times.

Campaign Comrades Table: the Campaign Comrades Table lists all the clients that are willing to participate in a given campaign at a given time. Clients participating in a campaign are called comrades in the following. The database key for the Campaign Comrades Table is the start date and time of the campaign concatenated with the URL. The corresponding hash value is used to access the DHT. If multiple start dates and times exist for one campaign, there are multiple database keys for the Campaign Comrades Table. Figure 2 gives an example for a Campaign Comrades Table: there is one start date and time with two comrades, the other start dates and times only have one comrade. Client B decided to participate in two campaigns, the other clients only participate in one campaign. It should be noted that the concatenated start date and time and URL are not actually stored in the DHT. Instead, the hash value of the concatenated start date and time and URL is used as access database key to the DHT. Please note that the URLs in this example are only examples - it is not known to the author that any of these URLs have been advertised using spam mails.

Inbox: The inbox is used to allow clients to receive encrypted messages. The database key of this table is the public key of the client. The hash value of the public key of the client is used to access the DHT. A message to a client is sent by encrypting the message with the public key of the client and storing in the DHT using the hash value of the database key to access the DHT. A message is received when a client gets all entries in the DHT for the hash value of its private key. Read messages should be removed. Figure 3 shows an example of the Inbox table: Client A has two pending messages, Client B and C have one pending message. $E_{PK}(message)$ denotes that $message$ is encrypted using public key PK.

The configuration of the Campaign Coordinator uses the following parameters:

- Keying Material: a public key together with the associated private key. The public key is used as identity of this client in the Campaign Comrades Table as well as the address of his Inbox in the DHT.
- Acceptable Time Interval: The parameters max_wait and min_wait are used to decide on valid campaign start dates and times.
- Minimum Size of Campaign: $min\_comrades$, the minimum number of comrades in one campaign to participate in this campaign.
- Minimum Accumulated Trust Values: $min_accummulated_trust$, the minimum accumulated trust value of all comrades in a campaign that is necessary that a Campaign Coordinator takes place in a campaign. See Section V for details of the so-called paranoid trust model used for the proposed anti-spam solution.
- Usage Time: information on when the client is available and can participate in campaigns. Usage time is either manually defined by the user, or usage of the mail client and uptime of the system are monitored.
- Trust Database: a database holding trust information about other clients. See Section V for details of the so-called paranoid trust model used for the proposed anti-spam solution.

The Campaign Coordinator is invoked by the Target Evaluator. It receives one URL identifying one campaign. After invocation, the Campaign Coordinator follows the following procedure:

- Step 1: The Campaign Coordinator checks if there is already one or more entries for the given URL in the Campaign Table. It either selects one or more campaign start date and time from the DHT or decides to propose a better suited start date and time to other clients. If the Campaign Coordinator proposes another start date and time it stores the alternative start date and time in the Campaign Table. It uses the hash value of the URL to access the DHT. A campaign start date and time must meet the following conditions to be considered suitable for the client:
  - A campaign start date and time may not be more than max_wait minutes or less then min_wait minutes in the future. max_wait and min_wait are local settings of the Campaign Coordinator. This setup avoids that a spammer inserts a campaign start date and time into the Campaign Table that lies many years in the future, hence, no attack takes place, or repeatedly adds a campaign start date in the very near future to avoid that many clients participate in a campaign.
  - A campaign start date and time should be in a time period where the computer of the client is typically running. To do so, the Campaign Coordinator considers the typical usage times of the client. For example, if the Campaign Coordinator is implemented as a plugin for an email client, it will prefer the date and times a user typically uses the email client.

  A new entry is inserted into the Campaign Table if the Campaign Table has no entry for the URL yet or if the present start dates and times are not suitable.

 ISBN: 978-1-61208-298-1

If multiple campaign start dates exist and more than one of them are suitable for the client, the Campaign Coordinator either joins all suitable campaigns or decides to join the campaign with the highest trust value. See Section V for details about the trust model.

- Step 2: The Campaign Coordinator adds his public key for all selected start dates and times to the associated Campaign Comrades Tables. The database key for a given start time and date in the DHT is the hash value of the campaign start data concatenated with the URL of the campaign. Hence, different start dates result in different database keys.
- Step 3: The Campaign Coordinator adds the URL of the campaign to a local database together with the selected start dates and times. The database holds all currently active campaigns for this Campaign Coordinator. The processing of the URL ends.

The Campaign Coordinator regularly checks if one of the campaigns in his local database recently started. If so, the client again checks if he wants to take part in the campaign. To do so, it retrieves the information from the Comrades Tables again. At this point in time, the Comrades Table holds a list of all comrades that take place in this campaign. Comrades are identified by their public key. The Campaign Coordinator checks if more than $min\_comrades$ clients participate in the campaign. Only if the number of comrades is high enough, hence, the campaign will involve enough clients to be effective, the client participates in this campaign. Optionally, the client checks if the enough participating comrades are trusted. See Section V for details on the used trust model. If the Campaign Coordinator decides to participate in a campaign, it invokes the Opt-Out Module. The Opt-Out-Module sends the opt-out requests to the URL of the campaign.

It should be noted that all information about campaigns are public. Hence, a spammer may know about a campaign in the future targeting one or more websites advertised by him. The actions of the website administrator may include setting up firewall rules to block access for certain nodes to mitigate the attack as well as increasing server capacity. However, both actions involve additional costs. Hence, the goal of a campaign is still achieved.

V. TRUST MODEL

The Campaign Manager holds a database with trust values for known clients. Clients are identified by their public key. These public keys come from the Campaign Comrade Table, see Section IV for details. Several methods are used to establish trust:

Paranoid Trust Model: In the paranoid trust model, the client only believes what it sees. During a campaign, a client probes the responsiveness of the URL of the campaign. If the responsiveness decreases significantly the campaign is a success, the campaign manager increases the trust value of all comrades involved in this campaign. A client may even test the responsiveness of a website if it decided not to take part in the associated campaign. This helps to built up trust quickly. If a client detects that several campaigns did not succeed, it resets its trust database. This is an extra countermeasure to avoid manipulation of the trust database.

Trust by Challenge-Response: A challenge response-approach is used to establish trust with an unknown client. Trust is only established with clients that participate in one of the own campaigns. A client A wants to establish trust with an unknown client B. To do so, client A generates a challenge that requires some computational effort to solve it (see below for details of the generation of the challenge). Client A stores the response to the challenge together with the public key of client B. Client A generates a message for client B holding Client A's public key as well as the challenge. Client A encrypts the message with client B's public key. The public key of client B usually comes from the Campaign Comrade Table. Client A stores the message in the Inbox of client B. When the Campaign Manager of client B checks its Inbox, it decrypts the message using its private key. Client B solves the challenge. Client B generates a message holding the solution to the challenge as well as the public key of client B. Client B encrypts the message with the public key of client A and stores it in the Inbox of client A. When the Campaign Manager of client A checks its Inbox, it compares the response received to the response stored locally under the given public key. Trust by Challenge-Response makes it harder for an attacker to generate thousands of identities in the DHT. Significant computational resources are needed to generate many identities. Also, many identities do only allow to manipulate campaigns in small boundaries. It cannot be avoided that a campaign takes place. A spammer may have a botnet at hand that offers many computational resources. However, using a botnet to manipulate the proposed anti-spam solution also includes costs. Hence, the goal of the proposed anti-spam solution is still achieved.

Campaign Coordinators check on a regular basis if there are messages in their inbox. Campaign Coordinators only answer a given maximum of challenges to avoid Denial of Service Attacks.

To generate a challenge for client B, a client A concatenates its public key $PK_{ClientB}$ with two random numbers $rand1$ and $rand2$. $rand1$ and $rand2$ are taken from the intervall $[0, max\_rand]$ where max_rand is a parameter for configuration of the time needed to solve the challenge. A hash value h is calculated using the hash function $hf()$:

$$h = hf(PK_{Client_B}||rand1||rand2) \quad (1)$$

The challenge for client B is to find out which value $rand2$ was used in the calculation of h. The challenge sent to client B is $rand1$. $PK_{ClientA}$ is known from the message as it identifies a client and is needed to send a message to client A.

To solve the challenge, client B follows the following procedure:

1) Initialize $test$ with 0
2) Calculate $h' = hf(PK_{ClientA}||rand1||test)$
3) If $h'! = h$ increase $test$ by 1 and go to step 2
4) $test$ is the solution of the challenge.

The challenge includes the public key of client A to avoid that an attacker forwards received challenges to a legitimate client that solves the challenge and sends it to the attacker that sends the response back to the client. This man-in-the-middle

 ISBN: 978-1-61208-298-1

attack is prevented by the inclusion of the senders public key in the challenge.

The trust values from the local trust database of the Campaign Coordinator can be used to decide if a Campaign Coordinator participates in a campaign or not. The Campaign Coordinator only participates if the sum of all trust values of the comrades of this campaign is higher than $min_accummulated_trust$. $min_accummulated_trust$ is a configuration parameter of the campaign manager. To allow for a quick start, $min_accummulated_trust$ starts at a low level and is increased with each successful campaign.

VI. Prototype Implementation

The Campaign Coordination described in Section IV is agnostic to the underlaying peer-to-peer network. For the implementation on Windows 7 the Campaign Coordinator relies on Overlay Weaver [10], however changing the DHT only requires minimal changes. Target Evaluator, Campaign Coordinator, and Opt-Out Module are implemented as a program running in the system tray. This program communicates with the Spam Classifier implemented as Addon for the Thunderbird mail client. The Spam Classifier is a simple button for the user to identify spam in this prototype implementation. A more advanced Spam Classifier will be part of further work. The separation of the Spam Classifier from the other components of the anti-spam solution allows an easy adaption of the implementation for different email clients as only the Spam Classifier must be adapted to different mail clients.

VII. Discussion of Attacks

This chapter evaluates the outcome of attacks on the Campaign Coordinator.

One possible attack is a *Man-in-the-Middle Attack* on the trust establishment. An adversary can forward challenges received by clients to built trust to other clients to solve it for them. However, this attack is not possible because the public key of the sender of a challenge is included in the challenge itself.

Another class of attacks are *Denial of Service Attacks*. An adversary can generate many challenges for one client. If this client wants to establish trust, it will solve all the challenges, doing many computations. This attack is not successful because clients only solve challenges sent by clients that participate in one of their campaigns and the number of challenges to solve is limited.

In a *Time Portal Attack* an adversary tries to move a campaign in the very distant future such that the attack never takes place. To do so, the adversary injects a campaign for a URL into the Campaign Table that has a start date and time that lies years in the future. Clients joining this campaign will never send opt-out requests. In a *Separation Attack* an adversary tries to reduce the number of comrades for a URL. This is done by repeatedly injecting campaigns for the URL with a very early start date and time. In this case, campaigns do not have the chance to get many comrades, hence, the Opt-Out Modules may decide not to send opt-out requests because the number of comrades is not high enough. Both the Time Portal Attack and the Separation Attack are avoided by the way a Campaign Coordinator checks suitable start dates and times. The attacks are further mitigated by the possibility to participate in more than one campaign.

VIII. Conclusion and Future Work

The anti-spam approach presented in this paper increases the costs of websites advertised by spam and decreases their revenue, hence, making advertising by unsolicited email unattractive. Contrary to existing solutions, a decentralized approach using a DHT has been chosen to avoid attacks. The anti-spam approach offers protection against attacks like Man-in-the-Middle Attacks, Denial of Service Attacks, Time Portal Attacks as well as Separation Attacks. An implementation of the anti-spam solution for the Thunderbird mail client exists. Future work will target on the Opt-Out Module and the Target Evaluator.

References

[1] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna, "The underground economy of spam: A botmaster's perspective of coordinating large-scale spam campaigns", LEET'11 Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats, 2011, pp. 4–4.

[2] Symantec, "Spam and fraud activity trends", http://www.symantec.com/threatreport/topic.jsp?id=spam_fraud_activity_trends&aid=analysis_of_spam_delivered_by_botnets, 2011, [retrieved: 06, 2013].

[3] Symantec, "Symantec intelligence report: january 2013", http://www.symantec.com/content/en/us/enterprise/other_resources/b-intelligence_report_01-2013.en-us.pdf, 2013, [retrieved: 06, 2013].

[4] M. Ilger, J. Strauss, and W. Gansterer, "The economy of spam", Technical Report FA384018-6, Institute of Distributed and Multimedia Systems, Faculty of Computer Science, University of Vienna, Vol. 9, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.85.536&rep=rep1&type=pdf, 2006, [retrieved: 06, 2013].

[5] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric", Peer-to-Peer Systems, Springer, 2002, pp. 53–65.

[6] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for internet applications", ACM SIGCOMM Computer Communication Review, vol. 31, no. 4, 2001, pp. 149–160.

[7] M. Conrad and H.-J. Hof, "A generic, self-organizing, and distributed bootstrap service for peer-to-peer networks", Self-Organizing Systems, Springer, 2007, pp. 59–72.

[8] A. Back, "Hashcash", http://www.cypherspace.org/hashcash, 1997, [retrieved: 06, 2013].

[9] A. Back, "Hashcash-a denial of service counter-measure", http://hashcash.org/hashcash.pdf, 2002, [retrieved: 06, 2013].

[10] K. Shudo, T. Yoshio, and S. Satoshi "Overlay weaver: An overlay construction toolkit", Computer Communications, vol. 31, no. 2, 2008, pp. 402-412.

[11] A. Bender, R. Sherwood, D. Monner, N. Goergen, N. Spring, and B. Bhattacharjee, "Fighting spam with the NeighborhoodWatch DHT", Proc. INFOCOM 2009, IEEE, 2009, pp. 1755–1763.

[12] R. Lemos, "Blue Security folds under spammer's wrath", http://www.securityfocus.com/news/11392, 2006, [retrieved: 06, 2013].

[13] BBC, "Screensaver tackles spam websites", http://news.bbc.co.uk/1/hi/technology/4051553.stm, 2004, [retrieved: 06, 2013].

[14] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian approach to filtering junk e-mail", AAAI'98 Workshop on Learning for Text Categorization, 1998, pp. 98–105.

[15] "The eMule Project" www.emule-project.net/, [retrieved: 06, 2013].

[16] "Tiny URL", http://tinyurl.com/, [retrieved: 06, 2013].

 ISBN: 978-1-61208-298-1

Smurf Security Defense Mechanism with Split-protocol

Harold Ramcharan
Department of Computer and Information Sciences
Shaw University
Raleigh, NC, USA
hramcharan@shawu.edu

Bharat Rawal
Department of Computer and Information Sciences
Shaw University
Raleigh, NC, USA
brawal@shawu.edu

***Abstract*—Network intrusion has been a difficult problem to solve due to the rapid growth of the Internet in recent years. Securing computers from harmful attacks are becoming the unprecedented challenging issues for internet users. Every day the recognition of new attacks is becoming a harder problem to crack in the field of Computer Network Security. Currently, Denial of Service (DoS) attacks is affecting the large number of computers in the world on a daily basis. Detecting and preventing computers from DoS attacks is a major research topic for researchers throughout the world. The migratory nature and role changeover abilities of servers in Split-protocol avoid bottleneck on the server side. It also offers the unique ability to avoid server saturation and compromise from DoS attacks. The goal of this paper is to present the idea of Split-protocol as a protection technique against DoS attacks.**

Keywords-Split Protocol; Protocol splitting; DoS; Tribal Flood Network; Bare Machine Computing.

I. INTRODUCTION

Overloaded serves are always at a higher risk for security compromise. The Split-protocol [1] offers a mechanism for server change over without involving clients. For example, as shown in Figure 1, a client on the network sends a request through the Connection Server (CS). This request will then be forwarded to the Data Server (DS), which in turn sends the requested data to the client. The symmetrical structure of CS and DS allows changing roles dynamically.

Should DS1 server crash, DS2 server will take the IP of DS1 and all of its data will be relinquished to DS2. Whenever DS1 is overloaded (CPU is around 96%), DS1 will shutdown as DS* takes over (DS* is back up to DS1, such as DS2, DS3…). By toggling between DS1 and DS*, one can avoid saturation of the server. A detail mechanism is explained in Section II. This mechanism is similar to the mobile defense mechanism [24].

Protocol splitting enables TCP to be split into its connection and data phases, so that these phases are executed on different machines during a single HTTP request [1]. In the basic form of splitting, the state of the TCP connection to the original server is transferred to a Data Server after receiving the HTTP Get request with no client involvement. The Data Server then transfers the data to the client, and connection closing can be handled by either the original server or the Data Server. Many variations on basic TCP/HTTP splitting are possible and have been used to improve Web server performance by use of delegation [1], split mini-clusters [2], and split architectures [3]. The security and addressing issues that arise due to protocol splitting can be solved in a variety of ways. The simplest solution is to deploy the servers in the same subnet or in the same Local Area Network (LAN) if host-specific routes are supported. The latter is used in this paper for testing migration performance by splitting. More generally, splitting can be applied to protocols other than TCP/HTTP by identifying protocol phases that are amenable to splitting. In this paper, we adapt TCP/HTTP splitting to devise a novel technique for Web server migration. It enables an alternate Connection Server to

dynamically take over active TCP connections and pending HTTP requests from the original Connection Server upon receiving a special inter-server message from it. Migration based on splitting can be used to improve Web server reliability with only a small penalty in performance. Additional benefits of splitting such as Data Server anonymity and load sharing can also be achieved with this approach to migration. We first implement Web server migration using split bare PC Web servers [1] that run the server applications with no operating system or kernel support. We, then, conduct preliminary tests to evaluate performance with migration in a test LAN where the split bare PC servers are located on different subnets. Protocol splitting is especially convenient to implement on bare machine computing systems due to their intertwining of protocols and tasks. However, the migration technique based on splitting is general, and can be implemented using conventional servers that require an operating system or kernel to run [4].

The rest of the paper is organized as follows. Section II discusses related work. Section III describes the Web server migration, and its design and implementation. Section IV describes a Smurf attack. Section V discusses possible ways to address these attacks. Section VI presents the design and the implementation of the proposal. Section VII contains the conclusion.

II. RELATED WORK

When an increasing number of users (or processes) accessing a website is beyond the tolerable threshold, the performance of the web server decreases. This is due to higher CPU utilization rates, thereby resulting in a greater

 ISBN: 978-1-61208-298-1

response time. Furthermore, as the response time increases, the ratio of the users accessing the site will also decrease. This higher CPU utilization can occur due to intruders launching deliberate attacks. These unwanted users use unnecessary data and techniques to occupy most of the server's bandwidth, degrading the server performance, thus, rendering the site useless. Kuppusamy and Malathi [6], implemented a particular technique to detect and prevent both individual Denial of Service (DoS) attacks [8], as well as Distributed Denial of Services (DDoS) attacks [6]. DDoS occurs when a multitude of distributed attack is launched against a single site or server, as opposed to a single user staging direct attacks. In response to mitigating the effects of spoofing IP source addresses, as is common in DoS attacks where packets lack a verifiable IP source address, the unicast reverse path forwarding (uRPF) [7] is a valuable tool for this purpose. It requires that a packet be forwarded only when the source addresses are valid and consistent with the IP routing table, or ensuring that the interface that the packet arrives on is matches the same used by the router to reach the source IP of the packet. If the interface does not match, then the packet will be dropped.

In Hop-Count Filtering (HCF) [8], each end-system maintains a mapping IP address aggregates and valid hop counts from the origin to the end system. Packets arriving at destination with significant variation in hop counts are considered unreliable and are either discarded or flagged. Li et al. [9] described SAVE, a mechanism for propagating only valid prefixes along the same paths that data packets will follow. By using the prefix and path information, routers can thus construct the appropriate filtering mechanism along the paths. Bremler-Barr and Levy proposed a Spoofing Prevention Method (SPM) [10], where packets are exchanged using an authentication key affiliated with the source and destination domains. Nowadays, there is an ever growing threat of intruders to launch attacks utilizing both-nets [11]. In this case, since the attacks are carried out through compromised intermediaries, often termed bots, it is difficult to discover the initiator of the attacks. However, current trends indicate that IP spoofing still persists [12] [13]. Man-in-the-Middle attacks (MitM), is a variant of TCP hijacking, as well as DNS poisoning [14] [15], and are carried out by the attacker masquerading as the host at the other end of the communication. IP spoofing attack is a hijacking technique in which an attacker masquerades as a trusted host to hide his identity [21].

III. MIGRATION WITH SPLIT-PROTOCOL

A. Overview

Split protocols require a minimum of two servers, i.e., a Connection Server and a Data Server. The CS establishes the connection via SYNs and ACKs. When the HTTP Get is received by the CS, it sends an ACK to the client, and uses an inter-server packet message referred to as a Delegate Message (DM). The DM1 is used to transfer the TCP state to the DS, which sends the data to the client. In bare PC servers, the TCP state and other attributes of a request are contained in an entry in the TCP table (known as a TCB entry). The CS also handles the TCP ACKs for the data and the connection closing via FINs and ACKs. Typically, the CS has information about the requested file (i.e., its name, size, and other attributes), and the DS has the actual file (the CS may or may not have a copy). When the DS gets DM1, it creates its own TCB entry and starts processing the request. When a DS sends data to the client, it uses the CS's IP address. After the CS receives the FIN-ACK, it sends another inter-server packet DM2 to DS. The receipt of DM2 closes the state of the request in the DS. More details of protocol splitting are given in a Split-protocol technique for Web Server Migration [5]. For Web server migration, inter server packet would be sent with a special massage, indicating that the CS is going to crash, and the TCB entry moved from one CS to another CS (called CS* for convenience), enabling the latter to take over the connection. Migrating server content in this manner and requiring that CS and CS* use the same IP address for two-way communication, poses a new challenge: now CS* must be able to send and receive packets with the IP of CS, which has a different prefix. Furthermore, the client must remain unaware that migration or protocol splitting has occurred. The main focus of this work is to address these issues and migrate (or transfer) a client connection to a new server, when the current connection server detects that it is going down or is being taken down. The means by which the server might detect its imminent failure is beyond the scope of this paper.

B. Design and Implementation of Role Change

Before the CS shuts down, it must send all of its pending requests to its alternate CS*. We assume that CS* is connected to the network, but that it will not process any normal requests (i.e., it is in stand-by mode). Also, CS and CS* are able to communicate with each other. Prior to the connection transfer, inter-server packets are being sent from CS to the DS according to the usual protocol splitting [6] when GET requests arrive. Under large load conditions, it is possible that CS could have many unprocessed requests in its TCP table. In addition to these pending requests, new requests may still continue to be sent by the client during the time between when CS shuts down and CS* takes over. These requests will be lost and will be processed later by CS* when the client retransmits them. Before CS shuts down, it also sends a final inter-server packet to CS* to confirm it is shutting down. Only minimal modifications had to be made to the current split server and inter-server packet format to implement the migration.

Alternatively, the DS can also assume the role of CS* (instead of using a separate CS*) if CS sends its pending requests to DS. If DS has some of its previous data transfer requests still to be processed, it will complete them before it begins to act as CS*. Protocol splitting is designed so that the same server can provide services as a CS and/or a DS; so, it is capable of assuming the role of CS* to implement the migration.

 ISBN: 978-1-61208-298-1

CSs in Split-protocol do not reserve resources for all requests it receives; this increases the capacity of servers in handling many folds of higher load than conventional servers. Also, the self-delegating mechanism in the splitting protocol allows the server to deny accepting any additional request to process, and changes his identity (IP) within a single TCP connection. Even if it changes its identity, it will still continue to serve that old request already in the queue (receiving system), until completion or reset. While original server (CS) is recovering, a new server (CS*) who has replaced it, will manage new requests. When it reaches a saturation point, it will also change its identity (IP address), and this time, the original server will handle all new incoming traffic. Toggling the same IP address between multiple servers will minimize the incoming load on Split-servers [5].

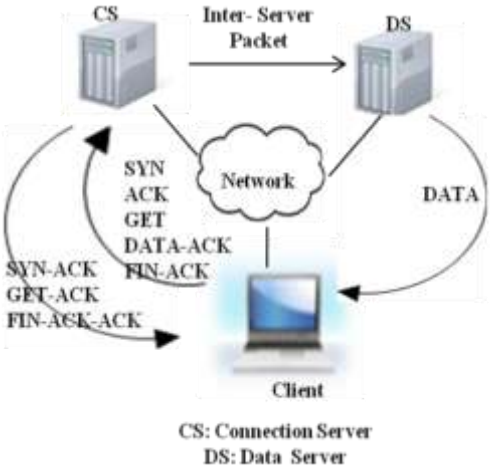


Figure 1. Split Architecture

IV. SMURF OR FRAGGLE

Smurf attacks can be considered one of the most overwhelming of the DoS attacks. In the Smurf [Internet Control Message Protocol (ICMP) Packet Magnification] attack [24], the attacker sends an ICMP echo request (ping) to a broadcast address with a spoofed source address. This source address is that of the victim's IP address. All the machines, when responding to the echo request, will flood the victim's system with their ICMP echo replies. As the flooding continues, this will ultimately results in the victim system crashing or freezing.

Smurf attack targets all available network bandwidth by consuming it with the intention to disrupt system's resources through bandwidth amplification. On a multi-access broadcast network, hundreds of systems could be responding to each packet sent, which could flood and render the victim's system useless even though using a much higher bandwidth type system [22]. The cousin of the Smurf's attack is the Fraggle attack, which uses the UDP echo packets in place of the ICMP echo packets in the same manner [35].

The recent rise in DoS attacks targeting high-profile web sites shows how overpowering these attacks are and how unprotected the Internet is under such attacks [28]. We present a survey of the up-to-date defense strategies against Denial of Service (DoS) attacks that gives hope in this area. We also present the weaknesses of the available methods pointing to the fact that no distinct adopted method has been in place. Also, future trends in DoS defense mechanism are discussed. The primary targets for these attacks are Web servers own by banks, online gaming websites, credit card payment gateways, domain name servers (DNS), E-commerce application tools, and Voice-over-IP (VoIP) services by prohibiting customer's access, or limiting access to resources such as bandwidth or degrading usage of these applications [21] [23].

The commonly employed attacks are:

A. Flooding

Flooding is the most basic method aimed to cripple a network by overwhelming it with large amounts of traffic directed to the victim. This utilizes all of the system resources [23] where the victim, in this case, can be a single PC or a high profile web server. The severity of such an attack depends more on the volume of traffic rather than the contents of the attack traffic.

B. Malware

Malware is malicious software used or programmed by attackers designed to overwhelm the system thereby allowing them unauthorized access. They will then have the capability to perform malicious operations. Known types of Malware includes: viruses, Trojan horse, adware and spyware, root kits, etc. The intended benefits for the perpetrator writing malware can range from financial gain to vengeance or for fun in seeing how fast and effective it can spread [24] [33]. These attacks exploit flaws in software vulnerabilities, such as in windows operating system or web server defects, cause these systems to reboot, crash or impede the system performance [25].

C. DoS attacks

Generally, DoS attacks can be categorized into two forms: (1) those that flood services affecting bandwidth, and (2) those that crash services by consuming resources [26]. Figure 2 describes different methods of DoS attacks. These DoS attacks become amplified when sent from unknown & unlimited sources termed Distributed Denial of service attack (DDoS) and usually occur in two phases, the recruitment phase and the actual attack [27] [28].

 ISBN: 978-1-61208-298-1

Method	Types
Protocol Based Attack	ICMP Flood, SYN Flood
Application Based Attack	HTTP Flood , SIP Flood
Distributed Reflector Attack	DNS Amplification Attack
Infrastructure Attack	

Figure 2. Methods of attack [28]

V. DEFENSE MECHANISM

From the standpoint of DoS, it is very difficult to completely remove the risk associated with such attacks. However, risk mitigation can be implemented through Avoid-Detect-Prevent cycle, as described here.

A. Avoid

Avoidance is an essential part of any defensive strategy even though many web sites choose to ignore it. Attacks can best be studied through collecting technical data such as network topology, Internet Service Provider (ISP) vendor agreements, insurance policy coverage, etc. However, from a risk management standpoint regarding DoS defense system, the need to identify and label critical services versus non-critical ones is important. The same apply for the corresponding vendors providing those services on the network. It is also important to have discussion with management, knowledgeable technical staff, service vendors, and law enforcement

B. Design network or system for survivability

This refers to the separation of critical services from non critical ones.

C. Monitoring

Prior to implementing monitoring procedures, special attention should be focused on target resources should an attack occur. Monitoring however, can be performed at two distinct levels; (a) at the network level, and (b) at the host level. Risks from DoS attacks can be reduced through the creation of effective incident response plans, establishing a sound partnership with service providers (vendor), and firewalls as intrusion prevention systems [26].

D. Detect

Modern networks can be very complex and diverse, therefore, an effective detection system is valuable to detect, prevent, and alert personnel of any DoS attacks in real time. Detecting an attack before becoming full scale can be vital to an organization's security posture. Modern Intrusion Detection Prevention Systems (IDPS) come equipped to combat these attacks and maintain state [24]. Detection systems should provide multiple detection mechanism, alerts, response mechanisms [25], and short detection time with low false positive rate [24]. These intrusion detection systems can take several forms such as anomaly detection, signature-based detection, and DoS attack detection, as discussed below [20].

E. Signature-based detection

This is simply searching network traffic and looking for a packet or series of bytes (signatures), which is considered malicious codes and comparing it to a set of attack signatures in order to detect the presence of an attack. A database of known signatures is usually developed by antivirus vendors for detecting known signatures [20]. This technique is also used by Snort (an IDPS), as it can perform real–time packet content searching and matching [19]. Snort and other IDPSs have one major weakness; they may take some time for a new exploit to become known. Later, after this new attack is known, a new signature can be developed and implemented. But, until then, well-defined signatures may go undetected [27].

F. Anomaly-based detection

This detection mechanism focuses on examining network traffic and comparing it with an established baseline [19], and is characterized by a set of pre-programmed thresholds [26]. This includes statistical approaches together with varying techniques such as those adapted from machine learning Models and Algorithms [17]. Neural Networks [31] and Bayesian Learning [32] can also be applied.

G. DoS-attack-specific detection

DoS attack traffic is instituted by the attacker as his objective is to direct maximum traffic to launch a powerful attack and may generate random patterns to make an attack signature undetected.

H. Prevent

Attack prevention measures aim to detect and prevent attacks before becoming full scale. Distributed packet filtering is possible through local routing information in order to prevent severe flooding attacks [24]; thus, a reaction and alert mechanism must be instituted to minimize the loss potential. This response mechanism should be effective in providing early detection automatically, dodging network overloading, and localizing the attack source with trace back techniques [18] [19], or mitigating the propensity of the attack [16] by denying unwanted packets.

I. Reaction

Reaction methods include effective incidence response plan, efficient backup systems, and filtering excessive traffic.

There are several mitigation techniques implemented for DoS and DDoS attacks. Avi Chesla [30] introduces an anomalous pattern for an HTTP flood protection. In this procedure, mitigation is controlled through a feedback mechanism that tunes a level of rate limiting factors. This is required for mitigating the attack effectively while allowing legitimate traffic to pass. A reliable trigger for an automated response system may be difficult to implement. Specht and Lee's [30] mitigation technique is based on similarities and patterns in different DDoS attacks. DDoS attack tools are normally designed to be friendly with different Operating

 ISBN: 978-1-61208-298-1

Systems (OS). Any OS system (such as UNIX, Linux, Solaris, or Windows) may have DDoS agents or handler code designed to work on it. Normally, a handler code is intended to support an OS that would be positioned on a server or terminal at either a corporate or ISP site. Most of the proposed mitigation mechanisms are also OS dependent. Split-protocol implementation on Bare Machine Computing (BMC) paradigm [4] does not use any kind of operating system. So, practically it is impossible to attack any BMC based system. On BMC, any DDoS agent or handler code designed for an OS cannot run. BMC codes are self-content. In addition, extra codes on existing applications or processes would not be allowed to run on the BMC system.

VI. DESIGN AND IMPLEMENTION

Split-protocol client server architecture design and implementation differ from traditional client server designs. As the traditional client server architecture is modified in this approach, we have designed and implemented a client server based on a bare PC, where there is no traditional OS or kernel running on the machine. This made our design simpler and easier to make modifications to conventional protocol implementations. Figure 3 shows a high level design structure of a client server architecture in a bare PC design. Each client and a server consist of a TCP state table (TCB), which consists of the state of each request. Each TCB entry is made unique by using a hash table with key values of IP address and a port number. The CS and DS TCB table entries are referred by IP3 and Port#. The Port# in each case is the port number of the request initiated by a client. Similarly, the TCB entry in the client is referenced by IP1 and Port#.

The TCB tables form the key system component in the client server designs. A given entry in this table maintains complete state and data information for a given request. This entry requires about 160 bytes of relevant information and another 160 bytes of trace information that can be used for traces, error, log, and miscellaneous control. This entry information is independent of its computer and can be easily migrated to another PC to run at a remote location. This approach is not the same as process migration [5], as there is no process information contained in the entry. The inter-server packet is based on this entry to be shipped to a DS when a GET message arrives from the client. Notice that the client uses IP1 and Port# to address the TCB entry. That means, when DS sends data or other packets, then it must use IP1 as its source address and its own MAC address in the packet. However, a client must be aware of IP1 and IP2 addresses to communicate to two servers for different purposes. The client knows IP1 through its own request and by resolving the server's domain name.

The client does not know IP2 address to communicate during the data transmission. We solved this problem by including the IP2 address in the HTTP header using a special field in the header format. In this design, a client could get data from any unknown DS and it can learn the Data Server's IP address from its first received data (i.e., header). This mechanism simplifies the design and implementation of Split-protocol client server architecture. This technique also allows the CS to distribute its load to DSs based on their CPU utilization without implementing a complex load balancing technique [1]. With implementing limited ACKs, the linear performance improvement continues up to 4 DSs [3]. This is also expected as CS poses no bottleneck for 4 DSs. For limited ACKs, the number of DSs connected to a single CS can be estimated to be 13 by extrapolating the CS CPU time and the number of DSs.

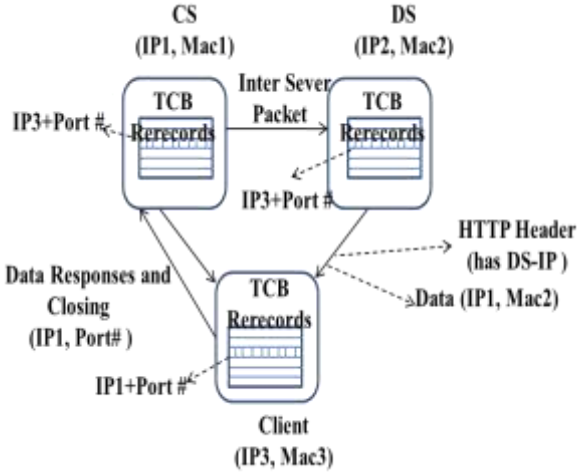


Figure 3. Design Structure

VII. CONCLUSION

Connection server in Split-protocol technique does not reserve any resource for all requests it receives, therefore it can handle many connection requests. In our empirical data, it suggests that CS only reserve 1% of CPU cycles compared to 95% for DS. Since there are many DSs in the system, they can handle very large loads without compromising services.

Also, the self-delegating mechanism in the split-protocol allows the server to deny accepting any additional request to process, and changes his identity within a single TCP connection. As shown in Figure 4, toggling the same IP address between multiple servers minimizes the incoming load on Split-servers. In multiple ways, both the Smurf attack and the Fraggle attack involves the attacker, the intermediary, and the victim.

 ISBN: 978-1-61208-298-1

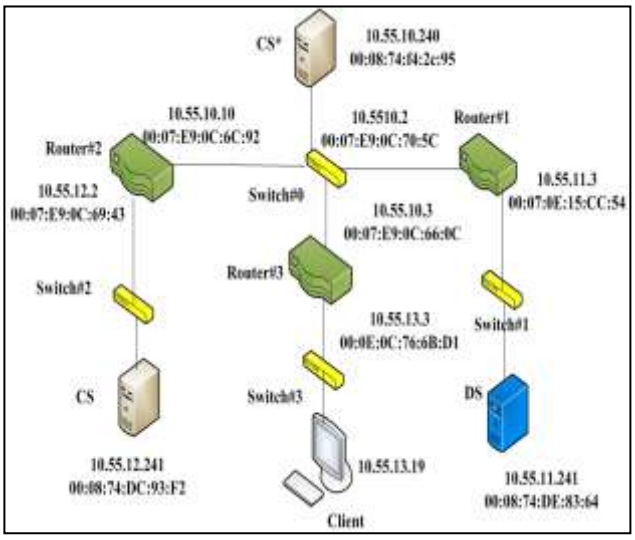


Figure 4. Role Change Over

Normally, both the intermediary and victim of this attack may suffer degraded network performance either on their internal network or on their connection to the Internet. Performance may be degraded to the point that the network cannot be used. Most of the time, the attacker identifies the underlying operating system from data structure of communication packets, which can further maximize the attack. Protocol-splitting, in our study, hides the underlying operating system thereby making it more difficult for Smurf attacker to circumvent. Furthermore, implementing protocol-splitting on BMC makes it harder to run a DDoS agent or handler code designed to work on operating systems. The anonymous nature of Data Server and migratory capability within single connections of Split-protocol architecture offers strong defensive mechanism against Smurf attacks.

REFERENCES

[1] B. Rawal, R. Karne, and A. L. Wijesinha. "Splitting HTTP Requests on Two Servers," The Third International Conference on Communication Systems and Networks: COMPSNETS 2011, January 2011, Bangalore, India.

[2] B. Rawal, R. Karne, and A. L. Wijesinha. " Mini Web Server Clusters for HTTP Request Split," 13th International Conference on High performance Computing and Communication, HPCC-2011, Banff, Canada, Sept 2-4, 2011.

[3] B. Rawal, R. Karne, and A. L. Wijesinha. "Split Protocol Client/Server Architecture," The 17th IEEE Symposium on Computers and Communications - ISCC 2012, July 1-4, 2012, Cappadocia, Turkey.

[4] L. He, R. K. Karne, and A. L. Wijesinha, "The Design and Performance of a Bare PC Web Server," International Journal of Computers and Their Applications, IJCA, vol. 15, no. 2, June 2008, pp. 100-112.

[5] B. Rawal, R. Karne, and A. L. Wijesinha, H. Ramcharan and Songjie Liang. "A Split-protocol Technique for Web Server Migration," The 2012 International workshop on Core Network Architecture and protocols for Internet (ICNA-2012) October 8-11, 2012, Las Vegas, Nevada, USA .

[6] K. Kuppusamy and S. Malathi, "An Effective Prevention of Attacks using GI Time Frequency Algorithm under DDoS", IJNSA journal, vol. 3, no. 6, November 2011, pp. 249-257.

[7] Team Cymru Inc "Bogon route server project", http: //www.cymru.com/BGP/bogon-rs.htm.[Retrieved:July, 2013].

[8] K. Park and H Lee, "On the Effectiveness of Probabilistic Packet Marking for IP Trackback under Denial of Service Attack," Network Systems Lab, Department of Computer Sciences, Purdue University, West Lafayette.

[9] C. Labovitz, D. McPherson and F. Jahanian, "Infrastructure attack detection and mitigation," ACMSIGCOMM 2005 conference, August 2005.

[10] J. Li, J. Mirkovic, M. Wang, P. Reiher and L. Zhang, "SAVE: Source Address Validity Enforcement protocol," In IEEE INFOCOM, vol.6, no.2, June 2002, pp. 81-95.

[11] S. Kandula, D. Katabi, M. Jacob and A. Berger, "Surviving Organized DDoS Attacks that Mimic Flash Crowds," NSDI'05 Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, 2005, vol.2, pp 287 – 300.

[12] D. Moore, G. Voelker and S. Savage "Inferring Internet Denial-of-Service activity," In proceedings of 10th Usenix Security Symposium, August 2001, pp.9-22.

[13] R. Pang, V. Yegneswaran, P. Barford, V. Paxson and L. Peterson, "Characteristics of internet background radiation," In Proceedings of ACM Internet Measurement Conference, October 2004.

[14] M. Dalal, "Improving TCP's robustness to blind in-window attacks," Internet- Draft, May 2005, work in progress.

[15] R. Beverly and S. Bauer. "The Spoofer Project: Inferring the extent of Internet source address filtering on the internet," In Proceedings of Usenix Steps to Reducing Unwanted Traffic on the Internet Workshop SRUTI'05, 2005, pp.53-59.

[16] P. Reiher, J. Mirkovic and G. Prier," Attacking DDoS at the source," In Proceedings of the IEEE International Conference on Network Protocols10, Paris, France, November 2002.

[17] T. Shon, Y. Kim, C. Lee, and J. Moon, "A machine learning frame work for network anomaly detection using SVM and GA," IEEE Workshop and Information Assurance and Security US Military Academy West Point NY, 2005.

[18] H. Burch and B. Cheswick, "Tracing anonymous packets to their approximate source," In Proceedings of the USENIX Large Installation Systems Administration Conference, New Orleans, USA, December 2000, pp. 319–327.

[19] M. Roesch, "Snort - lightweight intrusion detection for networks" http://www.snort.org [retrieved: July, 2013].

[20] Y. Xu and R. Guerin, "On the robustness of router-based denial-of-service (dos) defense systems," SIGCOMM Comput. Commun. Rev., vol. 35, no. 3, pp. 47–60, 2005.

[21] K. Kuppusamy and S. Malathi, "Prevention of Attacks under DDoS Using Target Customer Behavior "IJCSI International Journal of Computer Science Issues, vol. 9, Issue 5, no. 2, September 2012.

[22] S. Ratnaparkhi and A. Bhangee, "Protecting Against Distributed Denial of Service Attacks and its Classification: A Network Security Issue," IJCSI International Journal of Computer Science Issues, vol. 3, issue 1, Jan 2013.

[23] J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms," ACM SIGCOMM Computer Communications Review, volume 34, no. 2, April 2004, pp. 39-53

[24] P Tao, C Leckie and K Ramamohanarao. "Survey of Network-based Defense Mechanisms Countering the DoS and DDoS Problems," ACM Computing Surveys (CSUR), vol. 39, issue 1, article no. 3, August 2007.

[25] D. Slee, "Common Denial of Service Attacks," Jul 10, 2007. http://www.infosecwriters.com/texts.php?op=display&id=589 [Retrieved: July, 2013]

[26] F. Gong, “Detection Techniques: Part III Denial of Service Detection,” McAfee Network Security Technologies Group Jan 03.
[27] H. Alefiya, J. Heidemann, and C. Papadopoulos, "A framework for classifying denial of service attacks," 2003 conference on Applications, technologies, architectures, and protocols for Computer Communications. ACM, 2003.
[29] P. Jain, J Jain and Z Gupta “Mitigation of Denial of Service (DoS) Attack,” International Journal of Computational Engineering & Management IJCEM 11 (2011).
[30] A Chesla, "Generated anomaly pattern for HTTP flood protection." U.S. Patent no. 7,617,170. 10 Nov. 2009.
[31] S. M. Specht and R. B. Lee. "Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures." In ISCA PDCS, pp. 543-550. 2004.
[32] A. Chonka, S. Jaipal and Z. Wanlei, "Chaos theory based detection against network mimicking DDoS attacks." Communications Letters, IEEE 13.9 (2009): pp 717-719.
[33] N Abouzakhar, A Gani, G Manson, M Abuitbel and D King, "Bayesian learning networks approach to cybercrime detection." proceedings of the 2003 Postgraduate Networking Conference (PGNET 2003), Liverpool, United Kingdom. 2003.
[34] S. Kumar, “Smurf-based distributed denial of service (ddos) attack amplification in internet,” In Internet Monitoring and Protection, ICIMP 2007. IEEE Second International Conference July 2007, San Jose, California.
[35] http://www.javvin.com/networksecurity/SmurfAttack.html. [Retrieved: July, 2013].

 ISBN: 978-1-61208-298-1

Distinguishing Legitimate and Fake/Crude Antivirus Software

Masaki Kasuya†, Kenji Kono†‡
†Dept. of Information and Computer Science
Keio University
Yokohama, Japan
‡CREST, Japan Science and Technology Agency
Email: kasuya@sslab.ics.keio.ac.jp, kono@ics.keio.ac.jp

***Abstract*—Fake antivirus (AV) software, a kind of malware, pretends to be a legitimate AV product and frightens computer users by showing fake security alerts, as if their computers were infected with malware. In addition, fake AV urges users to purchase a "commercial" version of the fake AV. In this paper, we search for an *indicator* that captures behavioral differences in legitimate AV and fake AV. The key insight behind our approach is that legitimate AV behaves *differently* in clean and infected environments, whereas fake AV behaves *similarly* in both environments, because it does not analyze malware in the infected environments. We have investigated three potential indicators, file access pattern, CPU usage, and memory usage, and found that memory usage is an effective indicator to distinguish legitimate AV from fake AV. In an experiment, this indicator identifies all fake AV samples (39 out of 39) as fake and all legitimate AV products (8 out of 8) as legitimate. It is impractical for fake AV to evade this indicator because to do so it would require it to detect malware infections, just as legitimate AV does.**

***Keywords*—Antivirus Software; Fake Antivirus Software; Behavior Analysis; Malware**

I. INTRODUCTION

Fake antivirus (AV) software is a severe threat to our computer systems. It pretends to be actual AV software and shows false security warnings to users as if their computer systems were infected with malicious software (malware). Fake AV persuades victim users to purchase a useless, "commercial" version of the fake AV to eliminate bogus threats. "Crude AV" poses a similar threat to fake AV. It differs from fake AV in that it detects malware, but its detection quality is too low to be practical.

The threat of fake AV is real. Symantec detected 43 million installation attempts of fake AV from July 2008 to Jun 2009 [6]. According to Rajab et al. [15], fake AV accounts for 15% of all malware detected by Google's malware detection infrastructure [14]. Fake AV business earns tremendous revenue. Stone-Gross et al. revealed that three kinds of fake AV have earned more than $130 million dollars [16]. McAfee disclosed that the annual revenue of one vendor of fake AV exceeded $180 million dollars [13]. To distribute fake AV, software download sites are sometimes exploited. In fact, one famous download site, CNET, distributed a fake AV sample called RegGenie in 2012 [5].

Fake/crude AV is similar to a social engineering attack; the victim users are deceived and never suspect that fake/crude AV is not legitimate because it is carefully designed to look like legitimate AV. One approach for defending against fake/crude AV is to use signature-based approaches. However, signature-based approaches are exploit-specific, and a signature must be prepared for each instance of fake/crude AV. Therefore, these approaches cannot detect previously unseen instances of fake/crude AV.

In this paper, we reveal behavioral differences between legitimate and fake/crude AV, and propose an *indicator* that captures the behavioral differences in AV software. The key insight behind our approach is that legitimate AV behaves differently in 1) clean environments and 2) infected environments, while fake/crude AV would not show such differences. A clean environment is the one in which no malware has been installed, whereas an infected environment is the one in which malware has been installed. Fake/crude AV is not expected to show behavioral differences in clean and infected environments because it does not analyze malware samples in the infected environment. On the other hand, legitimate AV instances are expected to show the differences because they deeply analyze suspicious instances in the infected environments.

Our indicator can be used in software download sites such as CNET [1] and PCMAG [2]. When AV samples are uploaded to download sites with the tag indicating AV, they can be checked as to whether they are legitimate or fake/crude. Since our indicator works by capturing behavioral differences instead of using signature, it can detect the latest fake/crude AV instances.

This paper shows that "memory usage" is an effective indicator of fake/crude AV software. Surprisingly, crude AV does not show differences in memory usage between clean and infected environments. In our approach, the memory usages of AV-like software in clean and infected environments are compared statistically. The Levene Test [10], an inferential statistic, is used to assess the equality of variances in memory usage. If a software sample that is suspected to be fake/crude AV has statistically the same distribution of memory usage in both environments, it is considered fake. Otherwise, it is considered legitimate.

Since our indicator is based on behavioral differences, fake/crude AV has to mimic the behaviors of legitimate AV in order to evade it. It is not easy to the mimic behaviors of legitimate AV. If fake/crude AV samples change their memory consumption at random, our approach can detect fake/crude AV correctly because it compares memory consumption under several settings, i.e., clean/clean, infected/infected, and clean/infected.

To demonstrate the usefulness of our approach, we have conducted experiments on 39 "real" fake/crude AV samples and 8 legitimate AV products. The results show that our indicator can identify all 39 fake/crude samples, which means

 ISBN: 978-1-61208-298-1

there are no false negatives, and all 8 legitimate products, which means there are no false positives.

The remainder of this paper is organized as follows. Section II describes the differences between fake AV and crude AV, and the current criteria to distinguish them from legitimate AV. Section III explains our basic approach and shows how to distinguish fake/crude AV from legitimate AV by using the Levene Test. Section IV presents our experimental results. Discussion and related work are presented in Sections V and VI. Section VII concludes the paper.

II. Fake AV and Crude AV

There are two types of malicious AV software: fake AV and crude AV. To understand the difficulties of distinguishing between fake/crude AV and legitimate AV, this section describes the behaviors of fake and crude AV, and briefly introduces recent guidelines to distinguish fake/crude AV and legitimate AV.

A. Fake AV

Fake AV mimics the behavior of legitimate AV and shows bogus security warnings without scanning for malware infections in the victims' file systems. To make the behavior resemble that of legitimate AV, fake AV searches the file system to obtain file and/or directory names to be displayed in warning messages. For example, Security Antivirus [8], a fake AV, displays the following message:

Virus name:
Virus.Win32.Faker.a
Infected file:
C:\Documents and Settings\Kasuya\Recent\snl2w.dll
Description:
These programs steal MSN Messenger passwords...

Pathname, `C:\Documents and ... snl2w.dll`, is the real one in the victim's file system. By showing real pathnames in the victim's file system, the fake AV deceives victims into believing the machine is infected with malware and encourages them to the purchase a product version of the fake AV.

The directory traverse of fake AV makes it difficult to distinguish it from legitimate AV. When observed from the outside, fake AV traverses directories just as legitimate AV does. Security Antivirus traverses most directories that all legitimate AV products commonly access. According to our investigation, the access coverage of Security Antivirus is over 99.7% (= 2393 / 2400). This result means that Security Antivirus carefully takes access patterns of legitimate AV into account. In other words, we cannot use directory traversal as an indicator to distinguish fake AV from legitimate AV.

B. Crude AV

Crude AV is low-quality AV software whose detection accuracy is too low to be useful. Crude AV differs from fake AV in that it scans file systems for malware and detects the infection. At the same time, crude AV differs from legitimate AV in that it cannot detect a large portion of widely deployed malware. To confirm that the detection rate of crude AV is very low, we measured the detection rate of Anti-Virus Elite [9], a well-known crude AV. We installed 905 unique instances of malware in Windows XP SP3. Anti-Virus Elite detected only 74 samples. The detection rate was 8.2%. Kaspersky, an example of legitimate AV, detected all 905 samples, i.e., its detection rate was 100.0%.

Crude AV is usually classified into malware. According to VirusTotal [3], an online antivirus scan service, Anti-Virus Elite is classified as malware in 65% (28 out of 43) of commercial AV products. Crude AV is malware because there are sites that urge the visitors to buy a "product" version, which in most cases is just as poor as the crude AV.

Crude AV blurs the boundary between fake and legitimate AV, and makes it more difficult to distinguish fake/crude AV from legitimate AV. Crude AV traverses file systems and inspects suspicious files that may contain malware. Aside from the quality of detection, crude AV behaves very similarly to legitimate AV.

C. Current Criteria to Distinguish Legitimate and Fake/Crude AV

Recently, a security industry has published a white paper [7] to help end-users identify fake security products such as fake AV. The document provides a helpful checklist to judge whether the users' computers are infected with fake AV. The checklist says, for instance, that fake AV reports an unreasonably high number of infections, shows a popup window frequently that warns your machine is infected with malware, and suggests the purchase of a commercial version.

This checklist is useful for manual inspection for discovering fake AV. However, these criteria do not suit automated distinction of fake/crude and legitimate AV. Suppose that we attempt to automate the process of counting the number of reported infections. Since the reports are shown in natural language, it is not easy for computers to understand the reports. Even if we could interpret the reports, there are samples of fake AV that do not show up in a lot of reports. For example, Anti Spyware Expert has only 18 reports of infection.

Furthermore, this checklist does not address how to distinguish crude AV from legitimate AV. Since crude AV behaves similarly to legitimate AV aside from the quality of detection, it is almost impossible to draw up a guideline for crude AV.

III. Searching for indicators

In this section, we describe our search for fake/crude AV indicators. As mentioned above, the key insight behind our approach is that legitimate AV behaves differently in clean and infected environments while fake/crude AV behaves similarly in both environments. A good indicator captures behavioral differences only in legitimate AV.

A. What is a good indicator?

A fake/crude AV indicator should satisfy at least three requirements. First, it should be applicable to as many instances of fake/crude AV as possible. In particular, it should be applicable to previously-unseen instances of fake/crude AV.

 ISBN: 978-1-61208-298-1

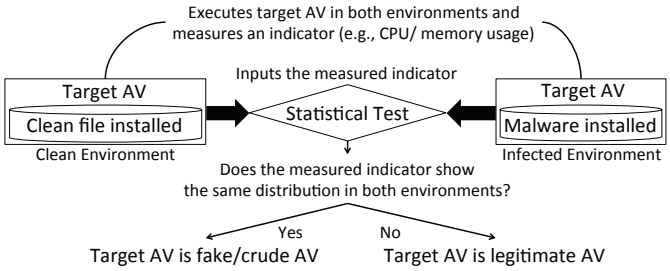


Fig. 1. Our basic approach

Second, it should be impractical for fake/crude AV to evade the indicator. The criminals that make use of fake/crude AV for their own profit do not want to spend a lot of money on development because it would reduce their revenues. A fake/crude AV indicator would thus be ideal since criminals would have to incorporate the same functionalities as legitimate AV in order to evade it and they would cost a fortune to develop software equivalent to legitimate AV.

Third, a fake/crude AV indicator should enable automatic distinction of fake/crude AV from legitimate AV. If the distinction process is automated, it can be incorporated into legitimate AV or deployed on software download sites [1][2]. The guidelines presented in Section II-C assume manual inspection of several aspects of suspicious behavior of AV-like software. In this paper, we seek a fake/crude AV indicator that does not require manual intervention. Rather than relying on visual inspection, we seek a fake/crude AV indicator from information that can be obtained in a systematic way. For example, we look for system call patterns or resource usage patterns that differentiate fake/crude AV from legitimate AV.

One approach to deriving a fake/crude AV indicator is to use binary analysis. We cannot rely on the source code because the source code of malware is not available in public. Binary analysis has the potential to identify a lot of operations (e.g., system calls and their arguments) that malware may do. However, it is not easy to apply binary analysis to fake/crude AV because malware can be obfuscated using a technique like binary obfuscation [17]. In addition, it is not straightforward to find a sequence of operations that can differentiate fake/crude AV from legitimate AV. Hence, we decide not to take this approach.

B. Basic Approach

Our basic approach is to *compare* potential indicators obtained in *clean* and *infected* environments. A clean environment is one in which no malware has been installed. We assume that an execution environment just after installing an operating system from read-only media such as DVD-ROM is clean. Therefore, an environment with harmless files is also clean. A clean environment can be prepared by using the recent technology of virtual machines. Once a clean environment has been prepared, we can reuse it by saving it as a virtual machine image. An infected environment is one in which malware has been installed. This environment is also saved as a virtual machine image for reuse.

Figure 1 illustrates the basic approach used in this paper. For each sample of AV-like software (it is unknown whether the sample is fake/crude AV or legitimate AV at this point of time), indicators are measured in clean and infected environments and *statistically* compared. The key insight behind this approach is that legitimate AV behaves *differently* in clean and infected environments while fake/crude AV behaves *similarly* in clean and infected environments, because legitimate AV thoroughly analyzes files suspected to be infected with malware. fake/crude AV *cannot* change its behavior depending on the presence of an infection because it does not detect malware infection.

This approach satisfies the three requirements described in Section III-A. First, the fake/crude AV indicator does not use features specific to each instance of fake/crude AV, and thus, should be applicable to a wide variety of fake/crude AV. As shown in Section IV, our indicator successfully identified all (39 out of 39) fake/crude AV samples and all (8 out of 8) legitimate AV samples. Second, it is impractical to try to evade the indicator. Since fake/crude AV must change its behavior depending on the presence of malware infection, it must be equipped with the detection facilities that are equivalent to legitimate AV; that is, criminals must develop a legitimate AV to evade the indicator. Finally, the distinction process can be automated. There is no need for manual intervention since the indicator can be used in a systematic way and indicator measurements are compared using a statistical test.

C. Examining Potential Indicators

To discover a good indicator, we examine three candidates that can be obtained systematically: 1) the file access pattern, 2) CPU usage, and 3) memory usage. To obtain them, we use system calls hook to get the file accesses and performance monitor, a default application of the Windows OS, to get the CPU usage and memory usage.

1) File Access Pattern: While a legitimate AV has to investigate a file's content to determine whether it is infected with malware, fake/crude AV only traverses directories to obtain real pathnames; it does not access files as often as legitimate AV does, because fake/crude AV does not look hard for malware infection.

Unfortunately, file access patterns are not a good indicator of fake/crude AV because the patterns of some fake/crude AV samples are similar to those of legitimate AVs. Figure 2 shows the similarity of file access patterns between 8 legitimate AV products and 39 fake/crude AV samples. Each bar corresponds to one fake/crude AV sample, and shows the ratio of files accessed by each fake/crude AV sample to those commonly accessed by legitimate AV samples. (There are 10 bars in the figure because the remaining 29 samples do not access the files). Figure 2 shows that six fake/crude AV samples resemble legitimate AV products in terms of file access.

2) CPU Usage: Next, we investigate CPU usage. Since a malware scan such as signature matching is executed in user mode not kernel mode, we focus on the proportion of user mode time of CPU usage. User-time is expected to increase in legitimate AV if it is executed in infected environments because sophisticated scanning algorithms consume a lot of user-mode time. On the other hand, fake/crude AV is not expected to increase user-mode time in infected environments because it does not search for malware infection.

 ISBN: 978-1-61208-298-1

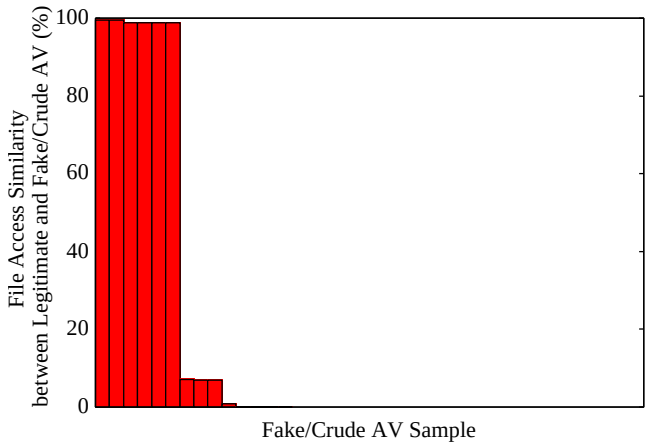


Fig. 2. Similarity of file access patterns between legitimate AV and fake/crude AV. The file access pattern is not be a good indicator because 6 out of 39 fake/crude AV samples access files accessed by legitimate AV products.

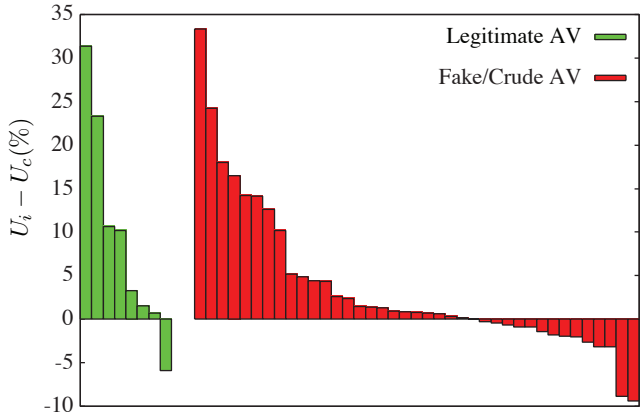
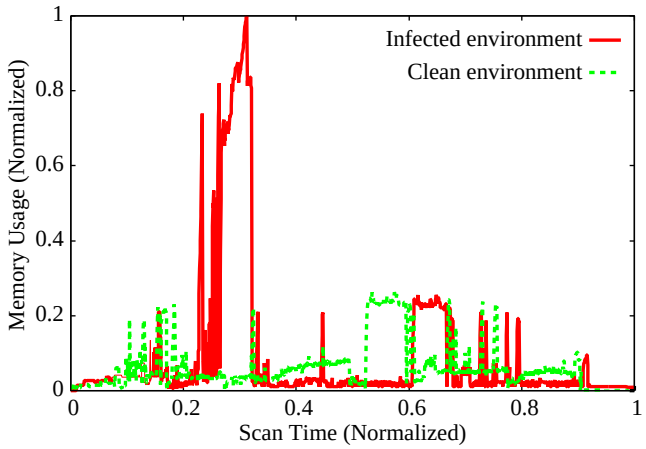


Fig. 3. Comparison of user-mode times in clean and infected environments. U_i and U_c represent user-mode time ratios in an infected and a clean environment, respectively. The green bars show $U_i - U_c$ of legitimate AV products, and the red bars show those of fake/crude AV samples. Regardless whether it is fake/crude or legitimate, $U_i - U_c$ ranges from -10% to 30%.

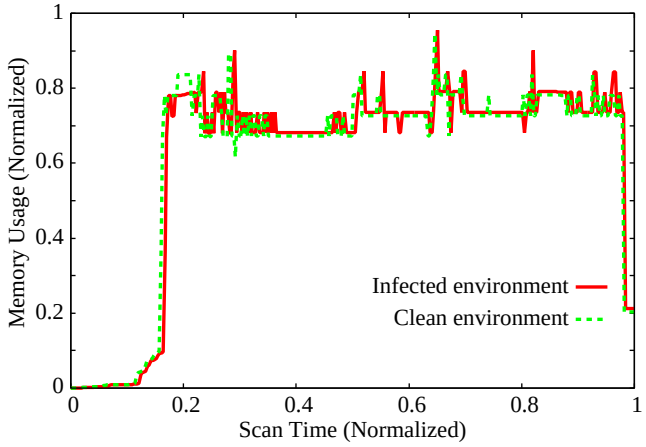
In spite of our expectations, the differences in user-mode time between clean and infected environments are not useful for distinguishing fake/crude AV from legitimate AV. Figure 3 shows the differences in user-mode time ratio between clean and infected environments. The y-axis shows $U_i - U_c$, where U_i stands for the user-mode time ratio measured in an infected environment and U_c stands for the user-mode time ratio measured in a clean environment. The green bars show $U_i - U_c$ of legitimate AV products, and the red bars show those of fake/crude AV samples. The overall trend in the user-mode time ratios is the same in the legitimate products and fake/crude samples. $U_i - U_c$ ranges from -10% to 30%.

3) Memory Usage: Finally, we examine memory usage. Memory usage is expected to increase in infected environments in legitimate AV because it requires more memory to perform in-depth analyses of malware. On the other hand, fake/crude AV is not expected to increase memory usage because it should behave similarly in clean and infected environments.

Our preliminary results show memory usage is an effective indicator to distinguish legitimate AV from fake/crude AV.



(a) McAfee (A legitimate AV product)



(b) Anti-Virus Elite (A fake/crude AV sample)

Fig. 4. Memory usages of legitimate AV and fake/crude AV. Memory usage and scan time are normalized. The legitimate AV uses a significant amount of memory when it detects malware. On the other hand, fake/crude AV hardly changes its usage in going from clean to infected environments.

Figure 4 shows the memory usages of one legitimate AV product and one fake/crude AV sample. It reveals memory usages differ in legitimate and fake/crude AV. In the legitimate AV product, the memory usage increases in infected environments. However, the fake/crude AV sample does not show such an increase in infected environments; it shows almost the same trend in both environments.

Figure 5 shows V_i/V_c for each legitimate AV product and fake/crude AV sample. V_i stands for the variance in an infected environment and V_c stands for the variance in a clean environment. As you can see from the figure, V_i/V_c shows different trends for legitimate AV and fake/crude AV. This suggests that memory usage is a good indicator of fake/crude AV.

D. Memory Usage as an Indicator

This section describes a concrete method to distinguish fake/crude AV from legitimate AV. On the basis of examinations in the previous sections, we choose memory usage as an indicator of fake/crude AV. A sample of AV-like software is installed in clean and infected environments and the

 ISBN: 978-1-61208-298-1

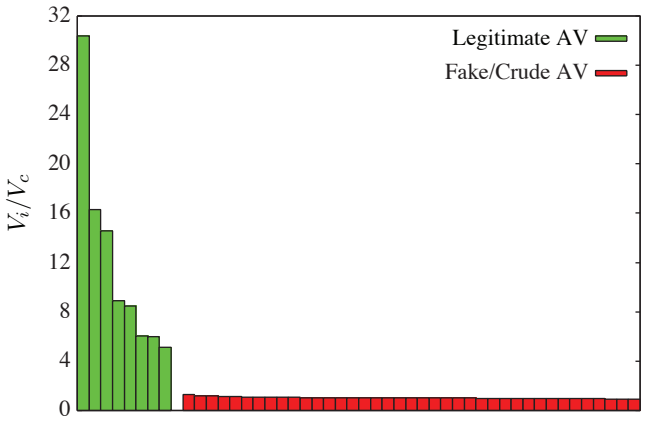


Fig. 5. Comparison of variances in clean and infected environments. V_i and V_c represent the variances of memory usage distributions in an infected and a clean environment, respectively. All legitimate AV products significantly increase the variances in the infected environment. However, fake/crude AV samples hardly change variances in these environments.

memory usage is measured in each environment. An infected environment is prepared by installing 905 unique instances of malware (about 500 MB in total), and a clean environment is by installing 500 MB of clean files. The installed malware instances and clean files are the same size and have the same directory structure. Note that we do not have to prepare these environments every time a test is performed, because a virtual machine image can be copied for reuse.

Memory usage is measured every second in each environment. For ease of mathematical formalization, the clean and infected environments are numbered 1 and 2. The measured memory usage values are grouped into a sequence for each environment and represented by Y_i, where i denotes the number of group ($1 \leq n \leq 2$).

If the distributions in Y_1 (clean env.) and Y_2 (infected env.) are not statistically different, we conclude that the tested sample of AV-like software is fake/crude AV. Otherwise, we conclude that the tested sample is legitimate AV.

To compare the memory usage distributions in each environment, we use the *Levene Test* [10], a well-known inferential statistic used to assess the equality of variances in different samples. It tests the null hypothesis that the population variances are equal. The Levene Test compares the distributions of two sequences Y_i and Y_j. If the results of the test are less than the significance level (0.05 in this paper), the difference is statistically significant. In our method, memory usage is measured M times to mitigate fluctuations and the Levene Test is repeated. If the M results of the Levene Test are all less than 0.05, we consider that the distributions are different. Since it is time-consuming to measure indicators M times, we measure indicators $\lceil\sqrt{M}\rceil$ times and perform the Levene Test on any pair of the measured indicators.

IV. Experiments

This section shows that memory usage can be used to distinguish fake/crude AV from legitimate AV. We collected 39 fake/crude AV samples listed in Table I from a malware collection site [12] and Malware Domain List [11]. In addition

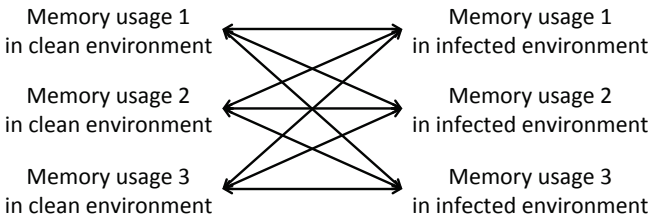


Fig. 6. Memory usages in different environments were measured for 3 times and the Levene test was performed on any pair of memory usages. If all pairs showed statistical significance, the tested AV sample was identified as legitimate. Otherwise, it was deemed fake/crude.

TABLE I. Fake/Crude AV samples

XP Internet Security 2011	XP Internet Security 2012 6.0.2900.2180
XP Home Security 2011	XP Home Security 2012 6.0.2900.2180
XP Anti Spyware 2011	XP Anti Spyware 2012 6.0.2900.2180
XP Antivirus 2011	XP Antivirus 2012 6.0.2900.2180
XP Security 2011	XP Security 2012 6.0.2900.2180
XP Total Security 2011	PC Privacy Cleaner 1.0.22.4
Patchup Plus	Virus Remover 2008 1.0.15.2
Security Tool	Virus Remover 2009 1.0.9.0
System Security	Anti Spy Safeguard 1.0.0.0
XL Guarder	Security Antivirus 2.0.2.18
Security Shield	Major Defense Kit 1.0.0.0
Protect Code	Anti Spyware Bot 9.6.9
Adware Bot 12.0.6	Security Defender 1.6.812.0
Reg Clean 1.0.0.1	Malware Removal Bot 12.0.6
Onescan 1.0.0.1	Anti Spyware Expert 1.0.22.2
Anti-Spyware 12.0.6	Anti-Virus Elite v5.0
Error Sweeper 2.8.0	Pest Detector 1.0.0.0
Registry Smart 2.10.0	Netcom3 PC Cleaner 9.1.10
Red Cross 1.0.0.0	Peak Protection 1.0.0.0
Privacy Control 2.6.0.0	

to the fake/crude AV samples, 8 legitimate AV products listed in Table II were collected. Clean and infected environments were prepared by using KVM with Qemu 1.1.1, in which Windows XP SP3 was installed and 1GB of memory is allocated. Although we used Windows XP in the experiments, our approach was not limited to a specific OS. These environments were prepared as described in Section III-D. The Levene Test was repeated 9 times. In other words, M equaled 9. Since $\lceil\sqrt{M}\rceil$ was 3 in this case, memory usage in each environment was measured 3 times.

The Levene Test was performed on pairs of measured memory usages in clean and infected environments as shown in Figure 6, and counts of less than significance level (0.05) were gathered. If the count reached 9, the tested AV sample was identified as legitimate. Otherwise, it was identified as fake/crude.

Our method identified all 39 fake/crude AV samples. The results are shown in Table III. The second row in Table III shows that none of fake/crude AV samples had positive counts of 9. In particular, the fake/crude AV could not evade detection by changing memory usage at random. For example, the two fake AV samples in Figures 7 and 8 change their memory usage at random. However, our indicator detects them correctly, because the Levene Test is performed on any pair of the measured indicators. Moreover, Figure 9 shows the memory usage of Anti-Virus Elite, a crude AV. Surprisingly, Anti-Virus Elite hardly changes its usage between clean and infected environments despite that it has a function to detect malware.

The rate of false positives is low; our method correctly

 ISBN: 978-1-61208-298-1

TABLE II. LEGITIMATE AV PRODUCTS

Avast Pro Antivirus 7.0.1426	G Data Antivirus 2011 21.1.0.1
AVG Antivirus 2012.0.1913	Kaspersky Anti-Virus 2011 11.0.2.556
McAfee VirusScan 15.0.294	ESET NOD32 Antivirus 4.2.71.2
Norton AntiVirus 18.7.0.13	Panda Antivirus Pro 2011 10.00.00

TABLE III. RESULTS OF LEVENE TEST

Name	# of Positive Results	Result
Adware Bot	0	Fake/Crude
Anti Spy Safeguard	0	Fake/Crude
Anti-Spyware	5	Fake/Crude
Anti Spyware Bot	0	Fake/Crude
Anti-Virus Elite	1	Fake/Crude
Anti Spyware Expert	3	Fake/Crude
Error Sweeper	3	Fake/Crude
Major Defense Kit	0	Fake/Crude
Malware Removal Bot	0	Fake/Crude
Netcom3	5	Fake/Crude
Onescan	0	Fake/Crude
Patchup Plus	0	Fake/Crude
PC Privacy Cleaner	5	Fake/Crude
Peak Protection	0	Fake/Crude
Pest Detector	3	Fake/Crude
Privacy Control	3	Fake/Crude
Red Cross	0	Fake/Crude
Reg Clean	0	Fake/Crude
Registry Smart	0	Fake/Crude
Security Antivirus	0	Fake/Crude
Protect Code	2	Fake/Crude
Security Defender	6	Fake/Crude
Security Shield	3	Fake/Crude
Security Tool	0	Fake/Crude
System Security	5	Fake/Crude
Virus Remover 2008	0	Fake/Crude
Virus Remover 2009	0	Fake/Crude
XL Guarder	0	Fake/Crude
XP AntiSpyware 2011	5	Fake/Crude
XP AntiSpyware 2012	4	Fake/Crude
XP AntiVirus 2011	6	Fake/Crude
XP AntiVirus 2012	2	Fake/Crude
XP HomeSecurity 2011	2	Fake/Crude
XP HomeSecurity 2012	4	Fake/Crude
XP InternetSecurity 2011	2	Fake/Crude
XP InternetSecurity 2012	2	Fake/Crude
XP Security 2011	4	Fake/Crude
XP Security 2012	3	Fake/Crude
XP TotalSecurity 2011	2	Fake/Crude
Avast	9	Legitimate
AVG	9	Legitimate
McAfee	9	Legitimate
NOD32	9	Legitimate
G Data	9	Legitimate
Norton	9	Legitimate
Kaspersky	9	Legitimate
Panda	9	Legitimate

identified the 8 legitimate AV products listed in Table II, i.e., no false positives in this experiment. All of these samples show statistical differences in clean and infected environments. Table III shows that the Levene Test reveals statistical significances with all of the legitimate AV products. As a result, our method judges these to be legitimate.

V. DISCUSSION

A. Evasion

1) Random Memory Usage: It is useless to change memory usage at random to evade our indicator. If memory usage is measured only once in clean and infected environments, the randomly changing memory usage could evade our indicator. However, as explained in Section III-D, memory usage is measured M times in our approach and the Levene Test is performed multiple times. To deceive our approach, fake/crude AV samples must change their memory usage based on the presence of malware. This means that the fake/crude AV would have to act as legitimate ones; that is, they would correctly detect malware infections.

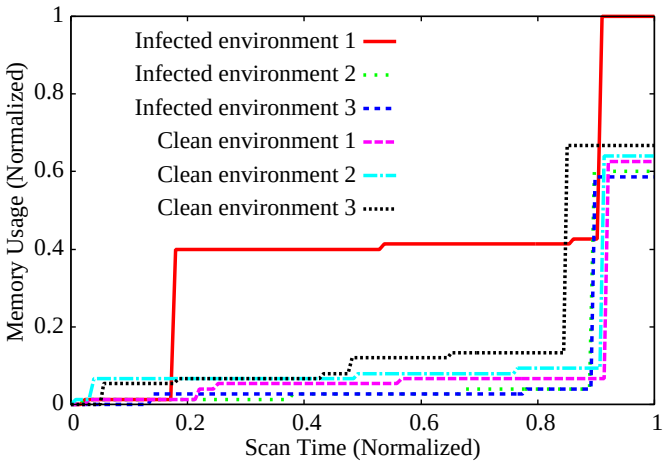


Fig. 7. Memory usages of Anti Spyware Expert, a fake AV sample. The memory usage in infected environment 1 is obviously different. However, since the other memory usages are similar, our indicator identifies this AV sample as fake. Memory usage and scan time are normalized.

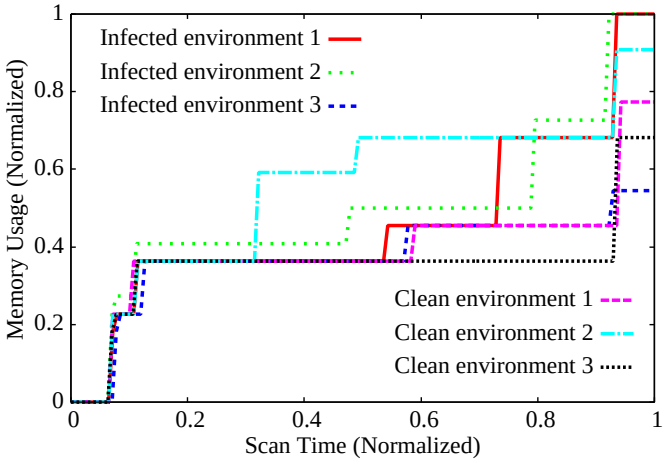


Fig. 8. Memory usage of System Security, a fake AV sample. Although the memory usages are different, the Levene Test does not show statistical significance for some pairs (e.g., infected environment 3 and clean environment 1). Memory usage and scan time are normalized.

2) Exploiting Open Source AV and Source Code of Legitimate AV: One possible approach to evade our indicator is to use open source AV or leaked source code of legitimate AV. Fake/crude AV samples based on legitimate AV could evade our indicator because their behavior is similar to legitimate AV. However, we believe our indicator raises the bar to developing fake/crude AV because fake/crude AV developers require the source code of product-quality legitimate AV. We also hope vendors can quickly develop effective signatures to detect fake/crude AV based on their products, since the legitimate vendors have the source code of their products and deeply understand the internal behavior of their products.

 ISBN: 978-1-61208-298-1

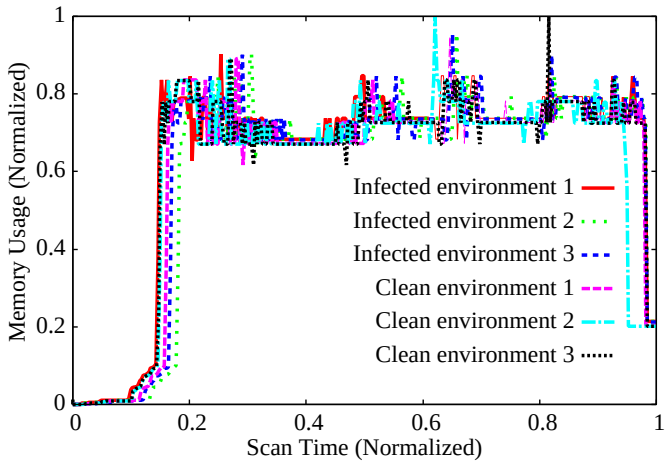


Fig. 9. Memory usages of Anti-Virus Elite, a crude AV sample. Although it can detect malware, all memory usages are almost same. Memory usage and scan time are normalized.

B. Deployment scenario

Our approach can serve to prevent software download sites form distributing fake/crude AV. Software download sites such as CNET [1] and PCMAG [2] should not distribute fake/crude AV. In spite of the careful management, though, CNET distributed a sample of fake AV in the middle of September 2012 [5]. A discrimination system based on our indicator can prevent the users of those sites from downloading fake/crude AV. Since AV software is usually indexed by tags such as "antivirus" in software download sites, all the pieces of software indexed by "antivirus" can be tested to decide if they are legitimate or fake/crude AV.

Our approach is ineffective at finding distribution sites for fake/crude AV. Since it uses the difference between legitimate and fake/crude AV, tested samples should be legitimate AV or fake/crude AV. However, it is difficult to automatically collect only AV-like software on the web. In this case, our approach cannot correctly classify non-AV samples. As a result, it reports a lot of false negatives and false positives.

C. Other Possible Indicators

We investigated three indicators, file access patterns, CPU usage and memory usage, and found that memory usage is a good indicator to distinguish legitimate AV and fake/crude AV. Although we have not sought other possible indicators in this paper, there may be other indicator to distinguish them. In the future, we plan to explore other possible indicators and determine which indicator is the best.

VI. Related Work

Recently, two studies have reported long-term analyses of fake/crude AV threat ecosystems. They show the traditional signature-based and blacklist-based approaches are useless against fake/crude AV. Rajab et al. show it is practically impossible to keep signatures with a high detection rate against fake/crude AV [15]. The detection rate rises and falls frequently. Cova et al. show that neither IP nor domain-based blacklists are effective on fake/crude AV [4]. Legitimate web sites are often blocked in IP-based blacklists, and domain-based blacklists are evaded by rotating short-lived domains.

Stone-Gross et al. suggest that credit-card companies should endeavor to identify fake/crude AV companies [16]. Fake/crude AV companies monitor the refunds that customers demand from their credit card providers, and they control these refunds so as to keep the chargeback rates low. However, this behavior leads to unusual patterns in chargebacks, which may be leveraged by credit-card companies to identify and ban fraudulent companies.

A white paper has been published to identify fake/crude AV by visual inspection [7]. It provides diverse characteristics about fake/crude AV. By using it, computer users can identify fake/crude AV by visual inspection. As described in Section II-C, some fake/crude AV samples do not have such characteristics.

VII. Conclusion

In this paper, we have searched for an indicator that captures behavioral differences in legitimate AV and fake/crude AV. We have conducted experiments showing that memory usage would be a good indicator, and developed a systematic method, based on a statistical test, to distinguish fake/crude AV from legitimate AV. To demonstrate the effectiveness of our method, we collected and tested 39 real fake/crude AV samples and 8 legitimate AV products. According to our experiments, our method correctly identified all fake/crude AV samples and all legitimate AV products.

References

[1] "CNET | Download.com," http://download.cnet.com [retrieved: Jul, 2013].

[2] "PCMAG.COM," http://www.pcmag.com/downloads [retrieved: Jul, 2013].

[3] "Virus Total," https://www.virustotal.com [retrieved: Jul, 2013].

[4] M. Cova, C. Leita, O. Thonnard, A. D. Keromytis, and M. Dacier, "An Analysis of Rogue AV Campaigns," in *Proceedings of the 13th International Symposium on Recent Advances in Intrusion Detection (RAID '10)*, Sep. 2010, pp. 442–463.

[5] S. Doyle, "How To Remove RegGenie Rogue Antivirus Software – Uninstall RegGenie Malware (Identity Theft Protection)," http://botcrawl.com/how-to-remove-reggenie-rogue-antivirus-software/ [retrieved: Jul, 2013], Sep. 2012.

[6] M. Fossi, D. Turner, E. Johnson, T. Mack, T. Adams, J. Blackbird, M. K. Low, D. McKinney, M. Dacier, A. D. Keromytis, C. Leita, M. Cova, J. Orbeton, and O. Thonnard, "Symantec Report on Rogue Security Software," http://www4.symantec.com/Vrt/wl?tu_id=TeCm125590003756772344 [retrieved: Jul, 2013], Oct. 2009.

[7] A. Karnik, J. Avelino C. Rico, A. Prakash, and S. Honjo, "Identifying Fake Security Products," http://www.mcafee.com/us/resources/white-papers/wp-identifying-fake-security-products.pdf [retrieved: Jul, 2013], 2009.

[8] U. Kiguolis, "Remove Security Antivirus," http://www.2-spyware.com/remove-security-antivirus.html [retrieved: Jul, 2013], Mar. 2010.

[9] U. Kiguolis, "Remove Anti-Virus Elite," http://www.2-spyware.com/remove-antivirus-elite.html [retrieved: Jul, 2013], Feb. 2012.

[10] H. Levene, *Robust Tests for Equality of Variances*, Ingram Olkin and Sudhish G. Ghurye and Wassily Hoeffding and William G. Madow and Henry B. Mann, Ed. Stanford University Press, 1960.

[11] MDL, "Malware Domain List," http://www.malwaredomainlist.com/ [retrieved: Jul, 2013].

[12] Open Malware, "Open Malware — Community Malicious code researh and analysis," http://offensivecomputing.net [retrieved: Jul, 2013].

 ISBN: 978-1-61208-298-1

[13] F. Paget, "Running Scared: Fake Security Software Rakes in Money Around the World," http://www.mcafee.com/us/resources/white-papers/wp-running-scared-fake-security-software.pdf [retrieved: Jul, 2013], 2010.

[14] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose, "All Your iFRAMEs Point to Us," in *Proceedings of the 17th USENIX Security Symposium*, Jul. 2008, pp. 1–16.

[15] M. A. Rajab, L. Ballard, P. Mavrommatis, N. Provos, and X. Zhao, "The Nocebo Effect on the Web: An Analysis of Fake Anti-Virus Distribution," in *Proceedings of the 3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '10)*, Aug. 2010.

[16] B. Stone-Gross, R. Abman, R. A. Kemmerer, C. Kruegel, D. G. Steigerwald, and G. Vigna, "The Underground Economy of Fake Antivirus Software," in *Proceedings of the 10th Workshop on Economics of Information Security (online) (WEIS '11)*, Jun. 2011.

[17] Z. Wu, S. Gianvecchio, M. Xie, and H. Wang, "Mimimorphism: A New Approach to Binary Code Obfuscation," in *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10)*, Oct. 2010, pp. 536–546.

 ISBN: 978-1-61208-298-1

Behavior Risk: the Indefinite Aspect at the Stuxnet Attack?

Wolfgang Boehmer
Technische Universität Darmstadt
Email: wboehmer@cdc.informatik.tu-darmstadt.de

***Abstract*—In 2009, the Stuxnet virus was first observed in the wild and was considered as a novelty among the viruses. The Stuxnet virus is classified as a *game changer* and so we denote it *causa Stuxnet*. For the critical infrastructures, it was inconceivable, that a specific virus has been developed for industrial systems. Besides this novelty, the infection path was different from the typical patterns of attack and infection in the field of office communication. In this article, we focus only on the infection path of Stuxnet. We use the Game Theory to analyze the infection path. We found that the infection path is one game in a complex multi-layer game. As a result, based on a Nash equilibrium, a cooperative solution is proposed to arm the existing IT security concepts against such infections. Nevertheless, the existing IT security concepts are not useless, but the behavioral risk has to be taken into account.**

***Index Terms*—Event risks; behavioral risks; trust/investor game; IT security concept; industrial control system**

I. Introduction

In the past, the focus was placed on attacks and countermeasures on purely IT systems in the business community or, better, in the field of office communication. However, it had also previously been attacks on industrial facilities, such as D. Denning analyzed in the article [1]. In their article, a comprehensive analysis and categorization of attacks on industrial facilities has been done. The Stuxnet virus was called a *game changer* in their article,

In 2009, as N. Falliere et. al. from Symantec wrote [2], the Stuxnet virus was first observed in the wild and was considered as a novelty among the viruses. For the critical infrastructures, it was hitherto unthinkable that a specific virus has been developed for industrial control systems. Besides this novelty, the route of infection was different from the typical patterns of attack and infection in the field of office communication.

Typically, an attack on IT systems is analyzed and described by so-called attack trees. Attack trees are a good way to detect possible attacks or mimic attack routes in advance.

In 1994, Edward Amoroso's book Fundamentals of Computer Security Technology described threat trees, a tree structure very similar to attack trees. By the late 1990s, papers were beginning to appear describing the attack tree analysis process in some detail. For instance, in the 1998 paper Toward a Secure System Engineering Methodology [3] the authors describe a mature, attack tree-based approach to analyzing risk. One of the first who dealt systematically with this type of technical attacks was Bruce Schneier. In 1999, he published the idea of *attack trees*, which were directed towards technical systems [4]. The idea was taken further and improved by the company Amenaza Tech. Ltd.

Game theory considered – very roughly speaking – conflict situations between one or more Individuums. Fiona Carmichael [5] wrote in her book (2005) on page 3: the idea of Game Theory is to analyze situations where two or more Individuums (or institutions) the outcome from action by one of them depends not only on the particular action taken by the individuums but also on the action taken by the other (or others). In these circumstances the plans or strategies of the individual concerned will be dependent on expectations about what the other is doing. Thus, Individuums in these kinds of situations are not making decisions in isolation, instead their decisions making is interdependently related.

In essence, game theory is primarily not used for the analysis of attacks [5] page 4. However, a certain relationship exists between an attack tree and game theory in a particular form of play (zero-sum game) of two players. In the article by Kordy et al. [6] it was shown in a comparison that an attack-defense tree and a strategic zero-sum game of two players to their binary information are equivalent in the extensive form. The extensive form denotes a game tree. In this context, a strategic game is a scenario or situation where, for two or more Individuums, their choice or behavior has an impact on the other (or others). A Player is participant in a strategic game and a strategy is a player's plan of action for the game. If only one player exist in a game, then this game is called a game against nature, which is generally called decision theory (rather than game theory). We will come back to this issue later on.

Another type of attack has been perfected, e.g., by Kevin D. Mitnick [7]. This type is classified as a social engineering attack and has been discussed multiple times in the literature.

From attack-trees, insights are gained and then countermeasures are designed to be armed for future attacks. However, the mere development of measures is not very helpful for an organization. These measures must be incorporated into a security concept and coordinated with other measures.

Security concepts have always been established for safeguarding companies and industrial plants. It can be shown that a good security concept cannot be reduced to a simple list of measures. However, the measures listed in the security concept must always result from a procedure or methodology. In the literature, there are numerous articles on the development of security concepts. But the objectives in the security

 ISBN: 978-1-61208-298-1

concepts discussed in the literature are often very different. This paper addresses three protection goals, pursued from different perspectives. However, security concepts based on the standards (e.g. ISO 27001) address three main protection targets (availability, confidentiality, integrity).

The relationship between security objective, systems and the security concept can be produced with the following definition:

Def. 1: A security concept includes measures $\mathfrak{M}$ to ensure the security objectives of confidentiality, availability and integrity of a system (ψ) aligned to a predefined level.

The three protection goals (*confidentiality, availability and integrity*) behave as random variables in a probability space and can counteract the risk of protection violation or deviation of the predefined level through appropriate security concepts. The *predefined level* is directly correlated with the risk appetite of a company (see Figure 1). The reader should note that the risk appetite must be defined for each of the protection goals. The risk appetite levels are set by the management for different activities or parts of the organization.

However, the different types of risks that may lead to a protection violation must be analyzed separately, because risks can be divided into state risks, behavioral risks, and hybrid risks (see Figure 2). Furthermore, not only the types of risks, but also the underlying systems ($\psi \in \Psi$) are to be differentiated in a security concept.

In general, for the term risk in this paper, we follow the definition from the Circular 15/2009: Minimum Requirements for Risk Management (MaRisk) issued by the Federal Financial Supervisory Authority (BaFin).

Def. 2: Risk is understood as the possibility of not reaching an explicitly formulated or implicitly defined objective. All risks identified by the management present a lasting negative impact on the economic, financial position or results of the company may have to be considered as much as possible.

The research contribution results from the analysis of the infection path of the Stuxnet virus [8] and combats it with methods of Game Theory. The result evident from the analysis is that only cooperative behavior between software manufactures for SCADA systems (e.g. Siemens, ABB, AREVA) and the software users (operator of the power plant) is sufficient for a Nash equilibrium. For a Nash equilibrium it is characteristic that actors cannot obtain a better position, if they deviate from their strategy. The cooperation (Nash equilibrium) will cause the software manufacturer for the SCADA systems, as a first step, to generate a signature in advance using the software and provide the signature to the power plant operator in advance, before any service technician arrives at the power plant. The signature makes it possible to uncover an evolution of the software (virus infection) in the IT equipment of service technicians as a second step with only a little effort. This constructive solution to this type of behavioral risk is already in the implementation stage at one power supplier in Germany.

It is also clear from the game analysis that the (current) practice of disinfecting the infected systems retroactively only represents the second best solution, because it is not ruled out that modifications of the Stuxnet virus could infect the sensitive control systems of industrial plants in the future. Moreover, conventional virus scanners in SCADA (supervisory control and data acquisition) is a type of industrial control system (ICS) Systems are generally hardly used. This preventive solution then enables, if the software manufacturer for SCADA systems cooperates, future modifications of the Stuxnet virus or a variant of a Stuxnet virus to be uncovered effectively.

The rest of the article is divided into four sections. In the next section, the underlying model equations are explained. In the third section, case studies are discussed for the different types of risks; for example, hybrid risk analysis (see Fig. 2, no. (2)). In better security concepts, this approach can be found in the development of security measures. According to, e.g., the ISO 27001 and ISO 27005, a scenario analysis is required to create a security concept (*statement of applicability*). Subsequently, behavioral risks are discussed on the example of the Stuxnet virus. Such risks are based only on the misbehavior of Individuums. These are marked with the no. 3 in Fig. 2. With the Game Theory, the Causa Stuxnet is analyzed. Here, the route of infection is analyzed as a partial game in a complex multi-layered game. A Nash equilibrium is achieved, the knowledge of which can eliminate general routes of infection preemptively. However, measures derived from the analysis of behavioral risks have rarely been included in the security concepts. Also, methods of Game Theory, which consider behavioral risks, have not previously been included in any standard.

In the fourth section, we discuss the related work and in the last section, there is a brief summary and an outlook on further research. This article is an extended version of [9].

II. The model

In essence, we will deal in this article with hybrid risks described in Figure 2, number (2) and the behavioral risks, number (3) for analyzing the Stuxnet virus.

An IT/Inf.-Security concept reflects the complementary relationship between security and risk for a system $\psi \in \Psi$. This complementary relationship is that the lower the security ($\mathcal{S}ec$), the higher the risk ($\mathcal{R}$) of a violation for a system ψ of the three control objectives (*cf.* Definition 1); therefore risk and security are negatively correlated.

Figure 1 illustrates this relationship qualitatively for the protection goal availability.

To illustrate this negative correlation, risk and security *simplified* are normalized by the interval $[0, 1]$, with

$$\mathcal{S}ec = 1 - \mathcal{R}. \tag{1}$$

The risk ($\mathcal{R}$) in the sense of operational risk is obtained as the probability (Pr) of an event (E) on the impact on a system, e.g., on an open system (ψ_1), as a value chain with a negative outcome ($Loss, L$) in monetary units (euro), in $\mathbb{R}^+$ [10]. This relationship can be expressed as follows

$$\mathcal{R} = Pr_E \times L\ [\mathbb{R}^+]. \tag{2}$$

 ISBN: 978-1-61208-298-1

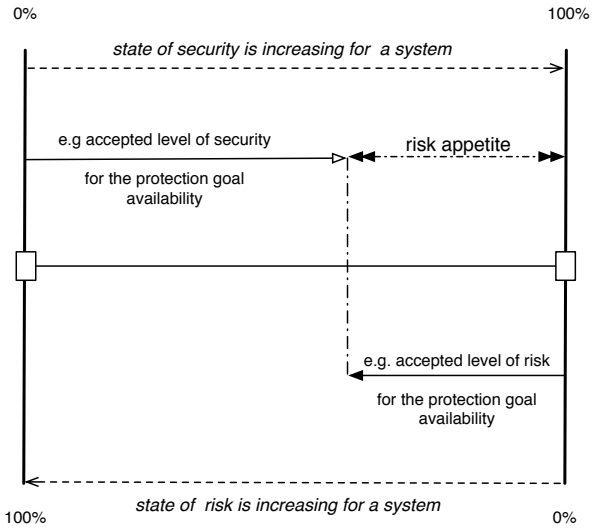


Figure 1: Security versus risk

At first glance, these two definitions (Eq. 1, Eq. 2) do not seem to attain anything. However, if the risk ($\mathcal{R}$) is regarded as a random variable in a probability space, then this is the missing link in the chain of reasoning. For a random variable let $X : \Omega \to \mathbb{R}$ be a measurable function in a probability space defined by the triple $\Omega, \mathcal{A}(\Omega), Pr$, where $\mathcal{A}$ a σ-algebra, is a certain subset in the probability space. By inserting (2) in (1) we produce (3)

$$\mathcal{S}ec = 1 - \Big(Pr_E \times L\Big). \tag{3}$$

Thus, security can be measured indirectly by measuring the risk.

A quantification of a random variable (X) is performed formally by assigning a value (x) for a range of values (W) using a certain event (E). For the random variable (X) the image of a discrete probability space then applies to the discrete result set $\Omega = \{\omega_1, \omega_2, ..., \}$ such that $X : \Omega \to \mathbb{R}$. For discrete random variables for the discrete value range that is interpreted in the context of operational risk as a monetary loss (L) (*cf. Def. 2*)

$$L_X = W_X := X(\Omega) = \{x \in \mathbb{R} \mid \exists\ \omega \in \Omega \text{ mit } X(\omega) = x\}. \tag{4}$$

In the field of operational risks, the probability (Pr) with the random variable (X) which may accept certain values (W_X) and losses (L_X) is of interest. For any event (E) with $1 \leq i \leq n$ and $x_i \in \mathbb{N}$:

$$E_i := \{\omega \in \Omega \mid X(\Omega) = x_i\} = Pr[\{\omega \in \Omega \mid X(\omega) = x_i\}]. \tag{5}$$

Since, in this context, only numerical random variables are considered, each random variable can be assigned to two real functions. We assign any real number (x) the probability that the random variable takes that value or a maximum of such a great value. Then the function f_X with

$$f_X : \mathbb{R} \to [0, 1],\ x \mapsto Pr[X = x] \tag{6}$$

is called a discrete (exogenous) density (function) of X. Furthermore, a distribution function (F_X) is defined with

$$F_X : \mathbb{R} \to [0, 1],\ x \mapsto Pr[X \leq x] \sum_{x \in L_x\,:\, x' \leq x} Pr[X = x']. \tag{7}$$

The value (W_X) can have both positive and negative values, depending on which is discussed in the context, the density or the distribution of values.

Now if the confidentiality (*Conf*) and integrity (*Int*) are seen as discrete sets of random variables in a probability space, it is possible to describe these two security objectives (8), (9) as the sets of a given indicator function.

Due to the binary property of the two subsets (*Int*, *Conf*), with $Int \subseteq X$ and $Conf \subseteq X$, for every x on $[0, 1]$, which for $x \in X$ is 1 when $x \in Int$ or $x \in Conf$, otherwise 0. It is

$$X \to [0, 1],\ x \mapsto \begin{cases} 1, \text{ if } x \in Int \\ 0, \text{ otherwise.} \end{cases} \tag{8}$$

Also (8) can be used for the random variable *Conf*, if we used *Conf* instead *Int* in (8), then we can derive (9)

$$X \to [0, 1],\ x \mapsto \begin{cases} 1, \text{ if } x \in Conf \\ 0, \text{ otherwise.} \end{cases} \tag{9}$$

Thus, the binary properties of the two discrete random variables are formally described. In this paper we write $\mathbf{1}_{Int}$ to the discrete indicator function, integrity, and $\mathbf{1}_{Conf}$ to use the discrete indicator function confidentiality.

It is different with the availability (Av), which can be formally described as a *complete partial order, CPO*. With a CPO we can easily find intermediate values in the interval [0,1] in $\mathbb{R}^+$ which are the subject of a binary relation. A binary relation over the set (Av) availability of all elements is a partial order, if $a, b \in$ Av and $a \leq b$ holds. We use the following notation in this paper

$$(\text{Av} \leq) \mapsto [a \leq b] \text{ or short and sweet } (\text{Av} \leq). \tag{10}$$

Finally, a security concept ($\mathcal{S}ecCon$ (11)) is the illustration by the measures ($\mathcal{N}_{Ma}$) and with (8), (9) and (10) and the map to the method $\mathfrak{M}$ (*cf. Def. 1*)

$$\mathcal{S}ecCon\,(|\mathcal{N}_{Ma}|) := \mathfrak{M}\Big((\text{Av} \leq), \mathbf{1}_{\text{Int}}, \mathbf{1}_{\text{Conf}}\Big) \mapsto \mathbf{\Psi} \tag{11}$$

for a system $\psi \in \Psi$.

However, the security concepts are not only the power of the measures ($|\mathcal{N}_{Ma}|$) to reduce the risk of a possible injury of the three security objectives for a system, but it is necessary that the measures $\mathcal{N}_{Ma}$ have been developed using a methodology. This methodology is the function $\mathfrak{M}$ in (11). The function $\mathfrak{M}$ must be able to map the different risk types according to the underlying (open, closed, isolated) systems. Thus, the following definition is formulated for the measures.

Def. 3: The identified measures ($\mathcal{N}_{Ma}$), included in a security concept, based on the methodology ($\mathfrak{M}$).

The idea of the open (ψ_1), closed (ψ_2) and isolated systems (ψ_3) has been borrowed from thermodynamics, but can be

 ISBN: 978-1-61208-298-1

easily transferred to computer science and business, too [11], [12].

A broad representation of different types of risks, relating to systems all the way up to Individuums in Figure 2, has been marked by the no. (1) - (3), illustrated by T. Alpcan [13] and is the brainchild of N. Bambos.

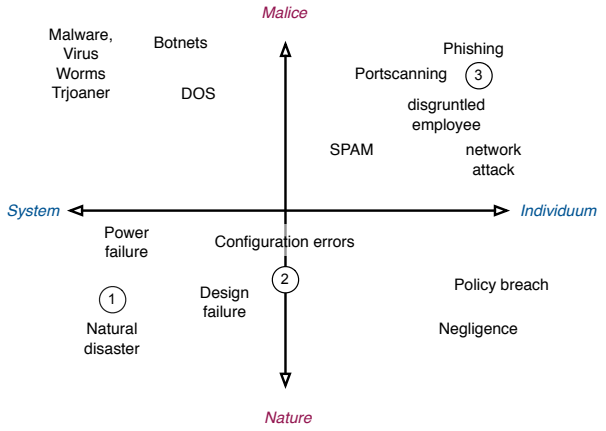


Figure 2: Different types of states and risks, according to N. Bambos [13]

Mark no. (1) denotes the *event risk*, for example, and this risk can be related to closed and / or isolated systems. No. (3), the purely *behavioral risks*, relates primarily to Individuums, as does the *hybrid risk* indicated by the no. (2). These are often considered using a scenario analysis. Hybrid risks are common in open systems.

After the different systems and risks are illustrated in Figure 2, the question becomes how to deal with the risks. It is not mandatory that every identified risk must be eliminated.

How to deal with risks is important for an organization. Risks generally could be reduced, avoided, transferred, accepted or eliminated (see Figure 3).

Economic aspects alone, and not technical aspects, are crucial in dealing with the identified risks. The different treatment methods are described below with the numbers 1 to 4. Several decisions have to be made regarding which of the risks can be avoided, reduced, transferred, or even accepted, see Figure 3.

The following are the decisions to be taken.

1) $R_1 = \sum_{i=1}^{n} R_{avoid}$, number of risks that can be avoided
2) $R_2 = \sum_{i=1}^{n} = R_{mitigate}$, number of risks that can be mitigated
3) $R_3 = \sum_{i=1}^{n} = R_{shift}$, number of risks that can be transferred
4) $R_4 = \sum_{i=1}^{n} = R_{taking}$, number of risks that can be accepted

Illustrated in Figure 3 are the upcoming decisions and these decisions are designated as the security posture. The ratio of overall risk and the various possibilities for reducing operational risks is shown.

The orientation of the possibility of a reduction of the risk depends on the cost. It is cheaper for the company to insure certain risks than to invest in adequate measures.

Companies may decide the quantification of risks according to their budget and their business objectives and their risk behavior (level of risk appetite). This turns risk management into cost management.

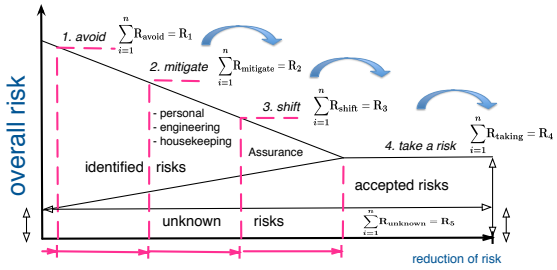


Figure 3: Aspects of risk treatment

It should be noted that in spite of a risk analysis unknown risks still exist. The goal of any risk analysis must, therefore, be to try to reduce the number of unknown risks $R_5 = \sum_{i=1}^{n} = R_{unknown}$ as much as possible. The unknown risks are not negligible, if we follow the groundbreaking book *The Black Swan* from N. Taleb [14]. He argues that the Gaussian bell curve can show probabilities, but the case of a devastating security event (Black Swan) will be an outlier to the bell curve, see Figure 4, and produce a worst case scenario.

III. Case study

Within this section, the next subsection (*A*) discusses hybrid risks from the Game Theory perspective. We argue that it is a game against nature. The second (*B*) and third (*C*) subsection discuss the Causa Stuxnet and analyze it using the trust game of the Game Theory. The solution achieved through a Nash equilibrium of the game analysis of the trust game used is presented in the fourth subsection (*D*).

A. Analyzing hybrid risks: a game against nature

The risks denoted by no. (2) in Figure 2 arise from both state risk and behavior risks. For this type of risk analysis, statistical methods and behavioral effects are both considered. This hybrid risk could be analyzed by the risk scenario technique going back to the three-point estimation method [15]. This three-point estimation method was used for the analysis of hybrid risks in the area of power plants and specifically in the field of SCADA systems. It was studied experimentally at 29 power plants, as one can read in [16]. Based on this analysis, a security concept for the SCADA system has been created.

Generally, a scenario is a possible event E_i, expressed formally in (5). It is the attribution of a certain value of a random variable ($X(\omega) = x_i$). In this context, an event E_i is understood as a risk event ($Rsz\varsigma$). Using the three-point (risk) estimate method, different loss probabilities (best case (BC), most likely case (mc), worst case (wc), (see Figure 4) of a risk event are identified. It relates the risk scenarios to the above protection objectives (Av $\leq$), $\mathbf{1}_{\mathrm{Int}}, \mathbf{1}_{\mathrm{Conf}}$. The risk of incident is related to an asset. The assumption is, an asset incorporates both a resource and a role that interact in a business process

[17]. (12) defines a risk event (RSz) with $\varsigma = \{bc, mc, wc\}$ as a possible result of variations of the risk event.

$$X(\omega) = Rsz_{\varsigma} :=$$
$$\begin{cases} if\,\varsigma = bc \,|\, (x_{bc} = x_{mc}) \wedge Pr[X(\omega) = x_{bc}] \rightarrow \text{best case} \\ if\,\varsigma = mc \,|\, (x_{bc} > x_{mc}) \wedge Pr[X(\omega) = x_{mc}] \rightarrow \text{most likley case} \\ if\,\varsigma = wc \,|\, (x_{wc} \gg x_{bc} \wedge x_{mc}) \wedge Pr[X(\omega) = x_{mc}] \rightarrow \text{worst case} \end{cases} \quad (12)$$

The possible result types (Rsz_{ς}) of a risk event were estimated by experts in workshops. An illustration of the stochastic process of (12) is presented in Fig. 4. Furthermore, a distribution (Gaussian curve) using three points of the estimates was created by a General Pareto Distribution [16]. This line shows the distribution of possible losses to absorb. In this case, the certain event $Pr = 1$ is no longer a stochastic event. In terms of operational risks, only the grey shaded area of interest is normally referred to as a downside risk with $X(\omega) = \{1, -\infty\}$. Assuming a time interval (t_1, t_3) in the probability space (Pr), the expected loss (VaR) can be determined for a confidence interval (α) using (13), which provides a lower bound

$$VaR_{\alpha} := min\{x \,|\, (Pr[X \leq x] > \alpha)\}. \quad (13)$$

The VaR is not a coherent risk measure, as demonstrated by

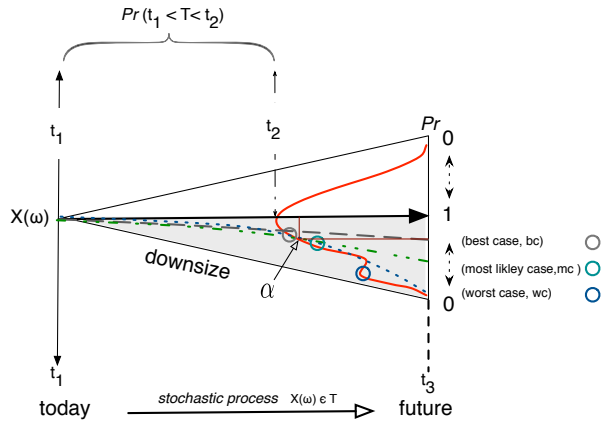


Figure 4: Risk corridor for the time interval (T)

Artzner [18], but, for this risk estimation using the VaR, the error made in this case is very small, because power plants use the standard BS25999 for the very rare risk with a catastrophic outcome [16].

After analyzing the risks created by the set $\mathcal{N}_{\mathcal{R}}$, the elements $\mathcal{N}_{\mathcal{R}} = \{r_1^{Rsz_{\varsigma}}, ..., r_{Rsz_{\varsigma}}^{Rsz_{\varsigma}}\}$ have the cardinality $|\mathcal{N}_{\mathcal{R}}|$. These can be addressed through appropriate measures. There are different measures possible. On the one hand, actions may be identified, when only one risk scenario works against one risk; on the other hand, other measures can be identified that counteract more than one risk. In general, measures are defined with the set $\mathcal{N}_{Ma}$ and the elements $\mathcal{N}_{Ma} = \{m_1^{Ma}, ..., m_{Ma}^{Ma}\}$. The cardinality is given with $|\mathcal{N}_{Ma}|$. These measures, based on the three security objectives (8), (9, (10), create the security concept according to (11).

Through risk analysis, using the risk scenario technology, which refers to assets in a process (business process), both a pure state risk and a behavioral risk are included, because in the scenarios unconscious and conscious actions (misuse) of an employee and its impact on the business process are considered.

From the perspective of Game Theory, this is still a game against nature. This is a decision problem $\mathcal{D}$ in strategic form under risk. They are making decisions for actions involving the different probabilities (Pr) in the probability space given for the environment states $z \in Z$. However, the classical decision rules (MaxiMin rule, MaxiMax rule, Laplace's rule, etc.) are not used as strategies in this paper. The decision maker who is responsible for developing a security concept (see (11)) will choose a strategy s from the set of all strategies S, to select those measures to (avoid, decrease, transfer, eliminate, accept) a risk event (see (12)). Five different strategies, $\tilde{s}_1, ..., \tilde{s}_5$, can be used

$\tilde{s}_1$ = Avoiding the outcome of the risk with a measure.
$\tilde{s}_2$ = Decreasing the outcome of the risk with a measure.
$\tilde{s}_3$ = Transferring the risk to an insurance company.
$\tilde{s}_4$ = Eliminating the outcome of a risk with a measure.
$\tilde{s}_5$ = Accept the risk.

Not all of the strategies listed above for $\tilde{s}$ are applied, because each measure requires certain costs ($\mathcal{U}$). In classical Game Theory, $u \in \mathcal{U}$ is often understood as the pay-off (function or utility function). We described $\mathcal{U}$ with the amount of the costs of all measures and u is a cost function, the decision problem $\mathcal{D}$ is a strategic decision

$$\mathcal{D} = \left(\tilde{S}, Z, \varsigma, \mathcal{U}, u\right) \quad (14)$$

under risk. (ς) represents a probability distribution on Z, the environmental conditions. The creation of the decision space (14) can be represented as follows

$$\tilde{S} \times Z \mapsto \mathcal{U}(\varsigma). \quad (15)$$

A decision matrix can be derived from the decision space, as illustrated in Table I. The decision maker (security officer) has to make a decision based on the decision matrix of Table I that included the strategies, the environmental conditions and the measures or the costs of the activities related to the risk scenario ($\varsigma = \{bc, mc, wc\}$), which should be provided in the security concept (see (11)). The decision matrix is shown above in Table I. Depending on the decision process by the security officer (see (14) and Table I) this will meet the security concept regarding the security objectives of confidentiality (9), availability (10) and integrity (8) of the identified risks with appropriate measures. With the consideration of the hybrid risks posed by the scenario technology (see (12)), consolidated findings are gained to complete the security concept. However, analysis of the hybrid risks with the scenario technology is not ideal for analyzing the Stuxnet virus.

 ISBN: 978-1-61208-298-1

Table I: CHANCE MOVES AGAINST NATURE

		nature / environment		
		z_1	z_2	z_3
security officer	$\tilde{s}_1$	$u(\tilde{s}_1, bc)$	$u(\tilde{s}_1, mc)$	$u(\tilde{s}_1, wc)$
	$\tilde{s}_2$	$u(\tilde{s}_1, bc)$	$u(\tilde{s}_1, mc)$	$u(\tilde{s}_1, wc)$
	$\tilde{s}_3$	$u(\tilde{s}_1, bc)$	$u(\tilde{s}_1, mc)$	$u(\tilde{s}_1, wc)$
	$\tilde{s}_4$	$u(\tilde{s}_1, bc)$	$u(\tilde{s}_1, mc)$	$u(\tilde{s}_1, wc)$
	$\tilde{s}_5$	$u(\tilde{s}_1, bc)$	$u(\tilde{s}_1, mc)$	$u(\tilde{s}_1, wc)$

There is no analysis of the behavior (actions) of employees or other service providers. This has led to the conclusion that the existing security concepts in the power plants have a gap, and that the infection of an authorized service technician – *albeit unwittingly* – is possible.

In the next subsection, we analyze the infection of the Stuxnet virus to compromise the protection target ($\mathbf{1}_{Int}$) with the trust game of the Game Theory.

B. Game analysis before the Stuxnet virus arose

Pure behavioral risks (*cf.* no. (3) in Fig. 2), in contrast to pure state risks (*cf.* no. (1) in Figure 2), could not be analyzed with statistical methods.

Therefore, we consider the Causa Stuxnet and the behavior between the service technician and the staff (security officer) causing the infection using the trust game (*note:* The trust game is a modified dictator game). from Game Theory. As one of the first, [19] deals with the trust game in reference to a social environment. Typically, for the trust game, there is a different trust relationship (imbalance) between the two players.

These behavioral risks are the types of decisions (strategies) of the player (service technician / security officer) that caused the infection.

Stuxnet — *The infection of Stuxnet virus, in the area of critical infrastructure (SCADA) systems, has not been an attack. The virus was transferred from a service technician equipped with the necessary permissions unconsciously, using an infected USB stick. The virus has arrived without the knowledge of the service technician on to his USB stick.*

Subsequently we analyze this critical incident with the Game Theory to derive a solution from the chance moves of the game. This solution leads into a cooperation of the software manufacturer with the power plant operator of the SCADA systems.

In the analysis, we use a slightly modified version of the trust game, because it cannot be ruled out that tomorrow another service technician from another company with a similar virus attends to the power plant.

Pure behavioral risks, which are designed to trust, could be analyzed with the trust game [20]. We analyze the chance moves of both players. Each player reacts to the behavior of the other player. One of the basic ideas of Game Theory is to study, analyze and evaluate the reciprocal response pattern of the players. Reciprocal reaction patterns, so-called pay-offs, are determined by distribution rules and play a significant role and in turn, depend on the incentives. It depends on the distribution rules of legal, contractual, historical or political power relations. Thus, a major difference is the probability models, as these know no incentive mechanisms.

Table II: CHANCE MOVES BEFORE THE STUXNET

		σ_2	
		infect	not infect
σ_1	trust	1, 0	3,3
	don't trust	1, 0	2, 2

Game analysis of virus Stuxnet pursues a causal chain of thought. First the chain of thought which was taken before the Stuxnet virus (*cf.* Table II and Fig. 5) is followed and another chain of thought follows after the onset of Stuxnet virus (*cf.* Fig. 6 and Table IV). The concerned chance moves are performed as a *one-shot game*. Thought chains are typically illustrated in the form of branching trees, to represent the individual moves. Another name for the game tree is the *extensive form* as is noted in [5].

Formally, a strategy game Γ consists of a triple. With Σ, the set of players σ is defined, it is $\sigma \in \Sigma$. With S the set of strategies is described and we have $s \in S$. This means that a game can be characterized as follows

$$\Gamma = \{\Sigma, S, \mathcal{U}, u\}. \tag{16}$$

$\mathcal{U}$ has the same intention as in (14). In the analysis of the Stuxnet virus, are two players (σ_1, σ_2) in the space of action $A = \Sigma \times S$. The action space (*cf.* Fig. 5) for player σ_1 ist $A^{\sigma_1} = \{t, nt\}$ and $A^{\sigma_2} = \{i, ni\}$ applies to player σ_2. In this analysis, pure strategies are postulated; therefore, a single pure strategy is expressed in s. Strategies include decision rules that the player implement to some benefit (u) to obtain a pay-off. In general, the trust game can be expressed as

$$\Sigma \times S \mapsto \mathcal{U}(u). \tag{17}$$

Compared with (17), in (15) the players S now are in the place of the environmental states Z.

Typically, games in the form of a bi-matrix, called the simultaneous logical reasoning circular, are presented in a sequential chain of thought the game tree. The Bi-Matrix in the trust game between the service technician (player σ_2) and the security officer (player σ_1) has been formulated in Table II. In this situation, this game represents the situation before the virus Stuxnet arrived. Before Causa Stuxnet, there was *no* distrust due to player σ_2 (service technician) and, consequently, the chance moves (1, 3, 6) in Fig. 5 are typical chance moves. To date, no incidents justified distrust of players σ_1 (security officer) toward players σ_2. Also, it was inconceivable to date, that a special virus [8] would be written which is used in an area with property software for the small *Program* Logic Controller (PLC), *cf.* with the Step 7 software. However, for a suspicious player σ_1 (security officer), the move (1, 2, 4) of Fig. 5 is also conceivable, but impossible because thus far virus infections were not encountered and therefore without

 ISBN: 978-1-61208-298-1

consequence. It also ruled out the usual (normal) route of infection in the power plant, due to the systematic separation of networks and hermetic sealing of the internal systems to the Internet and intranet. Thus, the combination *(do not trust / not infected)* is a Nash equilibrium. In a Nash equilibrium, none of the two player could obtain a better position through a change in their attitude.

In this respect, the cost function u (not much effort, because there are no security policies to follow) is greatest for the two players when combining *(trust / not infected)* in Table II. It leads also to a Nash equilibrium, since neither player can achieve a better position by changing moves.

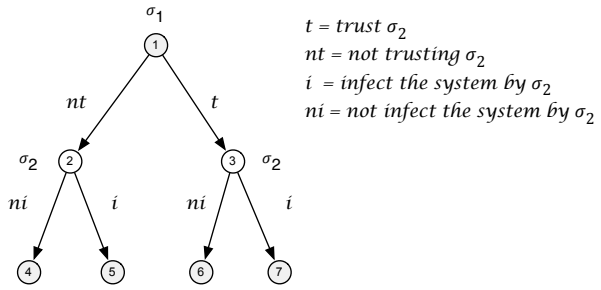


Figure 5: Change moves in a game tree before the Stuxnet virus arose

After the Stuxnet virus arose, this event changed the trust relationship drastically between player σ_1 (security officer) and player σ_2 (service technician). This change in position of trust is analyzed in the next subsection.

C. Game analysis after the Stuxnet virus arose

After the Stuxnet arose, the perspective of player σ_1 (security officer) has changed considerably. He does not know whether or not the service technician σ_2 brings an infected USB stick. The result is the typical trust game (*note:* This uncertainty could also be analyzed with a mixed strategy, a probability distribution over the pure strategy. However, in this investigation we use only pure strategies). situation because an imbalance of trust has occurred.

After the Stuxnet virus, the security officer (σ_1) could continue to trust the service technician (σ_2) or install comprehensive security policies. The behavior of σ_1 *trust* (t) is illustrated in the extensive form (see Fig. 6) in the right branch (1, 3). The player σ_2 then has the opportunity to infect the system (1, 3, 7) or not (1, 3, 6) with the result that σ_1 must check the system (c, nc) or possibly report a virus (r, n).

The left branch illustrates, on the other hand, the behavior strategy of σ_1 for *do not trust* (nt), therefore the game play (1, 2).

Security policies always increase the restrictions and the workload involved for everyone (σ_1, σ_2). Furthermore, it is evident that, when security policies are perceived as too restrictive by the players, they bypass (σ_1, σ_2) all of the policies. For the security policy for the USB stick, this leads to (bc, nbc). This realistic situation is illustrated by the extensive form in Fig. 6. If the security policies are not undermined, there is the game branch (1, 2, 5, 10). However, the branch with restrictions, imposed by the security policy, increased the workload.

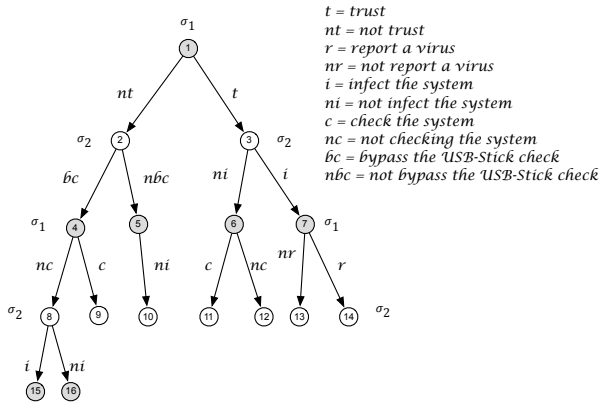


Figure 6: Change moves in a game tree after the Stuxnet virus arose

The game branch (1, 2, 4) represents the case where the service technician (σ_2) bypasses the security policy unnoticed and the security officer (σ_1), driven by their distrust, reviews the system for viruses (1, 2, 4, 9). This distrust does, however, again cause an increased effort for σ_1. Otherwise, the strategy (1, 2, 4, 8) is followed by σ_1 and a significant degree of uncertainty remains about the state of the system. It may be now, that the game play (1, 2, 4, 8, 16), or in the negative case, an infection occurs (1, 2, 4, 8, 15). As a result it can be stated that no Nash equilibrium can be achieved.

The game tree of Fig. 6 allows us to derive the bi-matrix of the Table IV with the pay-off Table III and the key parameters for the service technician (σ_2) and security officer (σ_1). In the following statements, the key parameters for the service technician (σ_2) are listed.

- G_S Benefit for successful service
- G_B Benefit to following the security policies (increasing of reputation)
- K_S Investment given by checking the security controls
- K_R Penalty for ignoring the security policies
- K_U Penalty for procedure to disinfect the system

For the security officer (σ_1), the following key parameters are provided:

- G_E Benefit given for the fact that the system was not infected by the maintenance procedure
- G_N Benefit given for the fact that the maintenance procedure was done without any problems
- G_V Benefit given for the fact that the maintenance procedure was done successfully and in a safe manner
- K_V Penalty in the case of a violation of the security policy

The payoffs are taken from a *real world assumption* and reflect observations in dealing with the service technician in a power plant.

The payoffs of the game matrix for the security officers and service technicians are illustrated in the Table III. If the payoffs in Table III are taken into account in the bi-matrix,

 ISBN: 978-1-61208-298-1

Table III: PAYOFF FOR BOTH PLAYERS

σ_1		σ_2	
G_S	6	G_E	3
G_B	6	G_N	4
K_S	3	G_V	5
K_R	8	K_V	11
K_U	2		

Table IV: MOVES IN A GAME AFTER STUXNET VIRUS

		σ_2	
		infect	not infect
σ_1	trust	$G_V - K_V$, $G_B - K_S - K_U$ (-6, 1)	$G_E + G_V$, $G_S - K_R$ (8, -2)
	don't trust	G_N, $G_S - K_S - K_V$ (4, -9)	G_E, G_S (3, 6)

the following Table IV is created. An appropriate strategy for both players is not apparent. It is also clear that for the current practice of using a virus scanner, no Nash equilibrium is appropriate and that it, at best, is only a stopgap.

D. Game of cooperation to find a solution for the Causa Stuxnet

In the previous subsection, it appears that the use of a virus scanner is only a stopgap measure in the sense of Game Theory. The game situation has been changed. In this subsection we will therefore, in the analysis of the Cause Stuxnet, initiate a modified game, which requires a collaboration of players. Then, a Nash equilibrium could be established and the utility function for the players is increased. Therefore, a pure strategy was sought that minimized the amount of costs (K_S, K_R, K_V, K_U) and correspondingly increased the value $\mathcal{U}$. The costs and benefits of the different values is still listed in Table III. These values have *not* changed after using a signature. Minimizing the cost and the strategy s is listed in (18)

$$min\,\mathcal{U}(s, K_s, K_R, K_V, K_U). \tag{18}$$

As a pure strategy, in terms of a cooperation between the two players, it means a lesser amount of work (benefits) for both players and the two companies. The effort must keep both sides balanced. This balance and the Nash equilibrium arise when the software manufacturer creates the SCADA software with a signature over a hash value and ensures that the signature was generated using the original software. Then, the signature was provided to the power plant operator. This change will change the behavior of the players in the Table IV with the same payments (benefits) in the Table III. The behavior of the two players with the appropriate response to the use of the signature is given in Table V.

The difference between Table III and Table V is apparent in the field *(don't trust / not infect)*. The use of the signature on both sides (producers and users) increases the benefit and the values (K_V, K_S, K_R, K_U) do not occur. Thus, it is possible for the field (12, 12) to obtain a Nash equilibrium. The game tree in Fig. 7 displays the game. The moves (1, 2, 3, 4) show the course.

Table V: MOVES IN A GAME AFTER STUXNET WITH AN IMPLEMENTED SIGNATURE

		σ_2	
		infect	not infect
σ_1	trust	$G_V - K_V$, $G_B - K_S - K_U$ (-6, 1)	$G_E + G_V$, $G_S - K_R$ (8, -2)
	don't trust	G_N, $G_S - K_S - K_V$ (4, -9)	$G_E + G_N$, $G_S + G_B$ (12, 12)

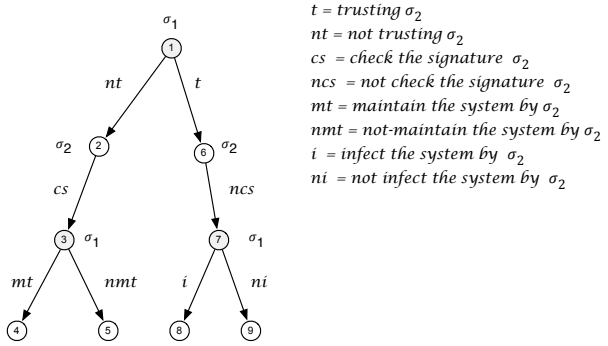


Figure 7: Moves in a strategic game in a game tree with a pre-exchanged signature

Thus, the measure (*use a signature*) met the two above-mentioned definitions 1 and 2, according to (11), and can be included in a security concept.

In essence, the strategic moves of the game in Table V and in Fig. 7 are only possible because a cooperative strategic game between the software manufacturers of SCADA systems and the power plant operators was recently initiated. A cooperative strategic game Γ consists of a tuple

$$\Gamma = \{N, v\}. \tag{19}$$

$N = \{1, 2\}$ is understood as the software manufacture (1) and the power plant (2). With

$$v \in \mathbb{V}(N) := \{f : 2^N \mapsto \mathbb{R} \,|\, f(\emptyset) = 0\} \tag{20}$$

being the characteristic function. The coalition function maps a value to each coalition. For example, if only one of the two coalition partners applied the signature, the result is given by $v(\{1\}) = v(\{2\}) = 0$. If the signature is used as described above by both, then $v(\{1\}) = v(\{2\}) = 1$. Only when both partners stick to the coalition is a benefit obtained, as one can see in the field *(don't trust / not infect)* in Table V. For the cooperative game, the Nash equilibrium is achieved. Should it not come to the coalition, the power plant operators may only use a virus scanner.

IV. Related work

The behavior of attacks on IT systems were studied in the Honeynet Project by M. Spitzer for first time [21] and has since received a lot of attention in the literature. The Honeynet Project, at the time, sought a novel approach in which the behaviors of the attacker were studied in order to develop them into conclusions for the protection of IT

systems. The behavior of the attacker was analyzed, but not with the methods of Game Theory. In the paper by W. Boehmer, human behavior was studied by entitling employees to perform unauthorized actions if necessary in a company. The method used is coupled to forensic analysis using data mining techniques [22]. Profiling was performed to identify the possible unlawful conduct by employees. The above examples did not use a game-theory approach. In the area of networking, T. Alpcan investigates attacks using game theory analysis. This game theoretical approach gave deep new insight into the defense of networks [13]. Based on the spy / inspector game, evidence was obtained from Alpcan. However, game theoretical methods have had hardly any or very little, but not systematic use in the field of IT security. In the case of the Stuxnet virus, infection was performed by a certified maintenance staff unconsciously and thus deviates from the usual spy / inspector game. All here above described methods of analysis would not reflect the infection realistically. We analyze the Causa Stxunet, or better the path of infection, therefore, using the trust game, to reflect the realistic situation.

V. Conclusion and further investigations

In this paper, we have studied the Stuxnet virus using Game Theory. In the multidimensional game of the Causa Stuxnet we have analyzed a part of the whole game, especially – *the infection path with the Trust / Investor game* – to compromise the security objective ($\mathbf{1}_{Int}$) of the SCADA system. From the game analysis, we derived that only a Nash equilibrium can be established if a collaboration is sought between the software manufacturer for SCADA systems and software users in the power plant. The cooperation is made possible by the implementation of a signature on the SCADA software from the software manufacturer in his laboratories. This signature can be checked very easily at the users site in the power plant, when a service technician arrives. This solution presented by the signature is a preventative solution and should be preferred to the current reactive solution of the virus scanner.

The path of infection is only one part in the multi-dimensional game, because there is an attacker in the background with the intention of damaging the power plants (compromise the security objective availability (Av $\leq$), see eq. 10). Against this background, the game analysis of the path of infection is only a part of a game in a multidimensional game. The analysis of the entire game of Causa Stuxnet and the embedding of this game into the whole game, will be part of another investigation.

References

[1] D. E. Denning, "Stuxnet: What has changed?," */Future Internet/*, vol. 4 (3), pp. 672–687, published online: 16 July 2012 / last accessed 2013-07-15/ doi:10.3390/fi4030672.

[2] N. Falliere, L. O. Murchu, and E. Chien, "Symantec security response, w32.stuxnet dossier, version 1.4." http://www.symantec.org., 02 2011.

[3] C. Salter, O. S. Saydjari, B. Schneier, and J. Wallner, "Toward a secure system engineering methodolgy," in *Proceedings of the 1998 workshop on New security paradigms*, NSPW '98, (New York, NY, USA), pp. 2–10, ACM, 1998.

[4] B. Schneier, "Attack trees, modeling security threats," http://www.schneier.com/paper-attacktrees-ddj-ft.html, 1999.

[5] F. Carmichael, *Guide to Game Theory*. Pearson Education Limited, ISBN 0 273684965, 2005.

[6] B. Kordy, S. Mauw, M. Melissen, and P. Schweitzer, "Attack-defense trees and two-player binary zero-sum extensive form games are equivalent," pp. 245–256, 2010.

[7] K. D. Mitnick and W. L. Simon, *The Art of Deception, Controlling the Human Element of Security*. Wiley, New York NY et. al., 2002.

[8] ENISA, "Stuxnet analysis, http://www.enisa.europa.eu/media/press-releases/stuxnet-analysis," 10 / 2010.

[9] W. Boehmer, "Dynamic systems approach to analyzing event risks and behavioral risks with game theory," *PASSAT/SocialCom 2011, Privacy, Security, Risk and Trust (PASSAT), 2011 IEEE Third International Conference on and 2011 IEEE Third International Conference on Social Computing (SocialCom), Boston, MA, USA, 9-11 Oct., 2011*, 2011.

[10] R. Giacometti, S. Rachev, A. Chernobai, and M. Bertocchi, "Aggregation issues in operational risk," *Journal of Operational Risk*, vol. 3, no. 3, 2008.

[11] F. Capra, *The Turning Point: Science, Society, and the Rising Culture*. Bantam, 1984.

[12] F. Capra, *The Web of Life: A New Scientific Understanding of Living Systems*. Anchor Books/Doubleday; 1st edition, 1996.

[13] T. Alpcan and T. Basar, *Network Security - A Decision and Game-Theoretic Approach*. Cambridge University Press, 1. ed., 2011.

[14] N. Taleb, *The Black Swan*. Random House, 2007.

[15] G. Dukic, D. Dukic, and M. Sesar, "Simulation of Construction Project Activities Duration by Means of Beta Pert Distribution," *in Information Technology Interfaces, 2008. ITI 2008. 30th. International Conference on*, pp. 203 – 208, 2008.

[16] W. Boehmer, *Information Security Management Systems Cybernetics*. *in* Strategic and Practical Approaches for Information Security Governance: Technologies and Applied Solutions, (ed.) M. Gupta, J. Walp, R. Sharman, IGI Global publisher of the Information Science Reference, 2011.

[17] JTC 1/SC 27/WG 1, "ISO/IEC 27001:2005, Information technology - Security techniques - Information security management systems - Requirements." Beuth-Verlag, Berlin, 10, 2005.

[18] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath, "Coherent measures of risk," *Mathematical Finance*, no. 9, pp. 203–228, 1999.

[19] J. Berg, J. Dickhaut, and K. McCabe, "Trust, reciprocity, and social history," *Games and Economic Behavior*, no. 10, pp. 122–142, 1995.

[20] T. Basar and G. J. Olsder, *Dynamic Noncooperative Game Theory*. No. 23, Society for Industrial and Applied Mathematics, Academic Press, New York, 2nd. ed., 1998.

[21] L. Spitzner, "Honeypots: Catching the insider threat," in *ACSAC '03: Proceedings of the 19th Annual Computer Security Applications Conference*, (Washington, DC, USA), p. 170, IEEE Computer Society, 2003.

[22] W. Boehmer, "Analyzing Human Behavior with Case Based Reasoning by the help of Forensic Questions." 24th IEEE International Conference on Advanced Information Networking and Applications (AINA-2010), 03 2010.

 ISBN: 978-1-61208-298-1

CAVEAT: Facilitating Interactive and Secure Client-Side Validators for Ruby on Rails applications

Timothy Hinrichs*, Michael Cueno*, Daniel Ruiz*, V.N. Venkatakrishnan* and Lenore Zuck*
*Dept. of Computer Science
University of Illinois at Chicago
Chicago, IL 60607 USA
Email: hinrichs, mcueno2, druiz22, venkat, zuck@uic.edu

***Abstract*—Modern web applications validate user-supplied data in two places: the server (to protect against attacks such as parameter tampering) and the client (to give the user a rich, interactive data-entry experience). However, today's web development frameworks provide little support for ensuring that client- and server-side validation is kept in sync. In this paper, we introduce** CAVEAT† **, a tool that automatically creates client-side input validation for Ruby on Rails applications by analyzing server-side validation routines. The effectiveness of** CAVEAT **for new applications is demonstrated by developing three custom apps, and its applicability to existing applications is demonstrated by examining 25 open-source applications.**

***Keywords*—Web applications, Data validation, Frameworks**

I. INTRODUCTION

Interactive processing and validation of user input are increasingly becoming the de-facto standard for Web applications. With the advent of client-side scripting there has been a rapid transition in recent years to validate user input in the browser itself, before it is submitted to the server. This client-side validation offers numerous advantages, among which are faster response time for users and reduction of load on servers. Yet, client-side validation exposes new vulnerabilities since malicious users can circumvent it and supply invalid data to the server. To defend against these so-called *parameter-tampering* attacks, the server must therefore perform data validation that is at least as strict as that of the client. The practical problem with this situation is that the client and server are often written in different programming languages, thereby requiring the development team to maintain two separate code bases that implement similar but not identical functionality, a notoriously difficult problem.

Several recent studies [1], [2], [3], [4] have uncovered parameter tampering vulnerabilities in both open source and commercial websites, most notably in websites for banking and on-line shopping, as well as those accepting payments through third party cashiers (such as PayPal and AmazonPayments.) These vulnerabilities enable takeovers of accounts and allow a malicious user to perform unauthorized financial transactions.

Ruby on Rails (RoR) has recognized the importance of server-side validation and includes special machinery to make the development and maintenance of server-side data validation especially simple. RoR includes several built-in routines that perform common data validation (e.g., checking that a field is numeric), and a developer only needs to declare which of those routines ought to be applied and to which data elements. For any validations not covered by these built-in routines, the developer extends the validators available in RoR by writing custom code. However, RoR fails to provide machinery that makes client-side validation as easy as server-side validation. Developers wanting to implement both client- and server-side validation must still write and maintain those validation routines in two separate code bases.

One popular extension to RoR, called `client_side_validations` [5], simplifies the creation of client-side validation by analyzing the built-in validations of a RoR application and automatically replicating those validations on the client. Each time a user enters a piece of data on one of these `client_side_validations`-enabled web forms, the validation routines immediately check for errors and signal them to the user, despite the fact that the developer implemented no client-side validation at all. This paradigm of copying server-side validation to the client is advantageous because it incentivizes the developer to spend time and energy perfecting the server-side validation code (which is necessary to protect the server against parameter-tampering attacks), achieves the desired client-side validation, and avoids the problem of manually maintaining two separate code bases with similar functionality. The main limitation of `client_side_validations` is that it fails to move the custom validators written by the developer over to the client.

In this paper, we present the design and implementation of CAVEAT†, a tool for analyzing and translating custom Ruby on Rails validators from the server to the client automatically. Developers who plan to employ CAVEAT write their custom validators in a fragment of Ruby that admits stronger compile-time analysis than the full Ruby language. The developer also follows several new conventions to guarantee that CAVEAT is sound. The developer can also provide hints to CAVEAT to control which server-side validations to move to the client, thereby leveraging application-specific information known only to the developer.

This paper makes the following contributions:

- Identification of a fragment of RoR custom validation that admits strong compile-time analysis.
- Design and implementation of a tool for replicating server-side validation on the client.
- Conventions for RoR that ensure the tool's soundness.

†Compiler For Automated Validation Extraction Analysis and Translation

 ISBN: 978-1-61208-298-1

- A comparison of 25 existing Rails validators against our fragment of RoR validation.
- Demonstration of the tool on 3 RoR applications.

The rest of the paper is organized as follows. After an overview including a running example (Section II), we describe our approach and the challenges of replicating server-side validation on the client (Section III). Next we describe our implementation (Section IV) and discuss our evaluation (Section V). Finally we compare CAVEAT to related work (Section VI) and conclude (Section VII).

II. Background

Ruby on Rails is a web application development framework written in the Ruby programming language. The goal of the framework was to make web programming easier, faster, and more productive. It comes standard with various sane defaults, assumes various conventions that users must also follow, embraces the REST pattern and HTTP verb semantics, highly promotes DRY or "Don't Repeat Yourself", and at its core, is fundamentally based on the Model View Controller (MVC) architecture.

The MVC architecture is centered around separating the concerns between the representation of information (the Model), the interaction with that information (the Controller), and its presentation (the View). Models usually wrap an interface around a database table. This interface usually consists of methods that contain the application's business logic. Views are the user interface of the application that, in the context of a web application, get sent to the browser. They are usually written in HTML and some templating code such as embedded Ruby (ERB) or HAML, and contain only functionality needed to present the information. Controllers are what bridge the connection between a View and a Model. They handle the incoming requests, query the models as needed, pass the resulting data on to the view, and return it as the response.

A. Running Example

Many web applications manage user information, e.g., people create new user accounts, provide information like name, email, and address, edit profiles when they become out of date, and delete users along with their profiles from the system. Figure 1 is a diagram depicting the profile information associated with each user. (We use this as a running example that explains our ideas through the paper.)

Every time a new user is created, the application needs to ensure that the profile information meets the following conditions:

- the username has not already been chosen;
- the name must be no longer than 80 characters;.
- the birthdate must be a valid combination of month-day-year;
- the user must be at least 18 years old;
- the two passwords must match;
- the email address must be properly formatted.

These requirements are typical of web applications that require users to create their profiles.

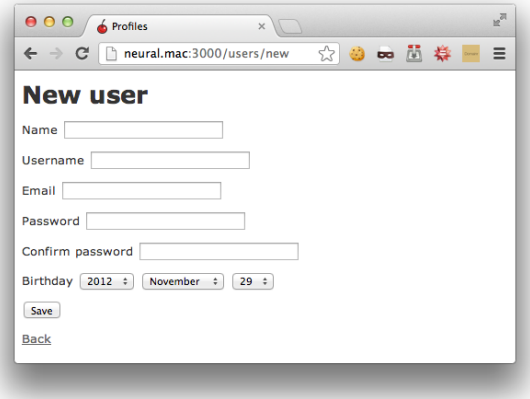


Figure 1: New User Form

B. Example in Rails

Rails has built-in machinery for constructing a significant portion of the web application for creating, retrieving, updating, and deleting (also known as CRUD) the basic objects of the application, e.g., user profiles. It even has built-in machinery for checking that those objects satisfy certain constraints, such as the user being over 18 years old.

In Rails, building CRUD functionality for a User object begins with a definition for the User model, which is given in Listing 1. The developer writes down the fields for the User object (in the line `attr_accessible`) along with statements dictating that those fields must satisfy conditions.

```
class User < ActiveRecord::Base
  attr_accessible :name, :username, :born_on,
    :email, :password, :password_confirmation
end
```

Listing 1: The User Model

The developer also writes Views for displaying the User model to the user (both for creating and editing a User object). The developer also writes Controllers that accept HTTP requests for creating a new user or editing an existing user, validates the User data against the required constraints, and either saves the data to the model or rejects the data and sends the user back error messages. Rails simplifies the process of writing such controllers by automatically validating a User object's data when it is saved to the database, accumulates any error messages that describe why a validation failed, and displays those error messages to the user. For example, Figure 2 shows the response of the Rails app when the submitted data fails the validation. Since the errors are stored within the User model passed to the view, the view can then use that information to display error messages.

C. Rails Validations

Rails includes a variety of built-in validation methods but allows the developer to write custom validators as well. The built-in validators since Rails version 3.0.0 are listed in Table I and can be configured to only be triggered for certain states in

 ISBN: 978-1-61208-298-1

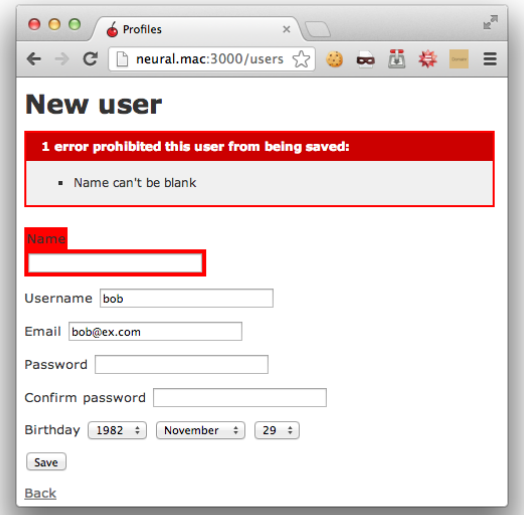


Figure 2: Example of Validation Failure

the record's lifecycle, such as creation, update, or save. Some built-in validators can even be configured to trigger only when a given function returns true.

In our running example, we can use the `presence` validator to ensure that `:name`, `:username`, and `:email` are always provided by the user. A developer would add the following code to the User model.

```
validates :name, :username, :email, presence
    : true
```

Listing 2: Presence Validation on User attributes

While convenient, Rails' built-in validators are not sufficient for every situation and hence Rails allows the developer to write custom validators that are treated similarly to its built-in validators. Just like built-in validators, custom validators are executed each time an object is saved to the database. For example, to ensure that all users are at least 18 years old, a developer could include the code in Listing 3 in the User model.

```
validate :ensure_not_minor

def ensure_not_minor
  unless born_on <= 18.years.ago
    errors.add :born_on, 'Must be at least
        18 years old to use this site.'
  end
end
```

Listing 3: User Model :born_on validation

Rails allows a developer to write validators for individual fields of an object or for the entire object and provides specialized machinery for both cases (`ActiveModel::EachValidator` and `ActiveModel::Validator`, respectively).

III. Approach

CAVEAT transforms a Rails application with server-side validation into a Rails application with both client- and server-side validation. To use CAVEAT, a Rails developer writes her application while obeying a few additional conventions, and CAVEAT automatically analyzes that application, generates JavaScript to implement client-side validation, and installs that JavaScript into the appropriate views. The CAVEAT-instrumented web forms provide the user with real-time feedback and ensure that every form submission passes all the client validations. That is, each time the user edits a form field, the JavaScript runs all the validations pertinent to that field, and alerts the user to any errors; each time the user submits the form, it allows the submission only if there are no outstanding errors. Below we outline the main challenges in transforming web applications in this way as well as the CAVEAT architecture that addresses those challenges.

A. Challenges

Identifying validation code. Web applications can choose how and where to validate input data. In `PHP` for example, validation code can be interspersed with code for generating HTML, manipulating the database, and changing the session. Ruby on Rails makes the task of identifying which code is related to validation much simpler because developers are expected to write validation code in one of a handful of ways. Thus one of the reasons CAVEAT targets Ruby on Rails is that identifying the code responsible for validating input data only requires understanding the Rails infrastructure, as opposed to performing sophisticated source-code analysis.

Translating server-side validation code into client-side validation. This challenge is by far the most complex. Most obviously, the language the server-side validations are written in, Ruby, is different from the language the client-side validations are written in, JavaScript, and hence there must be translation at the basic level of the programming language. But there is a more fundamental problem in the translation that a Ruby-to-JavaScript compiler would not itself be able to solve: Rails validators are written assuming that the user has completely filled out the web form—that all the data the user will provide has already been provided. In contrast, the client-side validations are often run before the user has completed filling out the form and thus must account for the fact that some data may be unknown at the time the validation is run. Clients that ignore the possibility that some data is unknown can produce surprising and/or unsatisfactory behavior. For example, suppose the server requires values for all of the fields on the form. If the client simply replicated this validation check, then as soon as a user entered a single value in a single field, all the remaining fields would be highlighted as errors.

To provide satisfactory, real-time feedback to the user, the client must be careful to validate only those fields with values. Given a collection of Ruby validations and a Ruby-to-JavaScript translator, the naïve approach to generating client-side validation is to generate one JavaScript validation routine for each Ruby routine and add a guard to each JavaScript validation routine so that it runs only if the form fields mentioned in the validator have values. For forms where the

 ISBN: 978-1-61208-298-1

Table I: Rails built-in validators

Validator	Description	Example
acceptance	Validates that checkbox on UI was checked when form submitted.	`validates :age_requirement, acceptance: true`
validates_associated	Validates model's other associated models.	`has_many :posts validates_associated :posts`
confirmation	Validates that two attributes have the exact same values	`validates :password, confirmation: true validates :password_confirmation, presence: true`
exclusion	Validates that attribute's value is not part of given set.	`validates :username, exclusion: { in: %w(admin manager user) }`
format	Validates that the attribute's value conforms to given regex.	`validates :email, format: { with: /\A([^@\s]+)@ ((?:[-a-z0-9]+\.)+[a-z]{2,}\z/i }`
inclusion	Validates that the attribute's value is part of a given set.	`validates :tag inclusion: { in: %w( security development rails ) }`
length	Validates that the attribute's value conforms to some length constraint	`validates :username, length: { maximum: 80, minimum:3 } validates :password, length: {in: 10...100}`
numericality	Validates that the attribute's value is numerical.	`validates :favorite_number, numericality: true`
presence	Validates that the attribute's value is not blank or nil.	`validates :username, presence: true`
uniqueness	Validates that the attribute's value is unique immediately before being saved.	`validates :email, uniqueness: true`

validation for each field is independent of all the other fields, this approach works; however, if the fields interact this basic approach fails to detect all of the validation failures in real-time. For example, consider a validation for three boolean fields: a, b, c, which the server requires to all have values. In addition, suppose the following validations are written in Ruby.

```
if a and !b then error
if b and !c then error
```

It is straightforward to translate these two validators to the client and add guards that ensure each validator is only run when the fields occurring within it have values. Now suppose the user sets a to True and c to False but sets no value for b. Since b appears in both validators, neither one is executed and the web form signals no error to the user. Yet there is actually already an error between a and c because as soon as either value for b is chosen, one of the validations will fail.

While this naïve approach is sound, it is incomplete, and incomplete validation leads to unsatisfactory clients. For example, if there were 10 a's and 10 c's, but a single b upon which all the a's and c's depended, an unlucky user who happens to set b last might end up with every one of those fields highlighted with an error with no idea how to fix the errors or even how many errors there actually were (at most 10 independent errors). While it is tempting to describe such examples as "corner cases", the heavily-studied problem of configuration management is replete with such examples [6]; moreover, such examples become more frequent as the complexity of validation increases. And the more complex the validation the more a tool like CAVEAT is useful: when writing and maintaining the validation code is hard enough to do just on the server, replicating that code for the client while taking unknowns into account is something a developer may not even attempt.

Plato [7] is a tool designed to build validation routines for the client that are both sound and complete; thus, in the example above, as soon as a is set to True and c to False, Plato's client would signal an error and highlight a and c for the user. The challenge to applying Plato is that its input is a fragment of first-order logic. Thus instead of translating Ruby validations to JavaScript validations, the use of Plato reduces the problem to translating Ruby validations to first-order logic. To address this challenge, we identify a fragment of Ruby that is commonly used and can be translated into first-order logic.

Installing client-side validation. Once we have JavaScript implementations of the server-side validation routines, those functions must be installed on the appropriate web page. In Rails, a developer writes Views that describe how to construct web pages; thus, the JavaScript implementations must be installed into the appropriate Rails Views. To do so, we must be able to track how each View's form fields are handled by Rails controllers when data for those fields is submitted by a user. We must also be able to trace how each controller uses form field data to instantiate Ruby objects and whether the controller runs server-side validations on that data.

 ISBN: 978-1-61208-298-1

Instead of attempting to apply program analysis algorithms to extract the necessary information from the Rails application automatically, we introduce a number of simple Rails conventions that if followed by the developer make installing client-side validation relatively straightforward. We believe that the conventions we propose are ones already followed in many applications.

Server-crucial validations. The discussion so far has focused on transforming server-side validation routines into client-side validation routines. But some server-side validation is conceptually more difficult to move to the client than others. Validation that relies on information the server has but the client does not can be difficult or even ill-advised to move to the client. For example, a user registration page that asks for a login ID, a password, first name, last name, address, etc. may require that the login ID is unique in the backend database. Moving such a validation to the client could be accomplished either by moving the entire list of current user IDs to the client or creating client-validation code that asynchronously communicates with the server (e.g., through AJAX).

To address this challenge we simply recognize which server-side validations can be implemented entirely on the client and which cannot, and then we transform only those validations that can be implemented entirely on the client. In addition to technical convenience, our choice to move just these validations to the client is motivated by a desire to preserve the overall security of the web application. Since many Rails applications are open-source, translating server-side code to the client tells users no more about the application than they could have learned from simply looking at the server code. But if we were to create client-side validations that moved some of the server's information to the client or communicated with the server asynchronously, we could potentially lessen the overall security of the application.

B. Architecture

Conceptually CAVEAT can be broken down into the three components shown in Figure 3, which together address the challenges discussed above: Validation identifier, Validation translator, and Validation installer.

- **Validation identifier**: Takes as input a Rails application and identifies all the server-side validations that the developer has written.
- **Validation translator**: Takes all of the validators from the Validation identifier, eliminates those that cannot be implemented entirely on the client, and generates one JavaScript validation function for each form field f that embodies all the validation that must be performed on the client when f's value changes. When field f's function is executed, it returns a list of form fields that fail to validate because of f's new value. If the list is empty, f's new value caused no validation failures.
- **Validation installer**: Augments each web form View in the Rails application so the resulting web page invokes the JavaScript functions created by the Validation translator. More precisely, the installer attaches JavaScript functions to the form fields' event handlers so that each time a user edits a form field, the validation routines run, and when a validation fails, those form fields that are the cause of the failure are highlighted for the user (e.g., by coloring the offending fields red).

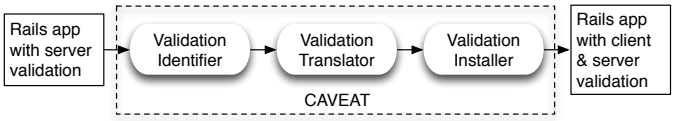


Figure 3: CAVEAT architecture

IV. Implementation

A. Validation Identifier

The first challenge in moving server side validation code to the client is identifying which server-side code is performing input validation. Ruby on Rails was chosen as a deployment target for CAVEAT because it has built-in constructs that make identifying server-side input validation straightforward. Each of Rail's MVC models includes a method called `_validate_callbacks` that returns all of the validation code for that model. The resulting list of validations includes built-in validators, custom validators, and Rails associations. CAVEAT is only interested in the custom validators, which can be identified based on the type hierarchy of Ruby. Thus identifying the validation code in a given Rails model amounts to a few lines of Ruby code.

B. Validation Translator

The Validation Translator takes as input a collection of Ruby validators and outputs JavaScript code that implements client-side validation for the given Ruby validators. The first step is converting each Ruby validator into an abstract syntax tree (AST), a task that CAVEAT carries out using the Ruby gem live_ast. Next CAVEAT converts each Ruby AST into a logical formula that Plato accepts as input or recognizes that the AST is not one that CAVEAT ought to move to the client. Finally, CAVEAT runs Plato once on the set of all logical formulae generated from the ASTs; the resulting JavaScript code implements the client-side data validation code.

The novel part of the Validation Translator is the second step: converting a validator (represented as an AST) into a logical formula or recognizing that the validator ought not be moved to the client. CAVEAT uses a whitebox approach to identifying which validators ought to be moved to the client. CAVEAT was designed to convert a specific fragment of Rails validators to the client; any validator that does not belong to this fragment is ignored. Figure II details the fragment of Ruby that CAVEAT moves to the client, which we call Ruby$_v$. Ruby$_v$ eliminates from full Ruby those language fragments that are perennially problematic for deep, semantic analysis of programming languages, e.g., loops, recursion, reflection, evaluation of dynamically-constructed code fragments, side effects, database operations, network connections.

While simple, we show in our evaluation that a reasonable percentage of custom validators in existing applications can

 ISBN: 978-1-61208-298-1

Table II: Ruby fragment Ruby$_v$

Construct	Example
Control logic	`if-then-else`, `switch`
Boolean Logic operators	`and`, `not`, `or`
Comparison operators	$<$, $>$, $>=$
Basic Arithmetic	$+$, $-$, $*$, $/$
Miscellaneous functions	string manipulation
Method Calls	calls to other custom validators

be expressed in Ruby$_v$. Furthermore, CAVEAT's ability to address validators written in a more expressive language than Ruby$_v$ is limited by Plato's input language, which is a heavily restricted fragment of first-order logic.

Converting a validator that is expressed in Ruby$_v$ is performed by recursively walking the validator's AST (and the ASTs of any Ruby function called from that validator). If the recursive walk determines that the validator has not been expressed in Ruby$_v$, it halts the translation. At the top level, the translator is a recursive method with a large switch statement that is conditioned on the root of the AST it is given. It translates each Ruby language construct (e.g., `if`) into a logical version of it (e.g., $\Rightarrow$). The translator also recurses into any other function that is called by a validator.

In addition, the translator recognizes those fragments of Rails code that signal an error. Sometimes an `if` statement is used to check for the presence of errors, while other times an `if` statement is used to check for the lack of errors. Recognizing error-signaling Rails code requires understanding how a model record is validated by Rails. Every model record has an array named `errors` that specifies what validations have failed during validation. By definition, a model record is valid if and only if its `errors` array is empty. Therefore, validation methods that want to block a record from being saved, must insert something into the errors array. For our purposes, we only care whether the validation has returned *true* or *false*. Thus, CAVEAT treats any statement that adds something to the `errors` array as a statement that signals an error. For the purpose of translation to logic, it suffices to replace any Ruby statement that adds to the `errors` array with *false*. Figure 4 gives pseudo-code for the main portion of the translator.

For example, the following Ruby validator would be represented by the AST shown in Figure 4. The translator would produce the logical formula $age < 16 \Rightarrow false$.

```
if age < 16 then
    errors[:age] << 'You are too young'
end
```

C. Validation Installer

After CAVEAT has generated JavaScript code implementing client-side validation, that JavaScript code must be installed on the client. Recall that each time the user changes a data element on a page, we want the CAVEAT-generated JavaScript to compute errors and for those errors to be communicated to the user via the client's graphical interface. Typical Rails

Algorithm 1 TRANSLATE(ast, parents)

Input: *ast* is an abstract syntax tree
Input: *parents* is a static version of the function call stack
Returns: First-order logic representation of AST. If the formula includes $\perp$, translation is not possible.

```
1: if ast[0] == :if then
2:     return TRANSLATE(ast[1]) ⇒ TRANSLATE(ast[2])
3: else if ast[0] == :and then
4:     return TRANSLATE(ast[1]) ∧ TRANSLATE(ast[2])
5: else if ast[0] == :or then
6:     return TRANSLATE(ast[1]) ∨ TRANSLATE(ast[2])
7: else if ast[0] == :not then
8:     return ¬ TRANSLATE(ast[1])
9: else if ast[0] == :call then
10:    if call represents a model variable then
11:        return ast[2]
12:    else if call adds to errors array then
13:        return false
14:    else if call represents arithmetic (e.g., in binary) then
15:        return TRANSLATE(ast[2]) ast[1] TRANSLATE(ast[3])
16:    else if call represents a method call then
17:        if ast[2].instance ∈ parents then
18:            // callee is recursive
19:            return ⊥
20:        else
21:            // callee is not recursive, so recurse and translate
22:            return TRANSLATE(get_ast(ast[2]))
23:    else
24:        return ⊥
25: else if ast[0] == :true then
26:    return true
27: else if ast[0] == :false then
28:    return false
29: else
30:    return ⊥
```

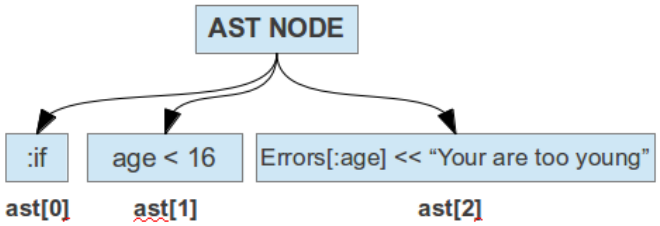


Figure 4: CAVEAT algorithms

clients already have the ability to display errors by highlighting offending fields in red, thus CAVEAT's main problem in terms of installing the JavaScript code is to ensure that it is invoked appropriately.

The JavaScript code generated by CAVEAT includes one function for each field of the MVC model that CAVEAT was invoked on. Thus each time the user modifies data entry element e, the client must run the CAVEAT code corresponding to the MVC model element e. We do this by leveraging jquery's bind() function to create a callback chain that runs the necessary CAVEAT functions for each field when it has changed value. In order to correctly bind these functions to the form fields we intend them for, we assume developers follow the best practice of labeling form fields using the exact name

 ISBN: 978-1-61208-298-1

of the attribute that the form corresponds to in the Ruby model.

V. EVALUATION

We evaluated CAVEAT by (i) writing new Rails applications using CAVEAT and (ii) analyzing the the ease with which CAVEAT can be applied to existing, open-source Rails applications.

A. New Rails Applications

We created three pages with complex validations from different domains: a new user-registration form (the running example), a vacation planning web form, and a checkout form for a shopping cart application.

New user form. Our running example is a typical form that solicits basic information about a new user: name, username, email, password, and birthday. The user must provide information for all fields. The username must be one that has not already been chosen. The name must be no longer than 80 characters. The birthdate must be a valid combination of month-day-year. The user must be at least 18 years old. The password and password confirmation must be the same. The email address must be properly formatted.

When writing the Rails server-side implementation of this form, we used built-in validators for ensuring that all fields have values and that the username is no longer than 80 characters. We used a database query within a validator to check that the provided username is unique. We wrote custom validators to ensure that (i) the birthdate is a proper combination of month-day-year, (ii) the new user is at least 18 years old, and (iii) the two passwords match.

The custom validators that we wrote for birthdate, age, and password were all expressed naturally in Ruby$_v$, and CAVEAT moved those over to the client properly. For example, each time the user enters an invalid month-day-year combination for the birthdate (e.g., February 30) the form highlights the offending fields in red (e.g., month and day). As expected, the remaining validators were ignored by CAVEAT.

Vacation planner form. This example models what a user might see on a travel agency's website when trying to plan a vacation. It has form fields for the budget, number of travelers, number of children, number of adults, vacation package options and departure and return dates. The number of travelers must be the sum of the number of children and the number of adults. The departure date must be before the return date. There are three vacation package options that vary how many nights the vacation lasts; the longer the vacation the higher the budget must be.

When implementing the form processing code in Rails, all of the validators were custom validators, and all of them were naturally written in Ruby$_v$. For example, the budget check was implemented as a `switch` statement on the number of nights to stay, and each case included an `if` checking if the provided budget was sufficient for the chosen number of nights. CAVEAT properly moved all of these validations to the client.

E-Commerce checkout form. This example models a shopping cart checkout form. Here we assume that a subtotal has already been set for an order and that the user has some store credit with the company. The example contains fields for a shipping option, donation, and a payment option (either None or credit card number and expiration date). The donation, payment, and subtotal fields must all have numeric values, and if the payment option is None then subtotal + shipping costs + donation - store credit is less than or equal to zero.

We used the Rails built-in validators to ensure all fields have values and that the donation, payment, and subtotal fields are numeric. To enforce the condition on the payment option, we wrote a custom validator. That custom validator was naturally expressed in Ruby$_v$, and CAVEAT moved it over to the client properly. As soon as all the fields are given values, the client highlights an error if the payment option None is selected but the outstanding cost is greater than zero.

Table III: Breakdown of custom validators without system errors

Custom Validation Method Status	Count	Percent
Written in Ruby$_v$	11	18 %
Could be written in Ruby$_v$	17	28 %
Failed due to loops	7	11 %
Failed due to database interaction	25	41 %
Failed due to networking calls	1	1 %

B. Existing Rails Applications

The second phase of our evaluation examined existing Rails applications. Our goal was to understand the prevalence of Ruby$_v$ validators in the wild. We gathered 25 open source applications from GitHub. We chose these applications based on the number of models that they contained, seeing this as a good indication on how many validations they would use. We also favored applications that were updated regularly and used a current version of Ruby and Ruby on Rails. Three notable applications in our sample were Redmine (a project management tool used regularly in industry), Diaspora (a social networking app) and Spree (a common e-commerce application). For each application, we ran a slightly modified version of CAVEAT that included additional logging capabilities on each of the 68 total custom validators across those 25 applications.

Of the 68 validators, 7 caused systems-level errors during our analysis. Most of the time these errors were due to conflicts with gem versions, specifically with the ast_live gem we use for generating the AST for the methods. Of the remaining validators, 11 validators were already expressed in Ruby$_v$, and 17 could easily have been written in Ruby$_v$. Thus overall, CAVEAT could be readily applied to 28 of 61 validators (46%). Those blocks not expressible in Ruby$_v$ included non-trivial and difficult to analyze loops, database interactions, and networking calls. See Table III for additional details. Although we are only able to support 46% of the analyzed legacy application validation code, new applications that can be written in Ruby$_v$ can be supported at 100%, as illustrated by the examples of the prior subsection.

VI. RELATED WORK

Two of the most germane works, Ripley [8] and [9], could seemingly be used to meet the same objective as CAVEAT:

 ISBN: 978-1-61208-298-1

write validation code once (on the client) and allow the system to automatically replicate it elsewhere (on the server). However, there is a crucial benefit to writing validation code on the server instead of the client: all constraints, whether static (not dependent on the server's database, file system, etc.) or dynamic (dependent on the server's state) can uniformly be written on the server, but only static constraints can easily be written on the client (implementing dynamic constraints requires AJAX and server-side support). Thus, even if all of a client's validation is moved to the server, the developer must write server-side validation.

The other relevant research work is WAVES [10], which automatically extracts server-side validation code from PHP and moves that validation code to the client. The benefit of CAVEAT compared to WAVES is that CAVEAT suffers from fewer analysis mistakes than WAVES. The first mistake WAVES can make is identifying which fragments of server-side code constitutes server-side validation—a mistake CAVEAT does not make because the MVC architecture of Rails facilities identification of validation routines. The second mistake WAVES can make is improperly moving a snippet of PHP validation code to the client—a mistake CAVEAT does not make because we have carefully identified the fragment of Ruby that CAVEAT moves to the client to ensure the resulting client-side code is correct.

Outside the research arena, the most sophisticated tools to aid web development are found within web development frameworks like Ruby on Rails (RoR) [11], Google Web Toolkit (GWT) [12], and Django [13]. Google Web Toolkit allows a programmer to specify which code is common to the client and the server. However, the programmer factors her code into the validation running on just the client, the validation running on just the server, and the validation running on both. Furthermore, GWT fails to provide the same robustness that CAVEAT does in the face of unknown values on the client as discussed in Section III.

We are only aware of the following two tools that allow a developer to write validation in one place and have it enforced in other places: (a) Ruby on Rails with the `client_side_validations` plugin [5], and (b) Prado [14]. With RoR, a developer writes the constraints that data should satisfy on the server, and `client_side_validations` enforces those constraints on the client. The limitation, however, is that the constraints extracted are limited to a handful of built-in validation routines and are implemented on the client using built-in validation of HTML5. Prado's collection of custom HTML input controls allows a developer to specify required validation at server-side, which is also replicated in the client using JavaScript. However, it also allows developers to specify custom validation code for server and client thus introducing avenues for inconsistencies in client and server validation. CAVEAT, in contrast, extracts constraints checked by the server and implements them on the client using custom-generated JavaScript code, thereby avoiding the possibility of inconsistency.

VII. Conclusion

CAVEAT obviates the need for developers to write and maintain two separate data validation code bases, a notoriously difficult problem in practice, and improves the overall security of the application by allowing developers to focus on the security-critical server-side data validation code. CAVEAT is implemented in Ruby on Rails, an especially attractive deployment target for CAVEAT since server-side validation is built into the framework itself, which makes the identification of server-side validation code much simpler than with other web programming languages and frameworks. CAVEAT was designed to operate on validations expressed in a fragment of the Ruby programming language that admits deep, semantic analysis. We evaluated CAVEAT's effectiveness by constructing three new applications using CAVEAT, and we evaluated its applicability to 25 existing applications by investigating the proportion of validation checks that CAVEAT could replicate on the client.

Acknowledgments

This research is support in-part by the following grants from the U.S. National Science Foundation: CNS-1141863, DUE-1241685, CNS-0845894, CNS-1065537, DGE-1069311.

References

[1] P. Bisht, T. Hinrichs, N. Skrupsky, R. Bobrowicz, and V. Venkatakrishnan, "NoTamper: Automatic Blackbox Detection of Parameter Tampering Opportunities in Web Applications," in *CCS'10: Proceedings of the 17th ACM Conference on Computer and Communications Security*, Chicago, IL, USA, 2010.

[2] P. Bisht, T. Hinrichs, N. Skrupsky, and V. Venkatakrishnan, "WAPTEC: Whitebox Analysis of Web Applications for Parameter Tampering Exploit Construction," in *CCS'11: Proceedings of the 18th ACM Conference on Computer and Communications Security*, Chicago, IL, USA, 2011.

[3] R. Wang, S. Chen, X. Wang, and S. Qadeer, "How to Shop for Free Online – Security Analysis of Cashier-as-a-Service Based Web Stores," in *Oakland'11: Proceedings of the 2011 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 2011.

[4] M. Alkhalaf, T. Bultan, S. R. Choudhary, M. Fazzini, A. Orso, and C. Kruegel, "ViewPoints: Differential String Analysis for Discovering Client and Server-Side Input Validation Inconsistencies," in *ISSTA'12: Proceedings of the 2011 International Symposium on Software Testing and Analysis*, Minneapolis, MN, USA, 2012.

[5] "Client-side validation Ruby gem," https://github.com/bcardarella/client_side_validations, last accessed: 01 Apr 2013.

[6] "CLib: Configuration benchmarks library," http://www.itu.dk/research/cla/externals/clib/, last accessed: 01 Apr 2013.

[7] T. L. Hinrichs, "Plato: A Compiler for Interactive Web Forms," in *PADL'11: Proceedings of the 13th International Conference on Practical Aspects of Declarative Languages*, Austin, TX, USA, 2011.

[8] K. Vikram, A. Prateek, and B. Livshits, "Ripley: Automatically Securing Distributed Web Applications Through Replicated Execution." in *CCS'09: Proceedings of the 16th Conference on Computer and Communications Security*, Chicago, IL, USA, 2009.

[9] D. Bethea, R. Cochran, and M. Reiter, "Server-side Verification of Client Behavior in Online Games," in *NDSS'10: Proceedings of the 17th Annual Network and Distributed System Security Symposium*, San Diego, CA, USA, 2010.

[10] N. Skrupsky, M. Monshizadeh, P. Bisht, T. L. Hinrichs, V. N. Venkatakrishnan, and L. Zuck, "Waves: Automatic synthesis of client-side validation code for web applications," *ASE Science Journal*, 2013.

[11] "Ruby on Rails," http://rubyonrails.org/Last accessed: 01 Apr 2013.

[12] "Google Web Toolkit," http://code.google.com/webtoolkit, last accessed: 01 Apr 2013.

[13] "django: Python Web Framework," https://www.djangoproject.com/ Last accessed: 01 Apr 2013.

[14] "Component Framework for PHP5," http://www.pradosoft.com, last accessed: 01 Apr 2013.

 ISBN: 978-1-61208-298-1

Need Only One Bit: Light-weight Packet Marking for Detecting Compromised Nodes in WSNs

Yuichi Sei Akihiko Ohsuga
Department of Social Intelligence and Informatics, Graduate School of Information Systems
The University of Electro-Communications
Tokyo, JAPAN
sei@is.uec.ac.jp ohsuga@uec.ac.jp

***Abstract*—In large-scale sensor networks, adversaries may capture and compromise several of the sensors. A compromised node can be used to create false messages by generating them on their own or by fabricating legitimate messages received from other nodes. Our goal is to locate the compromised nodes that create false messages and forward them to the sink. Existing works can only be used in situations where there is one source node and a routing path from it to the sink is static. This limitation is a big problem in wireless sensor networks because of node failures. They also must receive a lot of false messages before they can locate a compromised node. We propose light-weight packet marking for detecting compromised nodes. In our proposed method, each node appends its abbreviated ID and 1 bit code to messages and the sink detects a compromised node by a statistical method. Our method can be used in static and dynamic environments and can detect compromised nodes faster. Our mathematical analysis and the simulations we conducted prove the effectiveness of our method.**

***Keywords*—Wireless sensor networks; Security; Compromised node detection.**

I. INTRODUCTION

A core function of wireless sensor networks (WSNs) is to detect and report events. These networks are suitable for tasks like intruder detection [1], and deploy a large number of sensor nodes over a vast region. Sensor nodes detect events of interest and deliver messages to the *sink* over multihop wireless paths. However, an adversary may capture and compromise several of the sensors. They can obtain all information including the secret keys stored in the compromised nodes, and these nodes can then be used to create false messages i.e., generate false messages on their own and/or fabricate legitimate messages they have received from other nodes.

Although there are many works on detecting such false messages [2]–[6], they cannot detect compromised nodes. There are currently three ways of detecting compromised nodes: verifying the integrity of the code running on a node, monitoring conducted by the nodes themselves, and traceback from the sink. Verifying the integrity of the code mechanism requires a challenge-response protocol [7], [8]. This mechanism is usually used only after detecting a suspicious node using other mechanisms, and can check whether or not the suspicious node is compromised. In our proposal, the sink can detect a compromised node at a high probability, i.e., it can detect a suspicious node. Therefore, verifying the integrity of the code running on a node, and the use of our proposal can coexist. The monitoring done by nodes mechanisms is vulnerable to collusion attacks because the monitor nodes may be compromised as well (we discuss this in Section III). Works on traceback in WSNs also exist [9], [10]. However, they can only be used in situations where there is only one source node and a routing path from it to the sink is static. However, this situation is unrealistic in WSNs because of node failures [11]. Although AK-PPM [12] can be used in environments where the rouging paths are changeable, it cannot identify compromised nodes that fabricate messages.

Our goal is to detect the compromised nodes that create false messages and forward them to the sink. We use the packet marking method to detect the source nodes that generate false messages and the nodes that fabricated messages. In our method, each forwarding node appends its ID and only 1-bit code to the message. Of course, compromised nodes can generate a correct code with 50% probability. Even so, we can detect compromised nodes by using a statistical procedure when some false messages reach the sink. Moreover, to reduce communication traffic further, we propose an optional method of abbreviating node IDs. We propose and analyze our method in a mathematical way. The simulations we conducted prove the effectiveness of our method compared with existing works.

The rest of this paper is organized as follows. Section II presents the models of false messages and sensor networks. Section III discusses the related methods and their problems. Section IV presents the design of our algorithm. Section V analyzes security of LPM. Section VI presents the results of our simulations. Section VII discusses several design issues in our method. Section VIII concludes the paper.

II. SYSTEM MODELS

In this section, we define our assumed sensor network model and the model of false message attacks.

A. Sensor Network Model

We assume a sensor network composed of a large number of small sensor nodes. The nodes can detect an event of interest. Each of the detecting nodes reports the signal it senses to the sink. We also take into account a static sensor network where the sensor nodes do not move once deployed.

 ISBN: 978-1-61208-298-1

We assume that the sensor nodes are not equipped with tamper-resistant hardware, because they are normally inexpensively designed. Sensors are usually built with limited battery energy, memory, and communication capabilities. In our model, we assume that the destination of messages is the sink. Our target is to detect compromised nodes that create false messages and forward them to the sink. The sink is a data collection center with large computation and storage capabilities that protects itself using advanced security solutions.

B. Creating false message attacks

An attacker may compromise multiple sensor nodes in a network. Once a sensor node is compromised, all the secret keys, data, and codes stored on it are exposed to the attacker. The compromised node can be used to create false messages, i.e., generate false messages by itself and/or fabricate messages it has received from other nodes. Such bogus reports can cause the user to make bad decisions and can cause mission-critical applications to fail. They can also induce congestion, and waste a significant amount of network resources (e.g., the finite amount of energy in a battery powered network and the bandwidth) along the data delivery paths. Therefore, we want to detect and eliminate compromised nodes as quickly as possible.

To decide which messages without fabrication are false messages is out of scope of this research. We can use many existing works of detecting such false messages [2]–[6] although they cannot detect compromised nodes.

III. RELATED WORK

In this section, we describe related works on detecting compromised nodes and their problems.

A. Verifying the integrity of the code running on a node

Code attestation mechanisms have been proposed [7], [8], [13] to verify the integrity of the code running on a node. These mechanisms are usually used only after the detection of a suspicious node by using other mechanisms, and they can also check whether or not the suspicious node is a compromised node. This is because the verification process requires a large amount of communication traffic and computation cost. The authors of the attestation methods mentioned this and recommended using their proposal with other mechanisms that can detect a suspicious node.

B. Monitoring conducted by the nodes themselves

Mechanisms to overhear neighboring communications have also been proposed. Watchdog [14] focuses on message forwarding misbehavior. In the watchdog scheme, the sender of a message watches the behavior of the next hop node of that message. If the next hop node drops or fabricates the message, the sender announces it as a compromised node to the rest of the network. Other works [15], [16] have proposed a collaborative intruder identification scheme.

These mechanisms are based on monitoring by participating nodes. These mechanisms are vulnerable to collusion attacks, because the detector nodes may be also compromised [17]. We would need to use these kinds of mechanisms if we wanted to send and receive messages within only the sensor nodes without a sink. However, we take into account a situation where the destination of the messages from the nodes is the sink. Therefore, we can assign the task of detecting compromised nodes to the sink, not to the nodes.

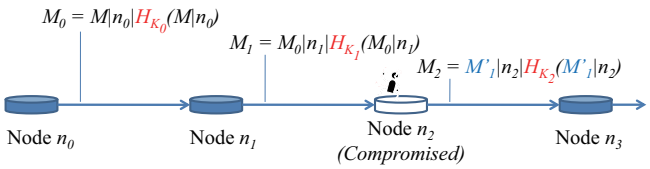


Figure 1. Algorithm of LPM (Compromised node n_2 fabricates the message from M_1 to M'_1)

C. Traceback from the sink

Many traceback mechanisms for the Internet have been proposed, such as [12], [18], [19]. They used a probabilistic packet marking algorithm in which each router appends its ID to packets with some probability. At the victim site, it can construct an attack graph i.e., the routing path of malicious packets. These mechanisms assume that the routers are reliable. Therefore, if forwarding router fabricates packets, the victim site cannot detect it. In WSNs, sensor nodes work as routers and they may be compromised. Therefore, we cannot use a probabilistic packet marking algorithm on the Internet without modification for WSNs.

PNM [9] modified probabilistic packet marking algorithm for WSNs. In PNM, each forwarding node appends its message authentication code (MAC) as well as its ID. Because each node appends its MAC, PNM can detect fabricated messages. In PNM, the sink constructs an attack graph from false messages in the same way as a probabilistic packet marking algorithm on Internet. However, the construction can be done only in the situations where the source node of messages is only one node and the routing path is static. Moreover, they also must receive a lot of false messages before they can construct an attack graph and locate a compromised node.

The mechanism in [10] can detect the source node that generated the false messages from fewer false messages than PNM. However, it cannot detect the node that fabricated a message. It also cannot be used in environments where the routing paths are changeable.

AK-PPM scheme was proposed for packet traceback in mobile ad hoc networks [12]. This method can be used in environments where the rouging paths are changeable. Although AK-PPM can identify the source node that creates a message, it cannot identify compromised nodes that fabricate messages.

IV. METHOD

We propose light-weight packet marking algorithm to detect compromised nodes that create false messages. Then we propose an optional method to abbreviate node IDs that are appended to messages.

 ISBN: 978-1-61208-298-1

A. Light-weight Packet Marking

In our proposed method, *every* forwarding node appends its ID and 1-bit hash value to messages. Its basic scheme is shown in Fig. 1. A MAC is used in PNM. The authors did not mention the bit length of a MAC. We usually consider the length of a MAC to be 64 bits or more in WSNs [20]. We reduce the length of a MAC to only 1 bit.

We assume that each sensor node has a unique ID n_i and shares a unique secret key k_i with the sink. H represents a secure hash function, and it is shared among all the nodes and the sink. $H_{k_i}(m)$ means the 1-bit hash value of message m calculated by a shared hash function H and node n_i's secret key k_i. We express a stream concatenation as $|$. The initial message M may contain the event type detected at node n_0, the detected time, and the location among other things. After creating an initial message M, node n_0 calculates the 1 bit hash value of $M|n_0$ by using its key k_0 and creates the message $M_0 = M|n_0|H_{k_0}(M|n_0)$. The next node n_1 receives message M_0. Node n_1 calculates the 1 bit hash value of $M_0|n_1$ by using its key k_1 and creates message M_1.

When the sink receives the final message $M_{n_r} = M_{n_{r-1}}|n_r|H_{k_r}(M_{n_{r-1}}|n_r)$, it starts a verification process. The sink has a shared hash function H and all the secret keys shared by each node. First, the sink calculates the 1 bit hash value of $M_{n_{r-1}}|n_r$ by using key k_r. If this value is the same as that included in message M_{n_r}, the sink retrieves the node ID of the previous hop $r-1$ and verifies $H_{k_{r-1}}(M_{n_{r-2}}|n_{r-1})$. The sink continues this process until it finds an incorrect hash value or verifies all the hash values. The node with the last verified hash value is represented as an *LVN*. If we use 64-bits MAC proposed in [9], [10], a compromised node (the source node of this message or the forwarding node that fabricated this message) is located in the LVN within a one-hop neighbor node.

However, in our proposed method, the compromised node (and its one-hop neighbor node) should not become always a LVN because the compromised node can generate a correct code for another node with 50% probability. Consider the situation shown in Fig. 1. When node n_2 fabricates a message, the candidates of a LVN are all the nodes between the source node and the compromised node, i.e., nodes n_0, n_1, and n_2 in this example. However, we can use the fact that the probability that node n_2 becomes a LVN is highest among these nodes. Therefore, we can decide which node is most suspicious by counting the times each node becomes a LVN.

Note that we consider only the node with the *last* verified hash value (LVN) although several codes might be incorrect. Therefore, even if a compromised node changes the content of the message or the 1-bit code appended by other sensors, our proposed method can detect the compromised node.

The problem becomes more difficult when we think that the routing paths can change. For example, consider the situation shown in Fig. 2. In this example, node n_2 may become a LVN with the highest probability. Moreover, node n_2 may change the next node for forwarding messages to node n' rather than nodes n_3, n_4, and n_5. In this situation, detecting compromised nodes without making wrong decisions becomes even more difficult.

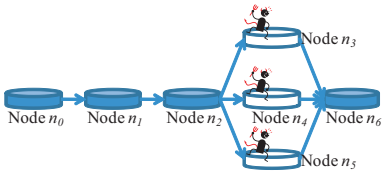


Figure 2. Collusion attacks in situations where routing paths can change

The compromised node can select whether or not it will append a *correct* hash value of a false message for its 1-bit code. In the following of the paper, we assume that compromised nodes always append the correct code of false messages to simplify the discussion. If this assumption is wrong, we may make the node next to a compromised node a compromised node. This limitation is common among existing works that use a traceback mechanism, such as [9], [10], [12].

1) Marking and Verification: We assume that each sensor node has a unique ID n_i and shares a unique secret key k_i with the sink. All forwarding nodes append their node IDs to the message and generate a 1 bit code by using their own secret keys. Then, the nodes append the 1 bit code to the message.

The procedure for detecting compromised node is as follows. Let us take a node n_u into consideration. We count the number of times node n_u became a LVN. We also count the number of times the nodes around node n_u became LVNs. Then, we calculate the probability of node n_u being a compromised node given these values. If the probability of node n_u being a compromised node exceeds a threshold (e.g. 0.999), we conclude that node n_u is a compromised node.

When the sink receives a false message, it records all the IDs included in the message, and calculates a LVN. Let the routing path of a false message be $p_i = \langle n_a, n_b, ... \rangle$ (here, $a, b...$ represents the node IDs). The number of hops from the source node to the sink is represented by $|p_i|$. A set of all the routing paths of the false messages the sink has received is represented by $P = \{p_1, ..., p_d\}$. The value d is the number of times the sink received false messages.

The node ID of a LVN in routing path p_i is represented by $L[p_i]$. The order of node n_a appearing in path p_i is represented by $M_a[p_i]$. The order of the LVN appearing in path p_i is represented by $M_L[p_i] = M_{L[p_i]}[p_i]$.

Every time the sink receives a false message, it starts a process of detecting compromised nodes. Let the last LVN be node n_u. We extract all the routing paths that include n_u and satisfy $M_L[p] > M_u[p]$ from P and let these paths be P_u. That is,

$$P_u = \{p_i \mid p_i \in P \,\&\, n_u \in p_i \,\&\, M_L[p_i] > M_u[p_i]\}. \quad (1)$$

Let the i-th path of P_u be $P_{u,i}$. We prepare the counters $B(u) = \langle b_1(u), ..., b_{N_u}(u) \rangle$ and $C(u) = \langle c_1(u), ..., c_{N_u}(u) \rangle$. Here, N_u is the maximum number -1 of hops from node n_u

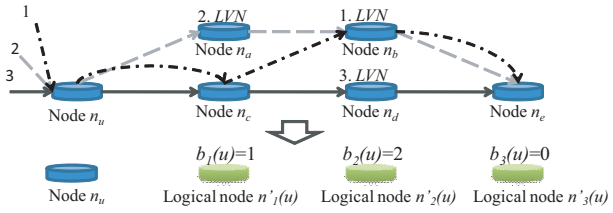


Figure 3. Logical nodes.

to the sink in P_u, that is,

$$N_u^{\max} = \max_i(|P_{u,i}| - M_u[P_{u,i}]) - 1. \quad (2)$$

Then, we calculate all $b_i(u)$ and $c_i(u)$ as follows. A $b_i(u)$ value represents the number of times that a node, which is situated nearer the sink in relation to node n_u and is i hops away from node n_u, became a LVN in P_u. Note that the node corresponding to $b_i(u)$ is changeable, because we assume that the routing paths can change (Fig. 3). In the figure, arrows represent routing paths of three false messages and text LVN represents that node became a LVN. We call the node corresponding to $b_i(u)$ a logical node $n'_i(u)$. That is, logical node $n'_i(u)$ represents a node situated nearer the sink in relation to node n_u and is i hops away from node n_u in each path of P_u. For example, in Fig. 3, logical node $n'_2(u)$ represents node n_b in paths 1 and 2, and node n_d in path 3. By introducing a logical node, we can deal with the change in routing paths and can reduce the amount of calculation at the sink. Each $b_i(u)$value is calculated by

$$b_i(u) = \sum_{j=1}^{|P_u|} \delta_{i,M_L[P_{u,j}]-M_u[P_{u,j}]}, \quad (3)$$

where $\delta_{i,j}$ is the Kronecker delta.

A $c_i(u)$ value represents the expected number of times that a logical node $n'_i(u)$ became a LVN *caused by the node itself* in P_u. That is, $c_i(u)$ is a value of $c_i(u)$ minus the number of times logical node $n'_i(u)$ became a LVN *caused by nodes other than the node*. For example, see Fig. 3. The $b_1(u)$ value is 1. The candidate nodes that created false messages are logical nodes $n'_1(u)$, $n'_2(u)$, and $n'_3(u)$. It is possible that logical node $n'_1(u)$ created false messages one or more times and the node became a LVN once. It is also possible that logical node $n'_2(u)$ and/or $n'_3(u)$ created false messages and logical node $n'_1(u)$ did not create any, and node $n'_1(u)$ became a LVN once. In the first case, the logical node n'_1 became a LVN caused by itself once. In the second case, logical node $n'_1(u)$ never became a LVN caused by itself. We can calculate the expected number of times that logical node $n'_1(u)$ became a LVN caused by itself, i.e., $c_i(u)$.

First, we initialize all $c_i(u)$ to $b_i(u)$. We prepare an integer j initialized to N_u^{max}. We update each $c_i(u)$ $(i = 1, ..., j-1)$:

$$c_i(u) \rightarrow \max(0, c_i(u) - c_j(u) \cdot 2^{\cdot(i-j)}), \quad (4)$$

where the function max gets two arguments and returns a greater one. We repeated this calculation from $j = N_u^{max}$ to $j = 1$.

Let the number of times node n_u became a LVN in P_u be L_u. The probability that nodes other than node n_u increased the number of times node n_u became LVNs by less than L_u is the same as the probability that node n_u increased it one or more times caused by itself, i.e., node n_u is a compromised node. The probability is represented by

$$\hat{P}(u) = \sum_{z=0}^{L_u-1} I(z), \quad (5)$$

where $I(z)$ represents the probability that logical nodes of n_u increased the number of times node n_u became a LVN by z. For example, $I(0)$ represents the probability that nodes other than node n_u have not increased the number of times node n_u became a LVN, i.e., only node n_u affected it. That is, node n_u created false messages more than z times, and because of this, it became a LVN z times. On the other hand, $I(L_u)$ represents the probability that only nodes other than node n_u affected the number of times node n_u became a LVN, i.e., node n_u is not a compromised node.

Equation 5 represents the probability that node n_u increased the number of times it became a LVN by at least one. This probability equals the probability that node n_u is a compromised node.

We calculate $I(z)$ in the following way. Let us take into consideration a node n_a, and that V_a (the number of times node n_a became a LVN caused by itself) is c and W_a (the number of times node n_a created false messages) is r. From Bayes' theorem, the conditional probability of node n_a creating a false message r times given the situation where it became a LVN c times caused by itself, $P_a(W_a = r|V_a = c)$ is represented by

$$P(W_a = r|V_a = c) = \frac{P(W_a = r) \cdot P(L_a = c|W_a = r)}{\sum_{i=0}^{\infty} P(W_a = i) \cdot P(L_a = c|W_a = i)}, \quad (6)$$

where $P(V_a = c|W_a = r)$ represents that the conditional probability of node n_a became a LVN c times caused by itself given the situation where node n_a created false messages r times.

Consider that node n_a creates a false message and the sink detects that the message is a false message. If the verification of the next node to node n_a fails, node n_a becomes a LVN. This probability is $1-1/2$. If the verification of the next node to node n_a succeeds, the node n_a does not become a LVN. This probability is $1/2$. Therefore,

$$P(V_a = c|W_a = r) = {}_rC_c(1-1/2)^c(1/2)^{r-c}. \quad (7)$$

$P(W_a = r)$ in (6) represents the probability that node n_a created false messages r times. Since the number of times that node n_a became a LVN caused by itself is c, the number of times node n_a create false messages r should be greater or equal c. Therefore, when $r < c$, $P(W_a = r) = 0$. When $r \geq c$, we can assume that every $P(W_a = r)$ has the same value, because a compromised node can create false messages arbitrary times. Therefore,

$$P(W_a = r|V_a = c) = \frac{{}_rC_c(1-1/2)^c(1/2)^{r-c}}{\sum_{i=c}^{\infty}[{}_iC_c(1-1/2)^c(1/2)^{i-c}]}$$
$$= (1/2)^{1-c-r}(1-1/2)^c(-1+2){}_rC_c. \quad (8)$$

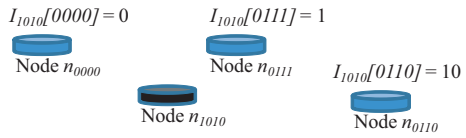


Figure 4. Node IDs and abbreviated IDs for node n_{1010} (IDs are represented by a binary numeral system).

The conditional probability of node n_a, which is h hops away from node n_u, having increased the number of times node n_u became a LVN by $D_a = q$ times, given the situation where V_a is c and node n_a created false messages r, is

$$P(D_a = q|W_a = r\&V_a = c) = \frac{P(W_a = r) \cdot P(D_a = q\&V_a = c|W_a = r)}{P(W_a = r) \cdot P(V_a = c|W_a = r)} \quad (9)$$

The conditional probability of node n_a, which is h hops away from node n_u, having increased the number of times node n_u became a LVN by $D_a = q$ times, given the situation where V_a is c, is

$$dp(h, q, c) = P(D_a = q|V_a = c) = \sum_{r=c+q}^{\infty} P(W_a = r|V_a = c) \cdot P(D_a = q|W_a = r\&V_a = c) = \frac{2^{-hq}(1 + 2^{-h})^{-1-c-q}(c + q)!}{c!\ q!}. \quad (10)$$

The probability that logical nodes $n'_1(u), ..., n'_{N_u(u)}$ of node n_u increased the number of times node n_u became a LVN by z is

$$I(z) = \sum_{q_1=0}^{z} \sum_{q_2=0}^{z-q_1} ... \sum_{q_{N_u-1}=0}^{z-\sum_{i=1}^{N_u-2} q_i} \left[dp(h, z - \sum_{i=1}^{N_u-1} q_i, c_N) \prod_{i=1}^{N_u-1} dp(h, q_i, c_i) \right] \quad (11)$$

From (5) and (11), when $\hat{P}(u) > th$ is satisfied, the probability that node n_u is a compromised node is greater than th. Therefore, we conclude that node n_u is a compromised node.

2) After deciding which node is a compromised node: If node n_u, in which we concluded that a suspected node is actually a compromised node, we can physically remove or isolate the node from the network. We do not mention in this paper how to do this after a compromised node is detected.

If node n_u, in which we concluded that a suspected node is not a compromised node, we reset the number of times node n_u became a LVN to 0 for the later process of detecting compromised nodes.

B. Abbreviation of Node IDs

Because each node appends its ID and a 1 bit code to messages in LPM, the node IDs can be a bottleneck. Therefore, we propose an optional method to abbreviate node IDs appended to messages. We assume that sensor nodes do not move after deployment.

1) Creation of Abbreviated IDs: The node IDs are represented by a binary numeral system here. After deployment, each node broadcasts its ID to one-hop neighbor nodes. Consider that node n_u receives IDs of its neighbor nodes from them. Then node n_u creates an abbreviated ID $n'_u[i]$ for each neighbor node n_i and teaches it to node n_i. That is, each node n_i learns its abbreviated ID $n'_j[i]$ from each neighbor node n_j and they can be all different.

Node n_u should identify node n_i from its abbreviated ID $n'_u[i]$. The pseudo-code of creating abbreviated IDs is described in Algorithm 1. When a node receives IDs of its all neighbor nodes, it starts this algorithm. An abbreviated ID $n'_u[i]$ is created by extracting the minimal part of bits, of an original ID i, that is different from all other abbreviated IDs $n'_u[j]$.

We use Fig. 4 for an example. Focus on node n_{1010} and consider that only other three nodes are neighbor nodes of node n_{1010}. For node n_{1010}, node n_{0111} can be identified by only ID 1 because the first bit of the ID of each other two nodes (n_{0000} and n_{0110}) is 0. In the same way, for node n_{1010}, node n_{0000} and node n_{0110} can be identified by ID 00 and ID 10, respectively. Then we delete prefix all '0's of the abbreviated ID except for the first bit. Therefore, ID 00 is further abbreviated to 0.

Let b denote the bit length of an original node ID. In the Algorithm 1, the function getBit returns the i-th bit of the ID and the function getBits returns from the i-th bit to the first bit of the ID. For example, when an ID is 01011, the function getBit(1) returns 1 and the function getBit(3) returns 011. The function deletePrefixZeros deletes the prefix all '0's of the ID except for the first bit. For example, when the function receives an input 000, it returns 0. If it receives an input 0100, it returns 100. Each node n_j teaches to its each neighbor node n_k a corresponding abbreviated ID $abbrIDs$.get(k).

Algorithm 1 createAbbreviatedIDs

Input: IDs = Set of IDs of neighbor nodes
Output: $abbrIDs$ = Table of ID and its abbreviated ID

```
while IDs is not empty do
    ID ← one of IDs;
    IDs ← IDs \ {ID};
    for i = 1 to b do
        isMatch ← false;
        for all eachID ∈ IDs do
            isEachMatch ← true;
            for j = 1 to i do
                if ID.getBit(j) != eachID.getBit(j) then
                    isEachMatch ← false;
                end if
            end for
            if isEachMatch is true then
                isMatch ← true;
            end if
        end for
        if isMatch is false then
            abbrIDs.put(ID, deletePrefixZeros(ID.getBits(i)));
        end if
    end for
end while
```

2) Using Abbreviated IDs: In LPM, the sink should identify node n_u from its abbreviated ID $n_x[u]$. The value of x is each ID of neighbor nodes of node n_u.

When node n_i transfers a message to node n_j for the first time, node n_i appends to the message its original ID i. After that, when node n_i transfers a message to node n_j, its appends to the message $n'_j[i]$ as its ID.

When the sink receives a message, it records all original node IDs on the path, specifically, it creates/updates list L_i of previous node IDs of each node n_i. For example, consider that node n_{j_1} transferred a message to node n_i and node n_{j_2} transferred another message to node n_i, and both messages were sent to the sink. In this case, L_i contains j_1 and j_2. If the sink receives another message and it knows from the message that node $n'_i[x]$ transferred it to node n_i, the sink seeks the corresponding original ID of the abbreviated ID $n'_i[x]$ from list L_i. The corresponding original ID is surely in list L_i. Let the bit length of $n'_i[x]$ is m. The sink get IDs whose the lower m bits are coincident with $n'_i[x]$. If the sink get only one ID, the ID is the corresponding original ID of $n'_i[x]$. Otherwise, the sink add '0' to the $m+1$-th bit of $n'_i[x]$ (i.e., the resultant bits is $0|n'_i[x]$, here $|$ represents a concatenation of bits), then gets IDs whose the lower $m+1$ bits are coincident with $0|n'_i[x]$. Until the sink gets only one ID, it increments m and repeats this process.

3) Analysis: We require the expected bit length of an abbreviated ID. Consider that there are three nodes n_u, n_1, and n_2, and node n_u is creating abbreviated IDs of nodes n_1 and n_2. If the first bits of IDs of nodes n_1 and n_2 are different, the bit length of their abbreviated IDs is 1 (i.e., 0 and 1). If not, the bit length is 2 if the second bits are different. In specifically, with probability $1/2^x$, the bit length of an abbreviated ID is x. Let X denote a random value of the bit length of an abbreviated ID in situations where there are two nodes determining their abbreviated IDs. Therefore, the probability distribution of the bit length of an abbreviated ID is represented by

$$f(x) = P(X = x) = \begin{cases} 2^{-x} & \text{if } x = 1, 2, ..., b-1 \\ 2^{1-b} & \text{otherwise (i.e., } x = b) \end{cases} \quad (12)$$

We approximate (12) by $f(x) = 2^{-x}$ for all $x = 1, 2, ..., b$ and we require an upper bound of the expected bit length of an abbreviated ID. Therefore, the *cumulative distribution function* of $f(x)$ is represented by

$$F(x) = P(X \leq x) = \sum_{u=-\infty}^{x} f(u) = 1 - 2^{-x}. \quad (13)$$

Let d denote an average number of neighbor nodes. Consider node n_a, which is one of neighbor nodes of node n_u. Let $n_i\,(i = 1, ..., a-1, a+1, ...d)$ denote another neighbor node of node n_u. Let $X_i\,(i = 1, ..., a-1, a+1, ...d)$ denote a random value of the bit length of abbreviated IDs of two nodes n_a and n_i when we consider only these two nodes. Every X_i is independent from each other. Therefore, the cumulative distribution function of the bit length of the abbreviated ID of node n_a when we consider all neighbor nodes of node n_u is represented by

$$F_A(x) = P(X_1 \leq x, ..., X_{a-1} \leq x, X_{a+1} \leq x, ..., X_d \leq x)$$
$$= [P(X \leq x)]^{d-1} = (1 - 2^{-x})^{d-1} \quad (14)$$

The probability distribution $f_A(x)$ of the bit length of node n_a when we consider all neighbor nodes of node n_u is required by differentiating the cumulative distribution function $F_A(x)$. Hence,

$$f_A(x) = \frac{dF_A(x)}{dx} = 2^{-x}(1 - 2^{-x})^{d-2}(d-1)\log 2 \quad (15)$$

Therefore, the expected bit length of an abbreviated ID is

$$E_L = \sum_{i=1}^{b} i \cdot f_A(i) \quad (16)$$

Next, we consider the effect of deleting prefix 0s. With probability $1/2$, an ID has no prefix 0s. With probability $1/2^{x+1}$, an ID has just x prefix 0s. Therefore, the expected number of prefix 0s is

$$E_0(b') = \sum_{x=1}^{b'} x(2^{-x-1}) = 1 - 2^{-1-b'}(2 + b') \quad (17)$$

Hence, the final bit length L_a is

$$L_a = E_L - E_0(E_L) \quad (18)$$

V. ANALYSIS

We analyze the security strength of LPM. LPM can detect compromised nodes within one-hop neighborhood area asymptotically as the sink receives sufficient number of false messages over time.

First, we prove Theorem 1 the unavoidableness of becoming a LVN for a compromised node. Next, we prove

- Theorem 2: $\hat{P}(u) \leq 0.5$ for legitimate nodes, and
- Theorem 3: $\hat{P}(u) \to 1$ for compromised nodes that create false messages infinitely,

when we assume that the probability of which node becomes a LVN follows exactly the probability distribution. That is, a compromised node shoule become a LVN one or more times if it created false messages several times from Theorem 1, and we define that a compromised node is a node that became a LVN one or more times to prove Theorems 2 and 3.

Theorem 1. *Node n_u or its one-hop neighbor node n_v becomes a LVN with some probability (> 0) if node n_u creates a false message.*

Proof. Consider that node n_u is a source node of a false message, and it appends a legitimate code to the message. In this case, the sink succeeds to verify all the codes. Therefore, node n_u becomes a LVN. Next, consider that node n_u is a source node of a false message, and it appends a false code to the message. In this case, the sink succeeds to verify codes until node n_v, which is one-hop next node of node n_u (i.e.,

node n_v receives the message from node n_u) and fails to verify code of node n_u. Therefore, node n_v, which is one-hop neighbor of node n_u becomes a LVN.

Consider that node n_u fabricates a message, and it appends a legitimate code to the message. In this case, the codes of from the source node to one-hop previous node n_t of node n_u (i.e., node n_u receives the message from node n_t) can be false. The probability that the code of node n_t is false is $1/2$. Therefore, node n_u becomes a LVN with probability $1/2$. Next, consider that node n_u fabricates a message, and it appends a false code to the message. In this case, the sink succeeds to verify codes until node n_v, which is one-hop next node of node n_u and fails to verify code of node n_u. Therefore, node n_v, which is one-hop neighbor of node n_u becomes a LVN. □

Theorem 2. $\hat{P}(u) \leq 0.5$ *for legitimate nodes.*

Proof. Here, we use Lemmas 1, 2 described after and we denote just node n_i in place of logical node $n_i'(u)$. Consider node n_u and a set of paths P_u, in which each path contains node n_u and a LVN that is situated nearer the sink in relation to node n_u.

Case 1. Consider the situations where there is no compromised nodes in paths P_u. In this case, node n_u never becomes a LVN. Therefore, $\hat{P}(u) = 0$.

Case 2. Consider the situations where there is only one compromised node n_v in paths P_u. Let node n_u become a LVN c times and node n_v is h-hop away from node n_u. We consider $O_1 = \hat{P}(u)$ in this case. We prove that $O_1 \leq 0.5$ when h, and $c \geq 1$.

The expected number of times that compromised node n_v becomes a LVN caused by itself is $c_v(u) = 2^{kh}c$ when node node n_u becomes a LVN c times. The expected value of $c_i(u)$ is 0 for each node (which is legitimate) in paths P_u other than node n_v from Lemma 1.

First, consider the situations where there are only two nodes a compromised node n_u and a legitimate node in paths P_u. Let $I_2(z)$ denote the value of (11) in this case.

$$I_2(z) = \sum_{r=0}^{z} dp(h, 2^h c, r) dp(h_2, 0, z-r). \tag{19}$$

$I_2(z)$ increases when h_2 increases. And,

$$\lim_{h_2 \to \infty} dp(h_2, 0, r) \to \begin{cases} 1 & \text{if } r == 0, \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

Therefore, $I_2(z)$ has a maximum value $dp(h, 2^h c, z)$ when h_2 is an infinite value. Therefore, $dp(h, 2^h c, z) \geq I_2(z)$. We can prove $dp(h, 2^h c, z) \geq I(z)$ when there are one compromised node n_v and arbitrary number of legitimate nodes in paths P_u in the same way. Therefore,

$$O_1 = \sum_{z=0}^{c-1} I(z) \leq \sum_{z=0}^{c-1} dp(h, 2^h c, z). \tag{21}$$

Here,

$$dp(h, 2^h c, z) = \frac{(1+2^{-h})^{-1-2^h c - z}(2^h c + z)!}{2^{hz}(2^h c)! z!}. \tag{22}$$

$dp(h, 2^h c, z)$ increases when h increases because $z > 0 \,\&\, c > z$. Therefore,

$$dp(h, 2^h c, z) \leq \lim_{h \to \infty} dp(h, 2^h c, z) \to \frac{e^{-c} c^z}{z!}. \tag{23}$$

Therefore,

$$O_1 \leq \sum_{z=0}^{c-1} \frac{e^{-c} c^z}{z!} = \frac{\Gamma(c, c)}{\Gamma(c)}. \tag{24}$$

Here, $\Gamma(x)$ is the Gamma function and $\Gamma(a, x)$ is the upper incomplete gamma function. $\frac{\Gamma(c,c)}{\Gamma(c)}$ increases when c increases. And,

$$\lim_{c \to \infty} \frac{\Gamma(c, c)}{\Gamma(c)} \to 0.5. \tag{25}$$

Therefore, we get $O_1 \leq 0.5$.

Case 3. Consider the situations where there more than one compromised nodes in paths P_u. First, consider that there are two compromised nodes n_{v_1} and n_{v_2} and one legitimate node in paths P_u. Let compromised nodes n_{v_1} and n_{v_2} be in h-hop and h'-hop away from node n_u, respectively, and let $h' < h$. We consider $O_2 = \hat{P}(u)$ in this case. We prove that $O_2 \leq O_1$ when h, and c are the same as that of O_1.

Consider that node n_v becomes a LVN r times caused by node n_{v_1} and $c-r$ times caused by node n_{v_2}. Therefore, node n_u becomes a LVN c times in total. In this case, the expected number of times node n_{v_1} becomes a LVN caused by itself is $2^h r$ and the expected number of times node n_{v_2} becomes a LVN caused by itslef is $2^h(c-r)$.

First, consider the situations where there are two compromised nodes n_{v_1} and n_{v_2}, and one legitimate node n_t. Let node n_t be in h_2-hop away from node n_u. Let $I_3(z)$ of node n_u denote the value of (11).

$$I_3(z) = \sum_{i=0}^{z} \sum_{j=0}^{z-i} dp(h, 2^h r, i) dp(h', 2^{h'}(c-r), j) dp(h_2, 0, z-i-j). \tag{26}$$

$I_3(z)$ increases when h_2 increases. When we consider that $h_2 \to \infty$,

$$\begin{aligned} I_3(z) &\leq \sum_{i=0}^{z} dp(h, 2^h r, i) dp(h', 2^{h'}(c-r), z-i) \\ &\leq \sum_{i=0}^{z} dp(h, 2^h r, i) dp(h, 2^h (c-r), z-i) \end{aligned} \tag{27}$$

Let $I_3'(z)$ denote the last expression. As mentioned above, both of $O_1 = \sum_{z=0}^{c-1} I_2(z)$ and $O_2 = \sum_{z=0}^{c-1} I_3(z)$ increase when h_2 increases. We get already $O_1 \leq 0.5$. Therefore, if we can prove

 ISBN: 978-1-61208-298-1

$\lim_{h_2\to\infty}(O_1 - O_2) \geq 0$, we get $O_2 \leq 0.5$.

$$\lim_{h_2\to\infty}(O_1 - O_2) \geq \sum_{z=0}^{c-1}(dp(h, 2^h c, z) - I_3'(z))$$
$$= \sum_{z=0}^{c-1} \frac{2^{h(1-z)}(1+2^{-h})^{-2^h c - z}(1+2^h c - 2^h z)(2^h c + z)!}{(1+2^h)^2 z!(1+2^h c)!}. \quad (28)$$

Because $c > z$, $O_1 - O_2 \geq 0$. We can prove $\hat{P}(u) \leq O_1$ when there are arbitrary number of compromised nodes and legitimate nodes in paths P_u, in the same way. Therefore, we get $\hat{P}(u) \leq 0.5$. □

Lemma 1. *$c_i(u) = 0$ for legitimate logical nodes $n_i'(u)$ and $c_i(u) > 0$ for compromised logical nodes $n_i'(u)$.*

Proof. Consider two logical nodes $n_i'(u)$ and $n_j'(u)$ and let $i < j$. Let logical node $n_j'(u)$ is a compromised node. Consider that logical node $n_i'(u)$ becomes a LVN c times caused by logical node $n_j'(u)$. In this case, the expected number of times logical node $n_j'(u)$ becomes a LVN is $2^{(j-i)}c$. Let the number of times logical node $n_i'(u)$ becomes a LVN caused by itself be c'. If logical node $n_i'(u)$ is a legitimate node, $c' = 0$. Otherwise, $c' > 0$. From (4), $c_i(u) = c'$. We can prove this when there are arbitrary number of compromised nodes, in the same way. □

Theorem 3. *$\hat{P}(u) \to 1$ for compromised nodes that create false messages infinitely.*

Proof. When a compromised node n_u creates false messages infinitely, it becomes a LVN infinitely. In this case, $\hat{P}(u)$ is,

$$\lim_{c\to\infty}\sum_{z=0}^{c-1} I(z) \to 1. \square \quad (29)$$

VI. EVALUATION

We conducted simulations to verify our analysis. We developed our own simulation platform, mainly because other simulators scale poorly for large numbers of nodes. Our simulator was implemented with basic geographic forwarding [21]. We set the length of the bits of the node ID to 10 by default and the bit length of a MAC to 64 just as in related work for PNM.

In the first simulation, we set the number of forwarding nodes on a path to 20 and the routing path was fixed. The first node is the source node of a message and the 10th node was set as a compromised node. The last node forwarded the messages it received to the sink. The 10th node always fabricated the messages it received. The source node repeatedly generated a message until the sink decided which node was the compromised node. This process was repeated 10,000 times.

Figure 5 presents the results. We changed the th from 0.1 to 0.99. The number of false messages the sink needed to conclude which node was a compromised node is shown in the figure. We know that we can detect compromised nodes earlier

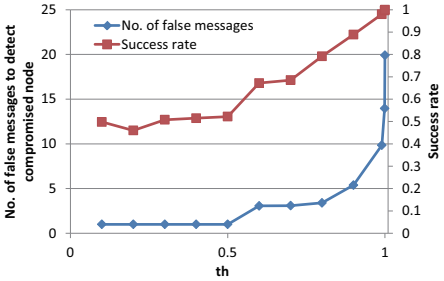


Figure 5. No. of false messages and success rate for detecting a compromised node

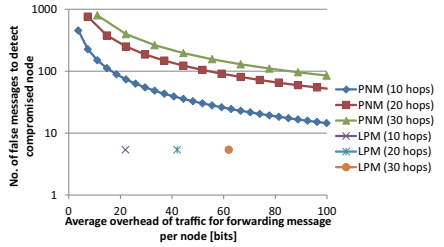


Figure 6. No. of false messages the sink received until it detected compromised node depending on the no. of hops from the compromised node to the sink

if we set th to a smaller one. The success rate in which the sink correctly detected a compromised node is shown. The success rate is the number of compromised nodes (here, the number is one), divided by the number of times that the sink decided a node was a compromised node until the sink detected all the compromised nodes (here, one compromised node). Figure 5 also shows the average of the success rates. We can say that the success rate in these simple simulations (i.e., the routing path is fixed) is almost the same as the threshold th. When the sink failed to detect a compromised node the first time, it followed the process outlined in Subsection IV-A2 and could finally detect a compromised node.

Then, we compared our method with that from a related work on PNM. The results are shown in Fig. 6. We set the average number of neighbor nodes to 8. The x-axis represents the average overhead of communication traffic for forwarding a message per node. This was calculated by dividing the total communication traffic from the source node to the sink by the number of hops. We set the success rate th to 0.9. The results in Fig. 6 helped us to determine that our method could detect a compromised node from fewer false messages than that of PNM. For example, we can detect a compromised node with only from 4% to 8% of the false messages as compared with PNM when the number of hops from the source node to the sink set to from 10 to 30.

To confirm the effectiveness of using abbreviated IDs, we conducted a mathematical analysis. The bit length of an original node ID was set to 10. We change the average number of neighbor nodes from 3 to 30. We assumed that the number of neighbor nodes followed a Poisson distribution. The results

 ISBN: 978-1-61208-298-1

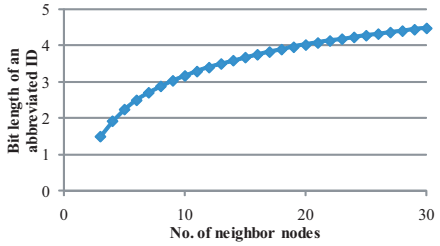


Figure 7. Bit length of an abbreviated node ID (An original length is 10).

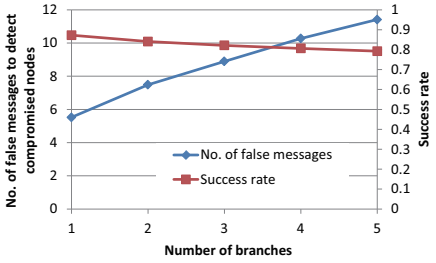
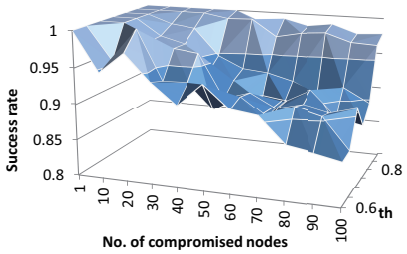
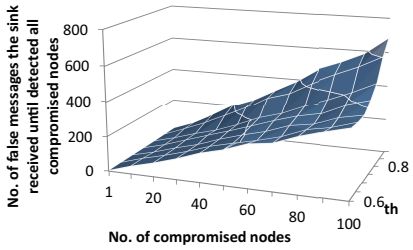


Figure 8. Resilience to collusion attacks



(a) Success rate for detecting compromised nodes



(b) No. of false messages sink received until it detected compromised node

Figure 9. The number of false messages and success rate

are shown in Fig. 7. We know from the figure that we could reduce the bit length of a node ID by from 55% to 85% by using our abbreviated IDs method.

Next, we conducted an experiment to know whether our method is resilient to collusion attacks. We set the nodes and routing paths following the information listed in Fig. 2. The legitimate node n_2 randomly forwarded a message to one of the next nodes and the node that received a message from node n_2 always fabricated it. The results are shown in Fig. 8. The x-axis represents the number of branches. For example, it was three in Fig. 2. We set th to 0.9. If the number of branches was large, the success rate for correctly detecting a compromised node decreased. However, the reduction rate was small; from 0.9 to 0.79.

Finally, we conducted an experiment to understand whether our method is resilient to changes in routing paths. The number of sensor nodes was set to 1,000. One of them repeatedly generated a message. We set the number of compromised nodes from 1 to 100. When a compromised node received a message, the node fabricated the message with random probability. Even if a compromised node received a message that had been already fabricated, it fabricated it further with random probability. Every time the sink received a false message, we randomly changed the locations of *all* nodes. The neighboring nodes of each node also changed based on the locations and then the routing paths of a message changed. Figure 9 shows the results. We set the threshold th from 0.6 to 0.9.

From Fig. 9(a), we know that all the success rates were higher than that for each th we set. We know that our method works effectively in the situation where the routing paths can change for the following reasons. For reference look at Fig. 1. If the routing paths cannot change, node n_1 always becomes a LVN with some probability when a message routes node n_1 and compromised node n_2. However, if the routing paths change, the neighbor node of compromised node n_2 can also change. Therefore, the probability that the sink decided a legitimate node was a compromised node decreases.

As a result from the information shown in Fig. 9(b), we know that the number of false messages needed until the sink detected all the compromised nodes increases, but we think that the rate is relatively scalable. When th was set to 0.9 and the number of compromised nodes was 100, the average number of false messages needed was about 7 per detection of one compromised node. This value is still less than that of PNM (the routing path in PNM is fixed and there is only one compromised node).

VII. DISCUSSION

In this section, we discuss several design issues for our method.

Cost overhead. Many works in WSNs set the default packet size to about 40 bytes [22], [23]. When the average number of neighbor nodes is 5 and the average number of hops from the source node to the sink is 20, the average overhead is $(\sum_i^{20}(1+3)\cdot i)/20 = 42$ bits = 6 bytes. Therefore, the overhead rate is 15%.

This value is much less than existing works for packet traceback as shown in Section VI. Moreover, we may reduce the average overhead by combining methods for detecting false messages described in II-B. Although existing works of detecting false messages [2]–[4], [24], [25] cannot identify the nodes that create false messages, they can notify the sink of the

 ISBN: 978-1-61208-298-1

existence of false messages. Only when the sink recognizes the necessity to identify the compromised node that create false messages, it floods a message to the network to start using the LPM protocol. When the sink identifies and removes the compromised node, it floods a message to stop using the LPM protocol.

Different ID attack. A compromised node can append a different ID to a false message. However, if nodes do not move after deployment, we can trace back to the one-hop neighbor of a compromised node (discussed also in [9]). We can improve our method by using neighbor authentication methods such as those in [26]. Moreover, we can use many studies of detecting faked IDs in wireless sensor networks [27] [28] [29].

Amount of calculation at the sink. We assume that the sink has large computation. However, the calculation of (11) may be a hard task for the sink. In the equation, we use N_u. The value of N_u can be large e.g., 30 and more. For practical purposes, we can set N_u to a smaller value e.g., from 5 to 10. This is because the probability, with which a node far from node n_u affects the number of times node n_u became an LVN, is very small. For example, consider a node that is 10 hops away from node n_u and creates a false message. The probability that node n_u becomes an LVN is only $1/2^{10}$.

VIII. CONCLUSION AND FUTURE WORK

Compromised nodes present severe security threats in sensor networks. We proposed a method to detect a compromised node that created a false message and reported it to the sink. Current solutions either are vulnerable to collusion attacks or cannot use the situation where the routing paths from a source node to the sink can change. We introduce a 1-bit code algorithm and a logical node to deal with changes in routing paths. Our method can be used if the routing paths are can change or not. We know that we can detect a compromised node from fewer false messages as compared with related works from our simulations. For our future work, we need to implement our method to real sensor nodes and examine their performance.

REFERENCES

[1] Z. Zhao et al., "QoE-aware FEC mechanism for intrusion detection in multi-tier Wireless Multimedia Sensor Networks," in Proc. IEEE WiMob. IEEE, Oct. 2012, pp. 689–696.

[2] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical en-route filtering of injected false data in sensor networks," IEEE Journal on Selected Areas in Communications, vol. 23, no. 4, April 2005, pp. 839–850.

[3] Z. Yu and Y. Guan, "A dynamic en-route scheme for filtering false data injection in wireless sensor networks." in Proc. IEEE INFOCOM, 2006, pp. 1–12.

[4] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks," in IEEE S&P, 2004, pp. 259–271.

[5] R. Lu, S. Member, X. Lin, and H. Zhu, "BECAN : A Bandwidth-Efficient Cooperative Authentication Scheme for Filtering Injected False Data in Wireless Sensor Networks," IEEE Trans. Parallel Distrib. Syst., vol. 23, no. 1, 2012, pp. 32–43.

[6] Z. Cao, H. Deng, Z. Guan, and Z. Chen, "Information-theoretic modeling of false data filtering schemes in wireless sensor networks," ACM Trans. Sensor Networks, vol. 8, no. 2, Mar. 2012, pp. 1–19.

[7] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla, "SWATT: Software-based attestation for embedded devices," in Proc. IEEE S&P, 2004, pp. 272–282.

[8] Y. Yang, X. Wang, S. Zhu, and G. Cao, "Distributed software-based attestation for node compromise detection in sensor networks," in Proc. IEEE SRDS, 2007, pp. 219–230.

[9] F. Ye, H. Yang, and Z. Liu, "Catching "moles" in sensor networks," in Proc. IEEE ICDCS, 2007, p. 69.

[10] Q. Zhang, X. Zhou, F. Yang, and X. Li, "Contact-based traceback in wireless sensor networks," in Proc. IEEE WiCom, 2007, pp. 2487–2490.

[11] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," IEEE Computer, vol. 35, no. 10, 2002, pp. 54–62.

[12] Z. Xu, H. Hsu, X. Chen, S. Zhu, and A. Hurson, "AK-PPM: An Authenticated Packet Attribution Scheme for Mobile Ad Hoc Networks," in Proc. International conference on Research in Attacks, Intrusions, and Defenses, 2012, pp. 147–168.

[13] X. Jin, P. Putthapipat, D. Pan, N. Pissinou, and S. K. Makki, "Unpredictable Software-based Attestation Solution for node compromise detection in mobile WSN," in 2010 IEEE Globecom Workshops, Dec. 2010, pp. 2059–2064.

[14] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in ACM MOBICOM, 2000, pp. 255–265.

[15] G. Wang, W. Zhang, G. Cao, and T. Porta, "On supporting distributed collaboration in sensor networks," in Proc. IEEE MILCOM, 2003.

[16] G. Dini and A. Lo Duca, "Towards a reputation-based routing protocol to contrast blackholes in a delay tolerant network," Ad Hoc Networks, vol. 10, no. 7, Sep. 2012, pp. 1167–1178.

[17] Y. Zhang, J. Yang, L. Jin, and W. Li, "Locating compromised sensor nodes through incremental hashing authentication," in Proc. DCOSS, 2006, pp. 321–337.

[18] Q. Dong, S. Banerjee, M. Adler, and K. Hirata, "Efficient probabilistic packet marking," in Proc. IEEE ICNP, 2005, pp. 368–377.

[19] D. X. Song and A. Perrig, "Advanced and authenticated marking schemes for IP traceback," in IEEE INFOCOM, 2001, pp. 878–886.

[20] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A pairwise key predistribution scheme for wireless sensor networks," in Proc. ACM CCS, 2003, pp. 42–51.

[21] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in Proc. ACM MOBICOM, 2000, pp. 243–254.

[22] M. T. Hansen, "Asynchronous group key distribution on top of the cc2420 security mechanisms for sensor networks," in Proc. ACM WiSec. ACM Press, Mar. 2009, pp. 13–20.

[23] C. Gezer, M. Niccolini, and C. Buratti, "An IEEE 802.15.4/ZigBee based wireless sensor network for Energy Efficient Buildings," in Proc. IEEE WiMob. IEEE, Oct. 2010, pp. 486–491.

[24] H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbaugh, "Toward resilient security in wireless sensor networks," in Proc. ACM MOBIHOC, 2005, pp. 34–45.

[25] F. Li and J. Wu, "A probabilistic voting-based filtering scheme in wireless sensor networks," in ACM MOBICOM, 2006, pp. 27–32.

[26] W. Gu, X. Bai, S. Chellappan, D. Xuan, and W. Jia, "Network decoupling: A methodology for secure communications in wireless sensor networks," IEEE Trans. Parallel Distrib. Syst., vol. 18, no. 12, 2007, pp. 1784–1796.

[27] Y. Sei and S. Honiden, "Distributed detection of node replication attacks resilient to many compromised nodes in wireless sensor networks," in Proc. 4th Annual International Conference on Wireless Internet, Nov. 2008, p. 28.

[28] M. Conti, R. Di Pietro, L. V. Mancini, and A. Mei, "A randomized, efficient, and distributed protocol for the detection of node replication attacks in wireless sensor networks," in Proc. ACM MobiHoc. ACM Press, Sep. 2007, pp. 80–89.

[29] B. Zhu, V. Addada, S. Setia, S. Jajodia, and S. Roy, "Efficient Distributed Detection of Node Replication Attacks in Sensor Networks," in Proc. 23th ACSAC, 2007, pp. 257–267.

 ISBN: 978-1-61208-298-1

Secure Distributed System inspired by Ant Colonies for Road Traffic Management in Emergency Situations

A. Peinado, A. Ortiz-García, J. Munilla
Dept. Ingeniería de Comunicaciones
E.T.S. Ingeniería de Telecomunicación, Universidad de Málaga
Málaga, Spain
apeinado@ic.uma.es, aortiz@ic.uma.es, munilla@ic.uma.es

***Abstract*—In this work, we present a distributed system designed for road traffic management. The system is inspired by the behavior of the ant colonies. The distributed design responds to the particular limitations of an emergency situation; mainly, the fixed infrastructures are out of service because no energy supply is available. The implementation is based on the VANET facilities complemented with passive RFID tags or GPS localization. The vehicles can use the information of previous vehicles to dynamically decide the best path. A scale prototype has been developed to validate the system. It consists of several small size robotic vehicles, a test road circuit and a visual monitorization system. The security of the system is provided by a combination of data aggregation and reputation lists.**

Keywords - security;emergency;reputation lists;ant colonies; road traffic;RFID;VANET;robotic vehicles

I. INTRODUCTION

Currently, road traffic management systems are based on centralized strategies that use a variety of technologies, such as cameras and sensors to obtain information about the actual traffic state. The data are analyzed in a data processing center where decisions are made and communicated to the operational services and the drivers through panels and displays located on the road itself. An example is the panels that inform drivers of the maximum speed on motorways and highways to access certain cities, which varies depending on traffic conditions [3], [15].

In emergency situations, it is very common that telecommunications networks do not work properly, mainly affected by falling energy sources that feed the fixed infrastructure of these networks. Consequently, although the data processing centers have alternative power sources, and even their own networks for data communication, the lack of energy affects the cameras and sensors in a very high percentage.

In this paper, we propose a system for road traffic management which can continue to operate when the energy supply is not available for the infrastructure, in short, a system that does not rely on such infrastructure to reach the objective. This system is mainly supported on the advantages of vehicular networks (VANETs), as no external energy supply is needed.

Vehicular ad hoc networks (VANETs) [6] are presented as a generation of networks oriented to improve the safety and driving comfort. These networks allow connectivity among mobile hosts (vehicles). This way, vehicles in a VANET can share information to each other in a short range by using WAVE wireless protocol, *i.e.*, the IEEE 802.11p technology [7]. This communication between nodes is named V2V (for vehicle-to-vehicle).

The use of a light infrastructure or a backup network is also considered in VANET scenarios as a mean to improve the services offered by the network providing the so called vehicle-to-infrastructure communication (V2I or I2V).

The vast majority of defined and developed applications in VANETs are related to road safety, and are based on the transmission of information between vehicles on accidents, congestion levels in certain parts of the road, signaling dangerous sections, etc.. Sometimes the information is generated in a vehicle that begins to transmit it to the others. Other times the information is generated in the infrastructure and transmitted through the Road side units (RSU) [1].

These architectures also allow the development, in a natural way, of traffic management systems getting information about the current traffic and informing drivers instantly. However, the use of the infrastructure represents a risk to consider in emergency situations [15].

Accordingly, the traffic management system should be based only on V2V communications. We proposed such a system, in which every vehicle decides the path at each node (crossroad) using the information provided by previous vehicles following a procedure inspired by the behavior of the ant colonies.

Section II describes the proposed model for road traffic management inspired by the ant colonies. Section III focuses on the security aspects of the proposed system. Sections IV and V deal with the implementation details and a scale prototype performed to validate the system and demonstrate its operation. Section VI describes the results and future work.

II. MODEL INSPIRED BY ANT COLONIES

Taking into account the particular conditions of emergency situations, it seems reasonable to implement a management algorithm that can be performed in a distributed

 ISBN: 978-1-61208-298-1

manner, *i.e.*, through the collaboration of the vehicles themselves, without the need for fixed infrastructure.

Therefore, in this paper, we propose the use of a system based on the behavior of ant colonies [5]. While there are numerous algorithms for transport and logistics based on this principle, all of them are applied in simulation and optimization tasks, in order to calculate the *a priori* best routes [4],[17]. The present proposal is a direct implementation of the algorithm used by ants to search for food. This implementation allows the vehicles to choose the suitable route in real-time while drive on the road.

In general terms, each ant initially chooses a random walk from the nest in search of food. When an ant finds food, it leaves a trail of pheromones on their way back to the nest so that the other ants can follow it. This substance disappears as time passes. This fact allows the ants always decide to follow the path with the higher level of pheromone. Thus, all ants end up finding the shortest path to the food.

This operating principle can be applied, with some minor modifications, to control the road traffic, providing every vehicle with the capacity to leave a "trace" and detect the presence of "pheromone"; that is, providing a distributed mechanism without the help of a fixed infrastructure. The goal, in this case, would not be to find the shortest path, but properly distribute the traffic so as to reduce overall congestion level, and sometimes, as in emergency cases, help the vehicles to find the available routes.

The main modifications to be applied to the original algorithm are the followings.

A. Route selection

Provided that the vehicles have the ability to detect a trail of pheromones, unlike ants, vehicles should choose the path with lower level of pheromones. In this way, the vehicles could be distributed uniformly avoiding large concentrations along the same route.

It is important to note that the choice of the route is done by the driver finally. This means that the algorithm only informs the driver about the route he should take. If the driver chooses not to follow the recommendation, the system also operates properly as the pheromone trail detected by other vehicles will always corresponds to actual state of the road, as the vehicles leave the trail on the route they are really driving.

Moreover, the criterion for choosing the recommended route may be configurable in order to meet the different needs of users. Sometimes the vehicles may be interested to take the road less congested and other times, in emergency situations, it may be more efficient that everyone follows the same route, because it is probably the only one available.

B. Trail generation

The trail of pheromone should be emitted and maintained in a discontinuous way. Instead of leaving the trail along the whole route, the vehicles should generate a given amount of pheromones concentrated on a specific point of the road. That amount should be the equivalent to the pheromones emitted during a bounded path. In this way, the vehicles detect the trail of pheromones only in those specific points.

This modification requires preliminary planning to analyze the strategic points where to generate and detect pheromones.

C. Pheromones storage

The level of pheromones cannot be stored on the road, but in the vehicles for reasons of synchronization, as described in the next section. Hence, the proposed algorithm for traffic management is a distributed implementation of the algorithm used by ant colonies to find food.

III. IMPLEMENTATION DETAILS

The algorithm presented in the previous section mainly relies on V2V communications. This resource allows the vehicles to send and receive messages representing the generation and detection of pheromones.

However, an auxiliary system is mandatory to manage the locations where the vehicles have to generate and/or detect the pheromones. In this paper, we consider two different proposals: RFID and GPS location. Next, this and other implementation aspects are described.

A. Location system

The radio frequency identification (RFID) systems have been outlined from the beginning as a way to extend the functionality of the Internet of Things (IoT) [2] to all kinds of objects, adding one tag with certain information that can be read and/or modified through a radio interface [9].

These systems basically consist of reader devices that interrogate some passive devices (tags), which respond by using energy they take from the field applied by the reader. Communication between the reader and the tag is established wirelessly by inductive coupling in the case of HF tags or backscattering in the case of UHF tags.

Since the communication between the tag and the reader is wireless and the tags do not need a power supply (they are passive), RFID system is a perfect candidate for our system. RFID provides I2V and V2I connections. Applications for toll control and automatic detection of traffic signs are supported by RFID [8],[13], [14].

Moreover, the Global Positioning System (GPS) is the most widely used location system based on satellite. At present, we can consider that the great majority of vehicles have a GPS signal receiver. In emergency situations, the vehicle may continue to use the system [16]. Therefore, it constitutes a good choice for the management system proposed in this paper.

In any case, RFID or GPS are necessary to establish the points of generation and detection of pheromones and route selection. These places are called control places.

B. Definition of control places.

The definition of control places is performed by means of the identification of the most significant nodes of the road. These nodes are the most important junctions or roundabouts. At each input, all possible outputs are considered.

Input locations are associated with locations where vehicles have to decide where to continue the route. To do

 ISBN: 978-1-61208-298-1

so, the vehicles consult the data received from other vehicles. Output locations are used to indicate the vehicles when they must emit pheromones (Fig. 1). This emission consists of a message that primarily contains two values:

- ID_{SVeh} : Identification of the vehicle that caused the message
- ID_{loc} : Identification of the location that corresponds to the sending of the message

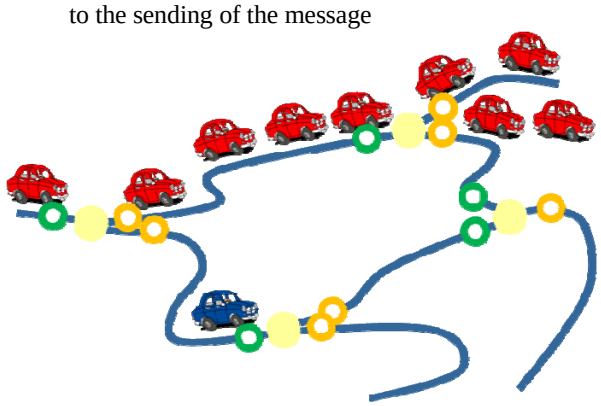


Figure 1. Identification of relevant places

C. Vehicle configuration

Pheromone levels generated by the vehicles are not stored on the road, as would ants, but kept updated in each vehicle independently. That is, each vehicle controls every pheromone emission produced in its area of interest.

To do so, vehicles use a register (internal local variable) for each output. We consider that, in this first stage of definition and prototyping, the vehicles know the control places before they begin to apply this system. In future stages, currently under work, we will consider that the control places will be dynamically discovered by the vehicles and incorporated to their internal maps.

The registers are initialized to zero. Each time a vehicle drives through a control place, a message that contains ID_{SVeh} and ID_{loc} is broadcasted, following the usual scheme employed in VANETS. When a vehicle receives a message, the register associated to ID_{loc} is updated.

Updating registers is not merely cumulative because we must take into account the effect of disappearance (evaporation) of pheromones in function of time. For this reason, it is essential to know the time between messages related to the same route.

This time control is locally (internally) performed. That is, the vehicles' clocks are not synchronized. The important thing is to know the arrival time difference of the messages related to the same route. Therefore, the vehicle will store the last timestamp received in each register.

D. Pheromones generation

Unlike ants, the vehicles generate pheromones at discrete points. In any case, the general behavior is very similar, in such a way that the generated pheromones are added to the existing quantity at every discrete point. At the same time, the accumulated amount of pheromones disappears as time passes. Hence, the generation and updating processes are as follows:

- **Generation process**. When a vehicle drives over a generation point, it sends a message with the identification number ID_{loc} of that location.
- **Disappearing effect**. When a vehicle drives over a generation point, it compares the timestamps of all registers with its current local time. In this way, the vehicle computes the time elapsed from the last pheromones generation. The time difference multiplied by a constant coefficient is applied to reduce the value of every register. Non-negatives values are not permitted.
  The rest of vehicles apply the same reducing algorithm when they receive the message. Each vehicle computes time differences and reduces the values of registers. Non-negatives values are not permitted.
- **Increase of pheromones**. Once the disappearing effect has been applied, every vehicle included the one that generates the pheromones, increases the register value in a constant quantity of 50. This value maybe modified. In the experiments performed 50 has resulted a suitable value.

E. Route selection

Each time a vehicle drives over a selection location, it performs the following operations:

- **Disappearing effect**. The procedure described in previous phase is applied to update the pheromones status of every decision node.
- **Selection**. The vehicle chooses the output road with the lowest level of pheromones. Actually, the vehicle only makes a recommendation. The driver chooses the output path following the recommendation. The driver could choose a different route. In any case, the final decision will be reflected on the system by means of pheromones generated when the vehicle drives over the route selected.

IV. SECURITY ANALYSIS

The ant algorithm has proven to be effective and ensures its adaptation to dynamic changes in the routes. This section discusses the security aspects of the proposed algorithm.

A. General considerations.

The proposed algorithm for traffic management is based on V2V message exchange through the VANET. Therefore, we should apply the same security mechanisms as other messages that are used to improve road safety. As described in [10] and [11], authentication is the main security mechanism to detect false messages. Confidentiality of information is not a priority because the main goal is to give maximum exposure while ensuring the authenticity of the same. Accordingly, the messages described in section III incorporate a signature to allow verification of the authenticity. The signatures are generated by means of asymmetric or identity-based cryptosystems.

 ISBN: 978-1-61208-298-1

Finally, since the algorithm of the ants has local significance, the life time of the messages must be small. That is, the number of hops should be limited in the routing protocol to a small value.

B. Reputation Lists and Data Aggregation

Authentication is the first barrier to prevent fraudulent messages. However, it is not enough since an authentic message can convey a false content devised by a user who wants to sabotage the system.

An attacker could try to overload a route sending false messages always indicating the same location, making believe that there are many vehicles on the road. In this way, the attacker would be reserving a path for himself, as other vehicles not choose a route congested.

In [11], reputation lists are proposed to identify vehicles that generate false content. Specifically, two types of lists were proposed: the individual reputation list (IRL) that internally updates each vehicle when it detects a fraudulent message, and global reputation list (GRL) that are shared with all vehicles with the help of some RSU. Messages generated by vehicles in one of these lists will be discarded and not retransmitted.

In this paper, we only use IRL because GRL rely on infrastructure and would not work in emergency situations. The IRL generation process is as follows:

- When a vehicle wants to emit pheromones sends a message containing its own identity ID_{SVeh} and its location ID_{loc}.
- Vehicles that receive the message check if ID_{SVeh} is in IRL. If so, then the message is discarded and not retransmitted.
- If ID_{SVeh} is not in the IRL, the vehicle checks whether ID_{loc} is consistent with their location, that is, that the message has been delivered from a nearby. Otherwise ID_{SVeh} is included in the IRL and the message is discarded and not retransmitted.

In a VANET, relaying messages cannot always be done at the right time. When a vehicle is not close to another vehicle, it cannot communicate. In these cases, the usual mode of operation of VANETs is to store the message and broadcast it later when another vehicle is found. This behavior affects the IRL, as a vehicle that receives a delayed message will have a different (distant) location, from which the message was originated, and therefore, will be considered as false.

To avoid this effect, this paper proposes aggregation of signatures as a method of message verification to complement the IRL. In [10], it is proposed the use of aggregation for warning messages on accidents and incidents affecting traffic safety. A vehicle that receives a message will add its signature only if it detects the same incident on the road.

This scheme is not applicable, as is, in the case of traffic management proposed in this paper. The ants algorithm only needs to send and receive messages containing the location of vehicles. Thus, vehicles can only detect if a message carrying ID_{loc} was sent from near or far to that location.

The combination of IRL and aggregation of signatures is performed as follows:

- When a vehicle wants to emit pheromones sends a message containing its own identity ID_{SVeh} and its location ID_{loc}. The message will be sent only if neighbor vehicles exist. Delayed messages will not be sent.
- Vehicles that receive the message check if ID_{SVeh} is in IRL. If so, then the message is discarded and not retransmitted.
- If ID_{SVeh} is not in IRL, the vehicle checks wether ID_{loc} is consistent with their location, that is, that the message has been delivered from a nearby. If so, the vehicle adds his signature to the message and retransmit it.
- If ID_{SVeh} is not in IRL, and ID_{loc} is not consistent with the current location, then if the message contains aggregated signatures, the message is accepted and retransmitted. Otherwise, ID_{SVeh} is included in the IRL and the message is discarded and not retransmitted.

C. Analysis of Possible Attacks

In a VANET, the main security threats come from fake content generation. An attacker could send messages indicating that is circulating in a location other than the one actually has, with intent to influence the overall perceived congestion on other vehicles.

In the particular case of the algorithm proposed, the security threats are less serious because it has implicit security. This means that the operation of the system itself makes attacker benefits are minimal.

Then we analyze security of the proposed system against possible attacks.

- **False messages**. They are detected by means of the usual authentication mechanism in VANETs.
- **False content** (fraudulent messages). They are detected by means of the combination of IRL and signature aggregation described in the previous subsection.
  In any case, if a false content escapes to the control of the proposed mechanisms, the implicit security of this traffic management algorithm minimizes the risk. So, if a vehicle sends a false content, the effect on the management system will be minimal, because if a single message gets to change the decision of other vehicles is because all the routes had a similar level of congestion. Equivalently, one could say that a single ant does not alter the behavior of the entire colony.
- **False content flooding**. Since a single message can not affect the rest of the vehicles, an attacker could generate a large number of fake messages containing the same location to achieve his goal. The procedure based on IRL and signature aggregation detects these fake messages. Furthermore, in the case that the procedure does not work properly, the situation is easily detected because the vehicles receive

 ISBN: 978-1-61208-298-1

messages with the same ID_{loc} and ID_{SVeh} in a short period of time. If the attacker decided to space in time the messages, then the success probability would decrease due to the effect of the disappearance of the accumulated pheromones.

- **Conspiracy**. A large number of attackers may collude to send messages with the same but different ID_{SVeh} ID_{loc}. If the messages are false, the IRL in combination with signature aggregation detect them. If this mechanism does not work properly, this type of attack is not the most efficient because it needs a large number of organized attackers to book the same route. They themselves would collapse the route.
- **Discarding an aggregate message**. The attacker will not retransmit the authentic messages, with the intention to perform a denial of service. This attack is not efficient. If the vehicle density is low, then the message will not be retransmitted but the influence on the traffic control will be negligible. If the density is high, there is a very high probability that another vehicle retransmit it.

V. PROTOTYPE

The model proposed in the previous section has been implemented as a scale prototype with two major objectives in mind: to broadcast the advantages of VANETs, and to provide a demonstrator platform where the effects on traffic management can be easily perceived. In this sense, we have chosen a real scenario determined by the road paths between two major hospitals at Málaga city, Spain.

The prototype is composed of one road circuit, several robotic vehicles, and a monitorization system. The communications between the actors are performed by means of RFID tags and readers and ZigBee modules.

A. Test road circuit

The road circuit has been printed on a 150 x 90 cm paper size. The dimensions are determined by the maximum width of the print area of the available printer device (90 cm), the size of vehicles and a value of length allowing to handle the prototype.

The road circuit has been previously designed using Microsoft Office Visio 2007 paying attention not only to the particular road paths to test, but also to aesthetic criteria in order to enhance the visual effect. Different road circuits, following the same guidelines, have been previously used to demonstrate the performance of other VANET functionalities [13].

The printed road circuit used in this protocol corresponds to a simplification of a real scenario located in Málaga city (Spain) between two major hospitals, "Carlos Haya" University Regional Hospital and "Virgen de la Victoria" University Clinical Hospital. Fig 2 shows the real map. Fig 3 shows the simplification that contains only the main avenues and streets, but maintaining the traffic senses. The circuit includes six decision nodes, marked with colors.

On the other hand, as one can see in Fig 3, the road is printed in white color (90 mm width) with a centered black line (20 mm width) to help the vehicles in the auto-drive functionality.

Figure 2. Real scenario (source: Google Maps)

Figure 3. Prototype test road

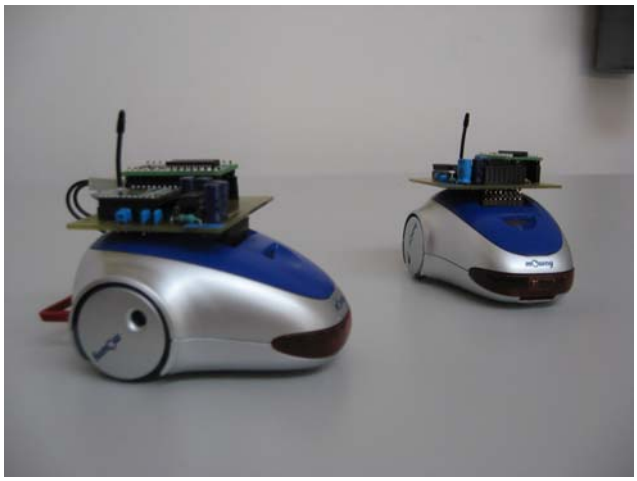


Figure 4. Robotic vehicles

B. Robotic Vehicles

The vehicles are little size robots with the ability to track a black line. This functionality is governed by a main PIC microcontroller and a secondary microcontroller dedicated to control the wheels movements.

 ISBN: 978-1-61208-298-1

The vehicles have been constructed using the educational Moway mini-robot as the starting point [12]. An additional PCB with an RFID reader and a Zigbee transceiver has been designed, developed and incorporated to every robot. Finally, the RFID antenna has been integrated in the own robotic vehicle (see Fig 4).

The robotic vehicles have been programmed to read the RFID tags located at the road and send and receive information over the ZigBee interface.

C. Wireless communications

The communication between the road and the vehicles is implemented by means of RFID technology. Read-only EM4100 passive tags operating at 125 kHz are located in the middle of lanes near the decision nodes. The vehicles are provided with the SM125-M1 RFID reader module by *SonMicro Electronics*.

V2V communications are implemented over a ZigBee interface. ZigBee is used, instead of another wireless technology, due to the reduced size of the prototype elements. Each vehicle is provided with a XBee module by *Digi International, Inc.* This module allows the vehicles the reception and transmission of pheromones.

The same Xbee module has been employed to develop an infrastructure point, in such a way that V2I communications could be performed. Actually, the model proposed in this work does not need the support of any infrastructure. However, the monitorization system, developed as part of the prototype, uses this service to obtain the status information from the circuit.

Finally, I2V communications are not implemented because the monitorization system does not return any information to the vehicles.

VI. CONCLUSIONS

We have proposed an algorithm, based on ant colonies, for road traffic management. The implementation of the algorithm does not rely on fixed infrastructures in order to operate in emergency situations. It only uses the VANET V2V communications and location systems that do not require contact with a fixed infrastructure.

The algorithm uses signature aggregation and reputation lists to ensure system security. Furthermore, the algorithm has an implicit security that minimizes the risks in case of attacks.

A scale prototype has been designed and implemented to validate the algorithm using RFID location system.

However, this proposal needs further work to optimize some configuration parameters and perform simulations to provide data about the final distribution of congestion.

Currently, identification of the locations is static. At a later stage, we will consider the integration of the proposal with dynamic maps, without prior planning. However, the current static solution is completely valid for emergency situations.

ACKNOWLEDGMENT

This work was partially supported by Ministry of Science and Innovation under Project TIN2011-25452 ("TUERI: Technologies for secUre and Efficient wiReless networks within the Internet of things with applications in transport and logistics") and by Universidad de Málaga. Campus de Excelencia Internacional Andalucía Tech.

REFERENCES

[1] Asset-road Project, 2010, http://www.project-asset.com [retrieved: June, 2013]

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey", Computer Networks, 2010, vol. 54, p. 2787-2805.

[3] E. S. Davison, H. Shmizu, and H. Naraki, “Traffic Signal control algorithms of a traffic network”, Proc. of IFAC Symposium Large Scale Systems, LSS’95, London, 1995, pp. 509-514.

[4] Y. Fang and L.B. Wu, "Parallel Ant Colony Algorithm for the Logistics Scheduling Problem”, International Conference on Multimedia Communications (Mediacom), Aug. 2010, pp.116-119.

[5] A. Gutiérrez Martín and F. Monasterio-Huelin, "Algoritmos de rastreo inspirados en colonias de hormigas", Actas de las XXVII Jornadas de Automática, 2006, pp 299-305.

[6] H. Hartenstein, VANET: vehicular applications and inter-networking technologies, John Wiley & sons, 2009.

[7] IEEE 802.11p, “Draft standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements: Wireless access in vehicular environments.” IEEE P802.11p/D9.0, September 2009.

[8] Y.M. Lee, C.G. Yoo, M. Park, M. Kim, and M. Gerla, “Installation and Evaluation of RFID readers on Moving Vehicles”, in Proc. of the sixth ACM international workshop on VehiculAr InterNETworking,VANET’09, 2009, pp. 99-108.

[9] S.B. Miles, S.E. Sarma, J.R. Williams, RFID. Technology and applications, Cambridge University Press, 2008.

[10] J.Molina, P.Caballero, and C.Caballero, “Data Aggregation for Information Authentication in VANETs”, Information Assurance and Security Letters, 1, 2010, pp. 47-52.

[11] J. Molina, C. Caballero, and P.Caballero, “Reputation Lists and Groups to Promote Cooperation”, International conference on Computer Systems and Technologies, CompSysTech’11, 2011, pp. 460-465, Vienna, Austria.

[12] Moway mini-robot. http://moway-robot.com [retrieved: June, 2013]

[13] J. Munilla, A. Ortiz, and A. Peinado, "Robotic vehicles to simulate RFID-based vehicular ad hoc networks", Proc 3rd International ICST Conference on Simulation Tools and Techniques (SIMUTools 2010), Torremolinos, Málaga, Spain, 2010, pp. 49,1-49,2.

[14] A. Ortiz, A. Peinado, and J. Munilla, “A Scaled Test Bench for Vanets using RFID Signalling”. in Comp. Intelligence in Security for Information Systems, 2009

[15] Roughan and O’Donovan – AECOM Alliance, Goodbody Economic Consultants, NRA National Roads Traffic Management Study, National Roads Authority Publications, Ireland 2011

[16] S. E. Shladover and S. K. Tan, “Analysis of vehicle positioning accuracy requirements for communication-based cooperative collision warning”, J. Intell. Transp. Syst., Technol., Plan., Oper., vol. 10, no. 3, 2006, pp. 131–140.

[17] D. Xiao-Hua Yu, "Traffic signal optimization using Ant Colony Algorithm", Neural Networks (IJCNN), The 2012 International Joint Conference on , June 2012, pp.1-7, 10-15.

 ISBN: 978-1-61208-298-1

Efficient Image Encryption and Authentication Scheme Based on Chaotic Sequences

M. Farajallah, Z. Fawaz, S. El Assad, Member IEEE
IETR / LUNAM University/LI University
Nantes, France
safwan.elassad@univ-nantes.fr
mousa.farajallah@etu.univ-nantes.fr

O. Deforges
IETR / INSA Rennes
Rennes, France
olivier.deforges@insa-rennes.fr

***Abstract*— Many image encryption algorithms based on chaos have been proposed since 1989. Most of them are slow and use a secret key of encryption/decryption independent of the plain image. We introduce a new fast and secure image encryption and authentication scheme based on the chaotic sequences. The main structure of the proposed algorithm consists of two layers (substitution and permutation) for encryption and decryption image values, and two components: a hash function and a chaotic generator. The hash function generate the secret key of the chaotic generator that provides the dynamic keys for the substitution-permutation layers, while the secret hash key is used to authenticate the decrypted image. The proposed algorithm is at least ten times faster than the AES (Advanced Encryption Standard) algorithm and faster than many chaos-based encryption algorithms of the literature. Furthermore, it is very secure against chosen/known plaintext and statistical attacks because the key of the chaotic generator is dependent on the plain-image. Simulation results show that the efficient performance is reached in only one round.**

Keywords- Chaos-based cryptosystem; Image encryption/authentication algorithm; Chaoti cgenerator; security analysis.

I. INTRODUCTION

Chaos based encryption algorithms are extremely sensitive to the initial conditions and system parameters [1][2][3]. This helps in producing the required confusion and the diffusion effects. Also, normally they are faster than the standard encryption algorithms because of the low complexity of their structures [4]. For that and during the last decade, a number of chaos-based encryption algorithms have been proposed [5][6]. Guanrong et al, introduced a symmetric image encryption scheme based on 3D chaotic maps, he generalized the 2D chaotic maps into 3D chaotic maps to shuffle the pixel positions and to confuse the relationship between the plain and the cipher images. The speed of this algorithm is not high for real time applications, and also the propagation error is large, since it works in cubes and each cube affects all other [6]. In his paper, Fridrich proposed a chaos-based encryption algorithm consisting of a permutation layer based on the Baker Map, following by a nonlinear feedback process based on a nonlinear feedback register. It is clear that Fridrich algorithm is time consuming and slow encryption scheme [7]. Song et al, presented a new image encryption scheme based on a new spatiotemporal chaos. From the security result was presented, it is clear that this algorithm has good security level, but it has also slow time of encryption/decryption related to the sort operation, which is well known that is a time consuming operation [8]. In [9], a fast and robust encryption algorithm for images has been proposed. The algorithm makes r rounds of an SP-network (Substitution-Permutation network) on each pixel using two PWLCM (Piecewise Linear Chaotic Map) maps. This algorithm is faster than previous algorithms but one of the weaknesses is the high error propagation caused by the used technique of perturbation. The paper is organized as follows, the next section describes the main structure of the proposed algorithm, and in its subsections the details of the cryptosystem components are described. Section 3 presents the security and time analysis results, while the conclusion part is at the last section.

II. GENERAL STRUCTURE OF THE PROPOSED CRYPTOSUSTEM

The general structure of the proposed cryptosystem is presented in Figure 1 for the encryption and in Figure 2 for the decryption. The encryption process consists of two layers. The first layer is the substitution one achieved by the skew tent map, followed by a permutation layer realized by the 2D cat map. The both layers need a dynamic key during the encryption and decryption processes, these keys are provided from a simplified version of the chaotic generator from El Assad et al [10]. The key of the chaotic generator is derived from a hash function (we use SHA-1 (Secure Hash Algorithm) for testing and we are working on SHA-256) [11], whose inputs are the secret hash key and the plain image. So, the confusion-diffusion properties of the cryptosystem are reached and the immunity against known-chosen plaintext is obtained.

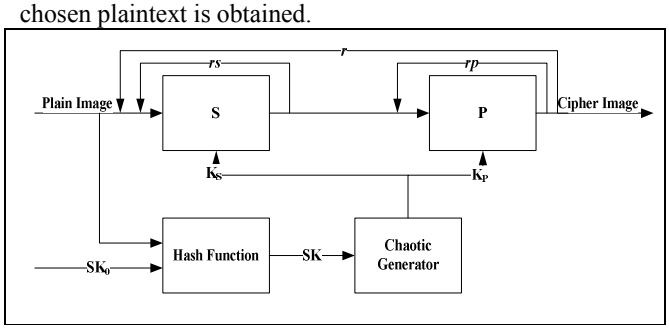


Figure 1. Encryption Parts of the Proposed Cryptosystem

 ISBN: 978-1-61208-298-1

The decryption process (see Figure 2), is based on an inverse permutation layer achieved by the same 2D cat map, following by the inverse substitution achieved by the inverse skew tent map. During this decryption process the rounds of each layer are starting in reverse order and also the dynamic keys are using in reverse order. Notice that, from the estimated plain-image and the hash secret key we can determine whether the estimated plain-image (the decrypted image) is the same one sent.

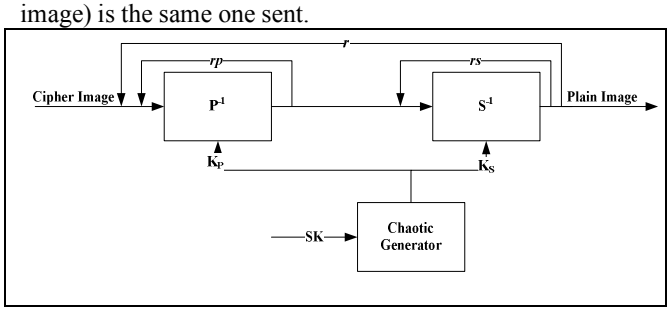


Figure 2. Decryption Parts of the Proposed Cryptosystem

The encryption steps are described in details in Figure 3. The plain image is divided into *NB* blocks, each one contains *BS* bytes. The first step is to generate the dynamical keys for the substitution-permutation layers. To do this, the plain image and the secret hash key are used as input for the hash function, and then the hash function produces the secret key of the chaotic generator, that provides the dynamic keys at each round of the cryptosystem. After that, for each plain block, first we use the CBC (cipher-block chaining) mode [12] (bit-wise XOR operation between the current plain block and the previous ciphered one, while in the first block the previous ciphered block is the initial random block IV (initialization vector)) and then we apply the substitution layer *rs* times and the permutation layer for rp times. Finally, these processes (CBC, substitution, permutation) are repeated *r* rounds and for each round a new dynamic keys are generated from the chaotic generator to be used in both layers, and so on.

Figure 4 shows the decryption part of the proposed cryptosystem. The decryption process is similar to the encryption one; the differences are in the inverse substitution and the reverse permutation layers, first of all, the dynamic keys are used in reverse order in both layers. Second, the permutation layer is starting the recover process from the last byte of the current block until the first one. Third, all counters will be starting in reverse order. Finally, the CBC mode function is used at the end of decryption of each block. The inverse substitution layer is starting as normal from the first to the last byte. The receiver uses the decrypted image to test the authentication source, and to test the integrity of the message.

The decryption and authentication processes suppose that the secret hash key and the secret key of the chaotic generator are known (transmitted in secret manner by the sender to the receiver.

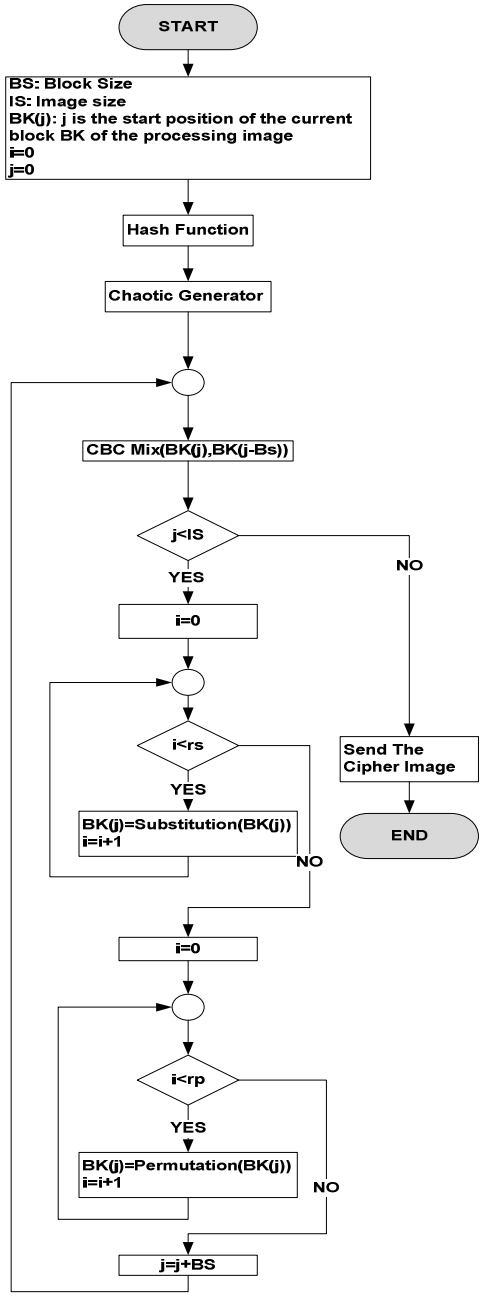


Figure 3. Encryption Components of the Cryptosystem

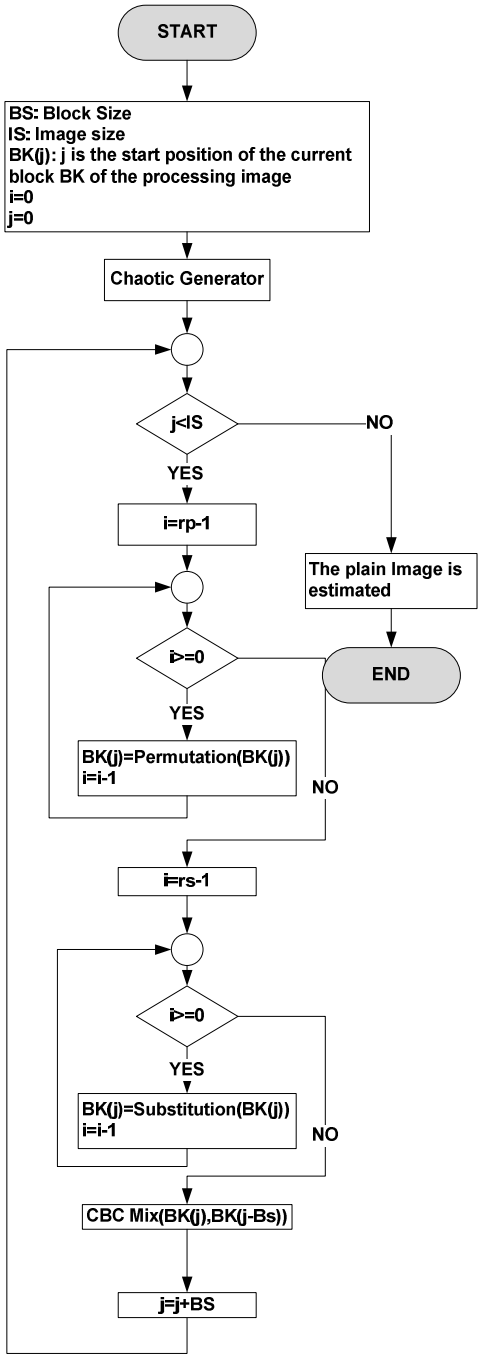


Figure 4. Decryption Components of the Cryptosystem

A. *Substitution Layer*

The substitution layer of this cryptosystem is based on the FSTM (Finite State Skew Tent Map). This layer is making some nonlinear transformation in the image data; the nonlinear step implies the cryptosystem to be more resistance against differential cryptanalysis attacks. This layer works on byte-by-byte transformation and it changes the value of the byte depending on a chaotic parameter a , which comes from a robust chaotic generator. The mathematical model of the encrypted part of FSTM is [5].

$$Y = S_a(X) = \begin{cases} \left\lceil \frac{Q}{a} \times X \right\rceil & ,0 \le X \le a \\ \left\lfloor \frac{Q}{Q-a} \times (Q-X) \right\rfloor + 1 & ,a \prec X \prec Q \end{cases} \quad (1)$$

Since FSTM function is a bijective one, it means that the FSTM function is invertible one, the inverse equations at the decryption part are:

$$X = S_a^{-1}(Y) = \begin{cases} \xi_1 & ,\theta(Y) = Y and \frac{\xi_1}{a} \succ \frac{Q-\xi_2}{Q-a} \\ \xi_2 & ,\theta(Y) = Y and \frac{\xi_1}{a} \le \frac{Q-\xi_2}{Q-a} \\ \xi_1 & ,\theta(Y) = Y+1 \end{cases} \quad (2)$$

Where

$$\xi_1 = \left\lfloor \frac{a}{Q} \times Y \right\rfloor \quad (3)$$

$$\xi_2 = \left\lceil \left(\frac{a}{Q} - 1 \right) \times Y + Q \right\rceil \quad (4)$$

$$\xi_3 = \left\lceil \frac{a}{Q} \times Y \right\rceil \quad (5)$$

$$\theta(Y) = Y + \xi_1 - \xi_3 + 1 \quad (6)$$

The structure of the dynamic key during the substitution process is:

$$K_s = \left[K_{s_0} \| K_{s_1} \| K_{s_2} \| K_{s_3} \| .. K_{s_{r-1}} \right] \quad (7)$$

$$K_{s_j} = a_j \quad (8)$$

Where r is the number of rounds of each block, and inside each round the substitution layer is repeated *rs* times.

Q=substitution processing unit-1

We chose the substitution processing unit to be 256 bits

$1 \le a_j \le Q \qquad j = 0,1,2,..r-1$

The total number of blocks in the plain image is $NB = L \times C \times P / Block\ size$, here L, C and P are the number of lines, the number of columns and the number of plains. The $Block\ size$ variable is used to save the size of the block in bytes (here 256 bytes).

The simplified version of the El Assad et al chaotic generator produces a 32-bit samples each calling time, we chose to set the substitution layer key to be 8-bit (image data is represent in one byte). Each sample of the chaotic generator will produce 32 bits, so we have two options:

1- Take the first 8 bits from each sample to be used as the dynamic key and skip the remaining 24 bits, if the first 8 bits are zero, then we will take the next 8 bits and so on.

 ISBN: 978-1-61208-298-1

2- Divide the 32 bits into 4 bytes and make xor operation between the 4 bytes to produce the required dynamic key.

We chose to work according to the first option, since the dynamic key must be greater than zero and it is the best case for this condition in terms of the time and the probability of getting zero output.

Since (1) and (2) are time consuming and the range values of the input, the output and the dynamic keys of those equations is limited in [0-255], we use them to produce a lookup tables in order to reduce the execution time of the substitution and inverse substitution operations. For comparison, the both versions (substitution, inverse substitution by equations and by lookup tables) are implemented.

B. Permutation Layer

The permutation process changes the pixel position on the block under the test without changing its value. In our proposed cryptosystem, the permutation process is based on the modified version of the basic 2D Cat Map [8]:

$$\begin{bmatrix} i_n \\ j_n \end{bmatrix} = Mod\left(\begin{bmatrix} 1 & u \\ v & 1+u\times v \end{bmatrix}\begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} r_i + r_j \\ r_j \end{bmatrix}, \begin{bmatrix} M \\ M \end{bmatrix}\right) \quad (9)$$

Where (i_n, j_n) are the new row and column position of the old (i, j) byte position inside the *block* of size $M \times M$ bytes after applying the 2D cat map. u , v , r_i and r_j are the system parameters in the range of $[0, M-1]$. The last two parameters are added to the basic model to overcome the fixed point problem of the basic 2D cat map model.

The structure of the dynamic keys during the permutation process is:

$$K_p = \left\lfloor K_{p_0} \| K_{p_1} \| K_{p_2} \| K_{p_3} \| .. K_{p_{r-1}} \right\rfloor \quad (10)$$

$$K_{p_j} = \left[u_j \| v_j \| rl_j \| rc_j \right] \quad (11)$$

It is clear from (9) that the determinant of the Jacobean matrix of this model is 1, and so the 2D cat map process is a bijective. From the modulo operation we can conclude that the 2D cat map function is non-invertible one, but it is reversible, by applying the permutation loop in reverse order, that means beginning of the decryption process from the last byte inside the encrypted block, so the dynamic keys are used also in reverse order.

The 2D cat map is one-to-one function (bijective), which means every point of the square matrix can be transferred to exactly one unique point, from this fact we can replace the swap operation of (9) by the copy operation during the permutation layer to reduce the execution time.

The required dynamic keys bits for the permutation layer are giving by the following equation:

$$q = \log(M) \quad (12)$$

Where q is the number of required bits for each sub dynamic key, so, we need for each block $4 \times q \times r$ bits from the chaotic generator. During the permutation layer it is not necessary to have all sub-keys greater than zero, at least one of them is sufficient to be different of zero, because the probability to have the four sub-keys equal to zero is zero.

III. Time and Security Analysis

Time and security analysis is the most important part of any chaos based cryptosystem. Performance analysis of the proposed cryptosystem is provided in the first subsection, while in the next subsections we present the experimental results of testing different attacks.

A. Time Analysis

We applied time test for our proposed algorithm based on both versions (with/without lookup table) and AES algorithm using a C compiler of 3.1 GHz Intel processor Core ™ i3-2100 CPU, 4GB RAM, and Windows 7 32-Bit Operating System. The proposed algorithm was applied to an image file Boat.bmp of size $256 \times 256 \times 3$. Table 1 presents the average time of applying the two versions of the proposed algorithm for $(r = 1, rp = 1, rs = 1)$ and for the AES algorithm. We observe that, the proposed algorithm with lookup table version at least (in average that means without warm up time that happen on the first execution) is 10 times faster than AES algorithm that we applied.

Remark: we have used the AES algorithm given by the following website:

https://code.google.com/p/rikiglue/source/browse/src/frame/aes.cpp?spec=svn9239a0474d811daae909075568688a46134858c6&r=9239a0474d811daae909075568688a46134858c6.

TABLE I. Encryption and decryption Time of the proposed algorithm vs. AES algorithm in seconds

Algorithm Name	Encryption	Decryption
Proposed Algorithm Based Lookup table	0.0060	0.0058
Proposed Algorithm	0.0097	0.0316
AES	0.0643	0.0668

Also, from our knowledge, the proposed scheme is faster than many chaos based encryption algorithms of the literature. Specifically, we compare the obtained results with Yang cryptosystem results [13]. The security level of both cryptosystems is almost the same, while our proposed one is approximately twice faster than Yang scheme.

B. Plain Text Sensitivity Attacks

One bit change on the original plain text P_1 it becomes P_2, then, encrypts the both plain text using the same secret key will produce cipher texts C_1 and C_2 , if we calculate the number of different bits between C_1 and C_2 , then divide the result by the total number of bits, this is called the hamming distance value, in mathematical form is:

$$d_{Hamming}(C_1, C_2) = \sum_{K=1}^{L \times C \times P \times 8} C_1(K) \oplus C_2(K) \quad (13)$$

We executed our proposed algorithm on Boat.bmp of size $256 \times 256 \times 3$ for 1000 different secret keys to take the

average value of the hamming distance, which is *0.500009* and this value is the optimal value that can be reached for plaintext sensitivity attack.

C. Key Sensitivity Attack

In a similar procedure, we can apply the key sensitivity test by changing one bit on the secret key and encrypts P_1 using the original secret key, encrypts P_2 using the key after one bit change, this also produces C_1 and C_2. Then, we calculate three parameters: HD (Hamming distance), the NPCR (Number of Pixels Change Rate) and the UACI (Unified Average Changing Intensity) [14]. The following mathematical forms are used to calculate the last two parameters:

$$NPCR(C_1,C_2)=\frac{1}{L\times C\times P}\sum_{K=1}^{L\times C\times P}D(K)\times 100 \quad (14)$$

Where

$$D(K)=\begin{cases}1 & if\ C_1(K)\neq C_2(K)\\ 0 & f\ C_1(K)=C_2(K)\end{cases} \quad (15)$$

$$UACI(C_1,C_2)=\frac{1}{L\times C\times P\times 255}\sum_{K=1}^{L\times C\times P}|C_1(K)-C_2(K)|\times 100 \quad (16)$$

Notice that UACI and NPCR are calculated in bytes level while HD is calculated in bit level. We executed our proposed algorithm for 1000 different secret keys on the same image. The following table presents the results of the key sensitivity attack. As, we can see these values are near the optimal values.

TABLE II. NPCR, UACI AND HD VALUES FOR THE KEY SENSITIVITY ATTACK TEST

Test Name	Test Value
HD	0.50002
NPCR	99.6098
UACI	33.4667

D. Histogram Analysis

The histogram of the ciphered image of any crypto-system should be uniform; to test the uniformity distribution of the ciphered image we applied the chi-square test.

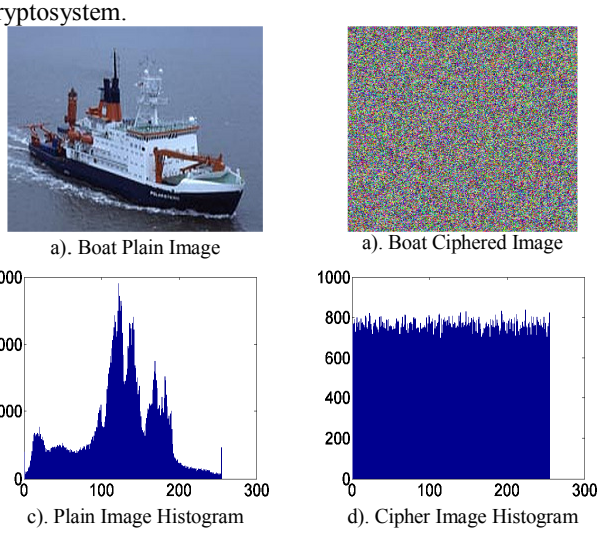
$$\chi^2_{\exp}\sum_{i=0}^{Ny-1}\frac{(O_i-E_i)^2}{E_i} \quad (17)$$

This test was applied on three different nature's images (Boat, Cameraman, and Jet) of size $256\times256\times3$. For a secure cryptosystem the experimental value must be less than the theoretical one, which is 293 in case of alpha=0.05 and num intervals=256. Table 3, presents the calculated chi-square value on each image.

TABLE III. CHI SQUARE TEST RESULTS ON THREE DIFFERENT IMAGES

Image Name	Chi-square Value
Boat	255.431
Cameraman	255.737
Jet	253.787

The histogram of the plain image and its ciphered one are presented in Figure 5, it is clear that the pixel values show a pattern in part c) of the figure which presents the histogram of the plain image, while in part d) the distribution of the pixel values are almost uniform and significantly different from the histogram of the plain image. As a result the statistical analysis of the histogram will be useless or most difficult for cryptanalysis of the ciphered images of this cryptosystem.



a). Boat Plain Image a). Boat Ciphered Image

c). Plain Image Histogram d). Cipher Image Histogram

Figure 5. Histograms of the Boat Plain Image and its Ciphered One

E. Correlation Analysis

One of the most difficult properties of the image encryption algorithms comes from the high correlation between adjacent pixels. To test the security of the proposed algorithm we randomly selected N = 10000 pairs of adjacent pixels in vertical, horizontal, and diagonal directions from the plain image and its ciphered one. Then we calculated the correlation coefficient according to the following equation [8]:

$$\rho_{xy}=\frac{\mathrm{cov}(x,y)}{\sqrt{D(x)\times D(y)}} \quad (18)$$

Where:

$$\mathrm{cov}(x,y)=\frac{1}{N}\sum_{i=1}^{N}(x_i-E(x))\times(y_i-E(y)) \quad (19)$$

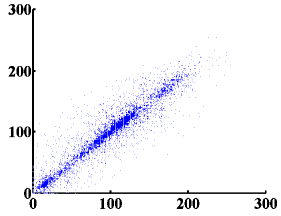
$$D(x)=\frac{1}{N}\sum_{i=1}^{N}(x_i-E(x))^2 \quad (20)$$

$$E(x)=\frac{1}{N}\sum_{i=1}^{N}x_i \quad (21)$$

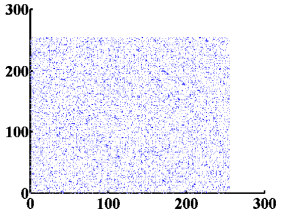
In the above equations, x_i and y_i are the values of the two adjacent pixels in the plain image and the corresponding ciphered image. Figure 6 shows the correlation coefficients of the adjacent pixels in vertical direction for both Boat plain image of size $256\times256\times3$ and the corresponding ciphered image; we omitted the figures of diagonal and horizontal directions since they are similar for vertical one. Table 4 presents the correlation values in the three directions for

 ISBN: 978-1-61208-298-1

Boat, Cameraman and Jet images of the same size $256 \times 256 \times 3$.



a-) Correlation of adjacent pixels in vertical from the Plain image



b-) Correlation of adjacent pixels in vertical from the Cipher image

Figure 6. Correlation Analysis of the Plain and Ciphered Images in Horizontal Direction

TABLE IV. CORRELATION COEFFICIENTS OF PLAIN AND CIPHER IMAGES

Direction	Plain image	Cipher Image	Image Name
Vertical	0.944191	0.008648	Boat
Horizontal	0.936227	0.008683	
Diagonal	0.892431	0.008371	
Vertical	0.869266	0.008267	Airplane
Horizontal	0.929387	0.007203	
Diagonal	0.900825	0.007230	
Vertical	0.980595	0.008538	Cameraman
Horizontal	0.970803	0.008434	
Diagonal	0.951154	0.009412	

It is clear from Figure 6, and table 4, that the correlation coefficients of the adjacent pixels in the plain and the cipher images are far apart, that means the cryptosystem success to convert the high correlation coefficients values from the plain image into very little correlation coefficients between adjacent pixels in the cipher image.

IV. CONCLUSION AND FUTURE WORK

In this paper, we designed and tested an efficient image encryption and authentication scheme based on chaotic sequences. The proposed cryptosystem uses the high sensitivity features of the chaotic systems for initial values by producing the secret key of the chaotic generator from the secret hash key and the plain image. This mechanism permits to achieve the required diffusion and confusion effects. Simulation results and performance analysis show that our proposed cryptosystem at least is ten times faster than AES algorithm and also better than most of the chaos based cryptosystems of the literature, for both security level and time consuming. Our future work will focus on designing and testing chaos-based hash functions.

REFERENCES

[1] A. Akhshan, A. Akhavan, S.C. Lim, and Z. Hassan, "An Image Encryption Scheme Based on Quantum Logistic Map," Communications in Nonlinear Science and Numerical Simulation, vol. 17, no. 12, 2012, pp. 4653-4661.

[2] S. El Assad, "Chaos Based Information Hiding and Security," in 7th International Conference for Internet Technology and Secured Transactions, IEEE, London, United Kingdom, 10-12 Dec. 2012, pp. 67- 72. *Invited paper.

[3] M. Chetto, H. Noura, S. El Assad, and M. Farajallah,"How to Guarantee Secured Transactions With QoS and Real-Time Constraints", in 7th International Conference for Internet Technology and Secured Transactions, IEEE, London, United Kingdom, 10-12 Dec. 2012, pp. 40-44.

[4] S. Rakesh, A. Ajitkumar, B, Shadakshari, and B, Annappa, "Image Encryption Using Block Based Uniform Scrambling and Chaotic Logistic Mapping," International Journal on Cryptography and Information Security –IJCIS, vol. 2, no. 1, 2012, pp. 49-57.

[5] F. Chiaraluce, L. Ciccarelli, E. Gambi, P. Pierleoni, and M. Reginelli, "A New Chaotic Algorithm for Video Encryption," IEEE Transactions on Consumer Electronics, vol. 48, no. 4, 2002, pp. 838–844.

[6] G. Chen, Y. Mao, and C.K. Chui, "A symmetric image encryption scheme based on 3D chaotic cat maps," Chaos, Solitons & Fractals, vol. 21, no. 3, pp. 749-761, 2004.

[7] J. Fridrich, "Symmetric Ciphers Based no Two-Dimensional Chaotic Maps," International Journal of Bifurcation and Chaos, vol. 8, no. 6, 1998, pp. 1259-1284.

[8] C.Y. Song, Y.L. Qiao, and X.Z. Zhang, "An Image Encryption Scheme Based on New Spatiotemporal Chaos," Optik, October 2012.

[9] D. Socek, S. Li, S. Magliveras, and B. Furht, "Enhanced 1-D Chaotic Key-Based Algorithm for Image Encryption," in First IEEE International Conference on Security and Privacy for Emerging Areas in Communications Networks, Athens, Greece, 2005, pp. 406 - 413.

[10] S. El Assad (85%), and H. Noura (15%), Generator of Chaotic Sequences and Corresponding Generating System WO Patent WO/2011/121,218, 2011.

[11] Secure Hash Standard, Standard Publication #180-1 (addendum to [1]), 1995, United States Department of Commerce, National Institute of Standards and Technology, Federal Information Processing.

[12] J. Katz and L. Yehuda, Introduction to Modern Cryptography. USA: CRC-Press, 2007.

[13] H. Yang, KW. Wong, X liao, W. Zhang, and P. Wei, "A Fast Image Encryption and Authentication Scheme Based on Chaotic Maps", Communications in Nonlinear Science and Numerical Simulation, vol. 15, no. 11, 2010, pp. 3507-3517.

[14] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," in Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology, Santa Barbara, California, USA, 1990, pp. 2-21.

 ISBN: 978-1-61208-298-1

Propagation of Truncated Differentials in GOST

Nicolas T. Courtois
University College London,
Gower Street, London, UK
n.courtois@cs.ucl.ac.uk

Theodosis Mourouzis
University College London,
Gower Street, London, UK
theodosis.mourouzis.09@ucl.ac.uk

***Abstract*—GOST 28147-89 is a well-known block cipher with 256-bit keys. Its excessively low implementation cost makes it a plausible alternative for major industrial cryptographic algorithms such as 3-DES and AES-256. In 2010, GOST was submitted to ISO to become a part of the international encryption standard ISO/IEC 18033-3. This stimulated intense research by the cryptographic community and lots of new attacks were developed which reduce its 256-bit security level. These recent attacks against full GOST belong to two main categories: complexity reduction attacks and advanced differential attacks. In differential cryptanalysis, the essential task is the exploration of the exponentially large space of differentials in a systematic way and the construction of complex distinguisher attacks. In this paper, we study the GOST cipher in the well-known theory framework of Markov cipher, which is a basis of many works on differential cryptanalysis. However, we prove that GOST is NOT a Markov cipher though in approximation it still seems to behave like one. We propose a heuristic black-box methodology for efficient discovery of interesting sets of differentials in GOST and we show that results better than any previously known can be obtained with this methodology. However, different sets will be the best possible solutions for various numbers of rounds and more work is needed in order to improve the best known single-key attacks on GOST and adapt them to other sets of S-boxes.**

Keywords*—*differential cryptanalysis, block ciphers, GOST, S-boxes, diffusion, optimization problems, truncated differentials, aggregated differentials

I. INTRODUCTION

GOST 28147-89 encryption algorithm is the state standard of Russian Federation and it is widely used for encrypting confidential documents. It is implemented in many crypto libraries such as OpenSSL and Crypto++ [15], [19] and is one of the Internet data security standards. In 1989, it was standardized and became an official standard for protection of confidential information. The specification of the cipher was kept secret until 1994 when it was declassified and published [24]. The first international translation was done in 1994 by Malchik and Diffie [18].

Until 2010, most researchers would agree that despite considerable cryptanalytic efforts spent in the past 20 years, GOST is still not broken. The very large 256-bit security level of GOST and its excessively competitive low implementation cost made it a plausible alternative to all major standard cryptographic algorithms such as 3-DES or AES [19]. Accordingly, in 2010 it was submitted to ISO 18033-3 to become a worldwide industrial standard. This has stimulated intense research and lead to the development of many interesting new cryptanalytic attacks.

In general, all these attacks fall in two main categories: differential attacks [7], [11], [12], [13] and complexity reduction attacks [6], [10], [14], where an attacker reduces the problem of attacking the full GOST to a simpler problem of attacking a smaller number of rounds. We have reflection attacks, attacks with double reflections, self-similarity attacks and advanced differential attacks and combinations of these attacks. The main aim of a differential attack is to distinguish a certain number of rounds of GOST from a random permutation on 64 bits and then some key bits can be recovered by following some extra steps. The construction of such distinguishers can be seen as a series of optimization problems which need to be solved for each variant of GOST. Additionally, the exponentially large space of differentials makes the systematic search computationally infeasible and thus some hidden combinatorial structure of the cipher needs to be explored. Courtois and Misztal developed an advanced differential attack with complexity which was later improved to 2^{179} against the full 32-round 256-bit GOST. This attack is based on constructing distinguishers for 20 rounds and by solving a series of combinatorial optimization problems [7], [12].

This paper is structured as follows. In Section II we briefly describe the specifications of the GOST block cipher and its variants. In Section III we study GOST with respect to the well-known notion of Markov cipher. Informally, a Markov cipher is a cipher where the probability of a propagation of a specific difference does not depend on the input plaintexts and does depend only on the XOR of the plaintexts. We prove that GOST is NOT a Markov cipher and try to see how much this cipher deviates from this ideal cipher notion. Early results suggest that it still behaves as a Markov cipher from the practical point of view.

In Section IV, we define a form of set differential cryptanalysis on GOST cipher by introducing some special sets constructed based on the internal connections between S-boxes from round to round.

Finally in Section V, we describe a heuristic methodology for finding sets of differentials whose propagation deviates from what expected in case of a random permutation on 64-bits. Such sets of differentials can later be used to build distinguishers for a larger number of rounds, cf. [7], [9]. More importantly we provide a precise analysis, important insights and theory, on the propagation of interesting differentials in GOST, both from the point of view of advanced differential attacks and combined differential-algebraic attack approaches. This can be exploited further for developing many advanced differential attacks against full cipher and for improving numerous already known attacks in these two families [10].

 ISBN: 978-1-61208-298-1

II. GOST BLOCK CIPHER

GOST is a block cipher with a simple 32-round Feistel structure which encrypts a 64-bit block using a 256-bit key, as shown in *Figure* 1.

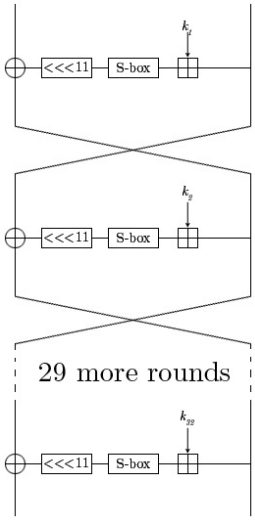


Fig. 1. Diagram of GOST cipher, 32-rounds of a Feistel network to encrypt a 64-bit plaintext using a 256-bit key

Each round of GOST contains and combines a series of logical and arithmetic operations as shown in *Figure* 2. Initially, we have a key addition modulo 2^{32}, then we have a substitution function which consists of 8 different 4-bit to 4-bit S-boxes and the output is then rotated by 11 positions to the left.

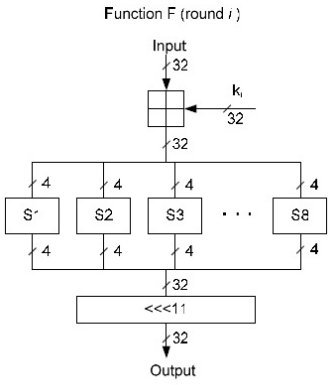


Fig. 2. Detailed description of the round function F_I used in GOST

Thus the image of any input $P = L||R$ after a single round of GOST, where L, R the left and right 32-bit halves respectively, is given by

$$(L, R) \rightarrow (R, L \oplus F_i(R)) \quad (1)$$

GOST block cipher consists of three main components; the key schedule, the S-boxes and the internal connections between them. We briefly discuss them in the next subsections.

A. Key Schedule

The 256-bits of key K are divided into eight consecutive 32-bit words $k_0, k_1, .., k_7$. The first 24 rounds use the keys in this order and only the last 8 rounds use them in the reverse order, as shown in *Table* I.

TABLE I. KEY SCHEDULE IN GOST.

R1-R8	R9-R16
$k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7$	$k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7$
R17-R24	R25-R32
$k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7$	$k_7, k_6, k_5, k_4, k_3, k_2, k_1, k_0$

Its very simple key schedule makes it suitable for crypt-analysis.

B. S-boxes

Each round function makes use of 8 4-bit to 4-bit S-boxes. According to the Russian standard, these S-boxes can be kept secret. Thus, the effective key size is increased to 610 by the addition of this extra $354 * (log_2(16!^8))$ bits of information. However, this information can be recovered in approximately 2^{32} encryptions by a chosen-key attack [21].

C. Internal Connections

Let S_i for $i = 1, 2, ..., 8$ be the i-th S-box used in each round as shown in *Figure* 3. Then we can number the inputs of the S-box S_i by integers from $4i+1$ to $4i+4$ out of $1, .., 32$ and its outputs are numbered according to their final positions after the rotation by 11 positions.

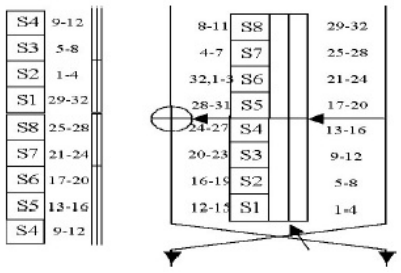


Fig. 3. The connections between different S-boxes from round to round inside GOST

For example the inputs of $S6$ are 20,21,22,23 and the outputs are 32,1,2,3. Such connections are of major cryptanalytic importance as they describe how these S-boxes interact within the general structure of the cipher.

III. MARKOV CIPHER

The concept of Markov block cipher is a very important theory concept in the development and study of differential and linear cryptanalysis [1], [16], [17].

Informally, in a Markov block cipher the average difference propagation probability over each round is independent of the rounds's text input. There are numerous examples of Markov cipher including DES, Rijndael, Camelia and many others.

In spite of the progress in mathematical foundations of differential and linear cryptanalysis, there are still difficulties in

analyzing and obtaining security proofs for non-Markov cipher, like GOST due to lack of adequate mathematical methods accounting for the structure and irregular properties of non-Markov cipher.

Definition 1: An iterated cipher round function $Y = f(X, Z)$ is a Markov cipher if there is a group operation $\otimes$ for defining differences such that, for all choices of α $(\alpha \neq e)$ and $(\beta \neq e)$, $P(\Delta Y = \beta | \Delta X = \alpha, X = \gamma)$ is independent of γ when the subkey Z is uniformly random.

If an iterated cipher is Markov and its round subkeys are independent then the sequence of differences at each round output forms a Markov chain.

Definition 2: We say that X may cause Y with probability p by the function F if for a fraction p of all the possible input pairs encrypted by all the possible subkeys values in which the input XOR of the F equals to X, the output XOR equals Y. If $p > 0$ we denote this by $X \rightarrow Y$.

In the rest of this section we will study GOST with respect to its Markov and non-Markov properties. We consider differences with respect to the bitwise XOR operation.

Let $X^{(i)} = L^{(i)}||R^{(i)}$ for $i = 1, 2$ be two distinct plaintexts.

Then the round function G maps $X^{(i)}$ to

$Y^{(i)} = R^{(i)}||(L^{(i)} \oplus ROT^{11}S(L^{(i)} \boxplus R^{(i)}))$

Xoring the two outputs we have

$\Delta Y = Y^{(1)} \oplus Y^{(2)}$

$= (R^{(1)} \oplus R^{(2)})||$

$(L^{(1)} \oplus L^{(2)} \oplus ROT^{11}(S(K \boxplus R^{(1)}) \oplus S(K \boxplus R^{(2)})))$

$= (\Delta Y^R)||$

$(\Delta Y^L \oplus ROT^{11}(S(K \boxplus R^{(1)}) \oplus S(K \boxplus R^{(2)})))$

As we prove below GOST is not a Markov cipher but it can be a Markov cipher if the modular 2^{32} addition $\boxplus$ is replaced by the bitwise XOR operation $\oplus$.

Theorem 1: GOST is a Markov cipher if the modular 2^{32} addition $\boxplus$ is replaced by the bitwise XOR operation $\oplus$

Proof: To prove this lemma it is suffices to prove the property for each of the S-boxes.

For each input XOR $S'_E = S_E \oplus {S^*}_E$ there exist $S'_I = S'_E$ regardless of the key since $(S_E \oplus K) \oplus (({S^*}_E \oplus K))$ is independent of key.

Suppose that there exist exactly k pairs $\{(S^i_I, {S^*}^i_I)\}_{1 \leq i \leq k}$ to the S-boxes such that $S_E \oplus {S^*}_E = X$ and the output XOR is Y, cf. 4

Fix a pair input $(P_E, {P^*}_E)$ such that $P_E \oplus {P^*}_E = X$. There exists unique K for each i such that $P_E \oplus K = S^i_I$ which forces ${P^*}_E \oplus K = {S^*}^i_I$. However, $X = P_E \oplus {P^*}_E = (S^i_I \oplus K) \oplus ({S^*}^i_I \oplus K) = S^i_I \oplus {S^*}^i_I$ is not affected by key addition and comes for free.

Thus we have that the XOR output will be Y for a fix pair of inputs for exactly k keys and thus this version of GOST with XOR bitwise operation instead of modular addition is a Markov cipher.

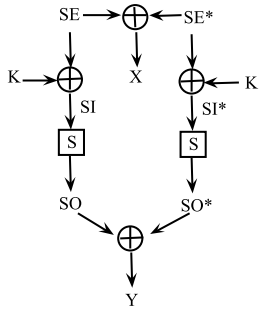


Fig. 4. The output given two input pairs for modified GOST

Theorem 2: GOST is **NOT** a Markov cipher

Proof: Suppose that there exist exactly k pairs $\{(S^i_I, {S^*}^i_I)\}_{1 \leq i \leq k}$ to the S-boxes such that $S_E \oplus {S^*}_E = X$ and the output XOR is Y, cf. 5.

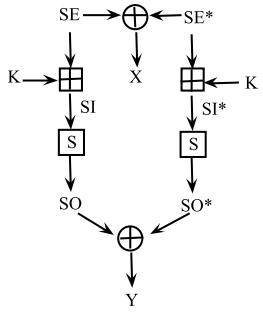


Fig. 5. The output given two input pairs for actual GOST

We follow the same methodology as before and we fix a pair input $(P_E, {P^*}_E)$ such that $P_E \oplus {P^*}_E = X$. Let l be the number of keys that follow the input and output difference property.

There exists unique K for each i such that $P_E \boxplus K = S^i_I$. However, now ${P^*}_E \boxplus K = {S^*}^i_I$ is not ensured by the same value of K as in general $(\alpha \oplus \beta) \boxplus \gamma \neq \alpha \oplus (\beta \boxplus \gamma)$.

We have $l = m$ is not always true and thus GOST is **NOT** a Markov cipher.

Remark - Another Proof: There is another way to disprove this result, it is by concrete analysis of concrete attacks. We sketch one such proof by conter-example on a very specific example which is dictated by our pragmatic cryptanalysis work. We have computed by simulation the probability $P(\Delta Y \in 0x8070070080700700 | \Delta Y \in 8070070080700700)$ over randomly selected keys and plaintexts and it is approximately equal to $2^{-3.73}$. Additionally, we computed some probabilities $P(\Delta Y \in 0x8070070080700700 | \Delta Y \in 8070070080700700, X = X_0)$ for fixed plaintext X_0 and we observed that they are not equal to $2^{-3.73}$ proving also experimentally that GOST is not a Markov cipher. For example for $X_0 = 0x000031A90F4A3312$ the probability equals to $2^{-3.61}$ and for $X_0 = 0x0000001000000010$ equals to $2^{-3.89}$. However, the difference between these probabilities is approximately equal to $|2^{-3.61} - 2^{-3.89}| \simeq 0.014$ which shows that GOST still is a somehow Markov cipher with a margin of

error which is only about 10 %. Moreover, we know from our Theorem 1 and 2 that this is due to the presence of the modular addition mod 2^{32}. Overall, we still conjecture that techniques applied to normal Markov cipher can be applied to GOST and lead to approximate but essentially valid cryptographic results.

IV. Advanced Differential Cryptanalysis

Differential cryptanalysis (DC) is one of the oldest known attacks on block ciphers. Biham and Shamir were the first to describe this method and applied it to DES algorithm, see [3], [4]. DC is based on tracking of changes in the differences between two messages as they pass through the consecutive rounds of encryption.

We apply an advanced form of differential cryptanalysis to GOST. We consider differences with respect to the bitwise XOR operation (and we still do apply them to the standard GOST cipher which also uses modular additions). We define an aggregated differential A, B as the transition where any non-zero difference $a \in A$ will produce an arbitrary non-zero difference $b \in B$ with a certain probability.

We need to experimentally determine the probabilities of these transitions from A to B for different number of rounds. This is achieved by carrying out lots of simulations and aggregating the results obtained in each simulation until the probability value converges to its actual value. According to the Central Limit Theorem because our experience is repeated many times, the average number of suitable events observed by the attacker is approximated by the Gaussian with reasonable precision. As the number of trials is increased, it is going to become closer to the expected value and the deviation can be predicted according to the Gauss Error Function.

We consider the following differential set: $\Delta = 0x80700700$ by which we mean all differences with between 1 and 7 active bits (but not 0) and where the active bits are contained within the mask $0x80700700$. Similarly, an aggregated differential (Δ, Δ) means that we have 14 active bits, and that any non-zero difference is allowed. There are $2^{14} - 1$ differences in this set of ours. The following fact can be verified experimentally for the version of GOST which uses GostR3411-94-TestParamSet set of S-boxes:

Experimental Facts: The aggregated differential (Δ, Δ) with uniform sampling of all differences it allows, produces an element of the same aggregated differential set (Δ, Δ) after:

1) 1 round of GOST with probability $P_1 = 2^{-3.73}$ on average over all possible keys
2) 4 rounds of GOST with probability $P_4 = 2^{-13.6}$ on average over all possible keys
3) 8 rounds of GOST with probability $P_8 = 2^{-25.0}$ on average over all possible keys

We partition the space of differences (Δ, Δ) into 63 disjoint classes. All these sets of differentials are constructed based on the connections between the S-boxes so they need to be re-invented for each new variant of GOST. In the next chapter, we explain our heuristic methodology for finding such differentials.

V. Direct Black-Box Discovery Method for Truncated Differential Attacks on GOST

In classical differential cryptanalysis, such as Biham-Shamir attacks on DES, [3], [4], the process of discovery of best differential attacks is rather straightforward. In advanced differential attacks however, the potential number of different sets of differences which could be mapped to other arbitrary differences makes systematic exploration impossible. Some heuristics are needed which basically group together similar differentials into sets of differentials of certain type, cf. [9].

In our work, we are basically looking for differential sets which are very similar to those already known from [11], [12], [22]. However, we do not want to follow the heuristics from [7] such as looking at "loops" of S-boxes which are connected to each other, because we believe that better attacks exist which do not exhibit this sort of regular structure. Ultimately what matters are the best differential properties we can find.

Our heuristic pseudo-code for finding new interesting attacks on GOST is as follows:

1) We select at random set of say 14 bits. This seems to be about right size, previous attacks have used sets of 14-24 bits [7], [11], [22].
2) We work in black-box way for a fixed number of rounds for example between 4 and 8.
3) For each set we run a simulation of whether flipping some random difference within the set of 14 active bits results in output difference in which the bits which differ are also a subset of these 14 bits.
   In other words we are studying an invariant truncated differential property for e.g. 8 rounds of GOST.
4) We use a variant of method called method of "Structures" by Biham and Shamir [3]. More precisely we fix the 64-14 bits and consider a fraction but not all possible plaintexts with such fixed 64-14 bits, and look at how many ciphertext also share the same 64-14 bits. This method gives a quadratic speedup: the number of pairs with suitable difference we see here is the square of the number of actual encryptions which we need to carry.
5) Thus we can measure the probability with sufficient precision to see which sets are within heuristically 2^{-5} form the best know set of 14 bits.
6) We keep a database of 100 best sets of 14 bits at any moment. The sets are ranked based on the observed propagation probability.
7) During the attack we mix fully random sets of 14 bits with sets obtained by flipping up to 4 bits in the 100 already found sets.
8) In this way we progressively update our set of 100 best results.
9) At the end we stop and run a much longer simulation to see which out of 100 is really the best, because at no moment during the above discovery process we know these probabilities with sufficient precision.

For the time being we have obtained the following quite remarkable and surprising results.

 ISBN: 978-1-61208-298-1

TABLE II. SOME SETS OF 14 BITS WITH PROPAGATION BELOW 2^{-25} FOR 8 ROUNDS

Set Name	Set	P(8R)
GostR3411_94_Test	78000078 07070780	$2^{-24.0}$
GostR3411_94_CryptoPro	00030780 703A0010	$2^{-22.8}$
Gost28147_Test	60707800 00000507	$2^{-22.2}$
Gost28147_CryptoPro A	70780000 80030780	$2^{-23.8}$
Gost28147_CryptoPro B	C0707000 00000707	$2^{-23.0}$
Gost28147_CryptoPro C	03070780 78000010	$2^{-25.3}$
Gost28147_CryptoPro D	70707000 80000207	$2^{-25.0}$
GostR3411_94_Sberbank	80080207 80707800	$2^{-22.4}$
ISO 18033-3 proposal	80000707 20707000	$2^{-22.7}$
GOST-P proposal	50703800 00000707	$2^{-25.2}$

A. Early Results With Our Simple Algorithm

First of all, it is possible to see that on the strict basis of results for 8 rounds, one can find better results than currently known not only for the default sets of S-boxes, but for absolutely every other known set of S-boxes. Our results are shown in Table II.

This is quite remarkable because it shows that other sets of S-boxes are maybe not stronger. Unhappily, the best results for 8 rounds are not necessarily the best results for other numbers of rounds. We give one detailed below. The aggregated differential $(0x78000078, 0x07070780)$ with uniform sampling of all input differences, produces an element of the same aggregated differential set with 4 or 8 rounds with the following probabilities on average over all possible keys:

1) For 4 rounds of GOST with probability $P_4 = 2^{-13.8}$. This is NOT as good as $P_4 = 2^{-13.6}$ for the previous set (Δ, Δ), however for 8 rounds it will be otherwise.
2) After 8 rounds of GOST we obtain probability $P_8 = 2^{-24.0}$ on average over all possible keys which is strictly better than $P_8 = 2^{-25.0}$ for the previous set (Δ, Δ).
3) This however does NOT mean that it will be better for 10 rounds. In fact for 10 rounds we obtain less than 2^{-35} which is not as good as $P_{10} = 2^{-31.0}$ with the previous set (Δ, Δ). see [12].

We see that discovery of interesting iterative invariant attacks on 8 rounds of GOST cannot rely on heuristic combination of 8=4+4 rounds, and that our new result is not very good for 4 rounds yet now becomes the best ever found for 8 rounds which however does NOT guarantee it is the best for 10 rounds. This justifies our black-box methodology but also shows that it is difficult to find a solution which works for various numbers of rounds.

B. Propagation of New Sets

It is interesting to see if for new sets we can observe the same sort of behavior as before, which amounts to saying that few paths in a certain transition graph dominate the whole attack for many rounds, which will be the best currently known attack. For the new property we have made the following observations which show that in many aspects the new attacks are similar to the old ones yet more irregular. Moreover, the entropy of states deeply inside the cipher is quite low. This suggests that in all cases design further advanced combined attacks such as differential-algebraic attacks [2] or attacks with simultaneous differentials cf. [8]. Below we show what we call the "dominating path" in the transition graph for our new discovery set of 14 bits which shows that special cases may happen quite frequently.

$$0R : 7000007007070000$$
$$1R : 0707000000000070$$
$$2R : 0000007007000000$$
$$3R : 0700000000000000$$
$$4R : 0000000007000000$$
$$5R : 0700000000000070$$
$$6R : 0000007007070000$$
$$7R : 0707000070000000$$
$$8R : 7000000007070780$$

In one propagation for 8 rounds (and similarly for any number of rounds) we have observed that the specification of input and output sets being $(0x78000078, 0x07070780)$ as above, with uniform sampling of all input differences, we have observed that for about 2^{-2} of the time, the above sets of differential sets are simultaneously satisfied at every round. Moreover, only few such paths amount for a majority of all events which are observed.

VI. CONCLUSION

GOST is an important government and industrial block cipher which is widely used and implemented in standard crypto libraries such as OpenSSL, Crypto++ and also in RSA Security products. GOST is a 32-round Feistel cipher with 64-bit blocks and 256-bit key. Until 2010, there were no serious attacks on full GOST which may threaten its 256-bit level security and thus it was submitted to ISO for standardization. This stimulated the cryptographic community to carry out a more extensive security analysis of GOST and as a result of this many new attacks were developed.

Most differential attacks of GOST are based on the construction of a distinguisher for a reduced version of GOST from a random permutation on 64-bits. This is combined with many additional complex technical steps to recover the full key. However, the construction of the initial distinguisher is difficult to achieve. It is based on the exploration of the exponentially large space of differentials for finding interesting patterns which propagate for a large number of rounds, and on combinations of such properties, see also [9].

In this paper, we study the fundamental question of how such differential attacks propagate inside the cipher. This question is fundamental both in order to enable efficient heuristic discovery of similar attacks, and in order to improve existing distinguisher attacks by more precise statistical analysis or by combination [9]. In order to study this question, one first needs to answer the question whether GOST is a Markov cipher. This question is about whether (at least on average over the keys) the differential transitions can be seen as events which are independent on the input value, and happen with more

 ISBN: 978-1-61208-298-1

or less stable probabilities. We have answered this question by negative: we prove mathematically that GOST is NOT a Markov cipher. However, from the practical point what matters is how much these probabilities will vary, and not for individual differentials, but really for the most interesting sets of differentials such as used in the best known differential attacks on GOST [7]. To this more pragmatic question our current answer is positive: it seems that GOST behaves like a Markov cipher in practice with deviations of about 10 $\%$ which question however deserves a further study.

The next important question is can we find better attacks on GOST? In this respect, in this paper we have not followed some regularly shaped sets from previous papers and we have introduced a new simple heuristic methodology for finding differentials of more irregular shape efficiently. We have applied this methodology to the main historical version of GOST and have found properties which are substantially stronger than in previous works, event though it remains very difficult to find differential sets which work equally well for various numbers or rounds, as a result of which we claim that almost certainly one can improve the best currently known single key attack on GOST in 2^{179} from [7] though the exact further adaptation of the attack to any new set requires a lot of attention to detail, see [7], [9]. However, we are now able to improve the central property which allows to construct such attacks. More such results will appear in the extended version of this paper.

Moreover, we have discovered that the internal propagation inside GOST cipher for properties we studied is quite remarkable. All cases we have studied in details lead to quite strong events inside the cipher which basically amounts to paths in a graph with very few dominant paths accounting for a larger proportion of all interesting events. This leads to surprisingly low entropy of differences inside the cipher which fact is already explicitly exploited in several differential-complexity reduction-algebraic attacks in [10] and is expected to lead to many interesting advanced differential-algebraic [2] and simultaneous differential attacks [8]. This sort of low entropy facts are already exploited in several attacks on GOST in [10].

REFERENCES

[1] Martin R. Albrecht and Gregor Leander: *A*n All-In-One Approach to Differential Cryptanalysis for Small Block Ciphers, preprint available at eprint.iacr.org/2012/401/.

[2] Martin R. Albrecht, Carlos Cid: *Algebraic Techniques in Differential Cryptanalysis*, In FSE 2009, LNCS, pp.193-208, Springer, 2009.

[3] E. Biham and A. Shamir: *Differential Cryptanalysis of the Full 16-round DES* In: Crypto'92, Springer-Verlag, 487, 1992.

[4] E. Biham and A. Shamir: *Differential Cryptanalysis of DES-like Cryptosystems* Extended Abstract. In: Crypto'90,Springer-Verlag, 2., 1990.

[5] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann and M.J.B. Robshaw: *PRESENT: An Ultra-Lightweight Block Cipher* CHES 2007, LNCS 4727, pp. 450466, Springer, 2007.

[6] N.T. Courtois: *Security Evaluation of GOST 28147-89 In View Of International Standardisation.* In Cryptologia, Volume 36, Issue 1, pp. 2-13, 2012.

[7] N.T. Courtois: *An Improved Differential Attack on Full GOST.* In Cryptology ePrint Archive, Report 2012/138. 15 March 2012, http://eprint.iacr.org/2012/, 2012.

[8] Nicolas T. Courtois: *T*he Dark Side of Security by Obscurity and Cloning MiFare Classic Rail and Building Passes Anywhere, Anytime, In SECRYPT 2009 International Conference on Security and Cryptography: pp. 331-338. INSTICC Press 2009.

[9] Nicolas T. Courtois, Theodosis Mourouzis: *E*nhanced Truncated Differential Cryptanalysis of GOST, in SECRYPT 2013, 10th International Conference on Security and Cryptography Reykjavik, Iceland, July 29-31, 2013

[10] N.T. Courtois: *Algebraic Complexity Reduction and Cryptanalysis of GOST.* In Cryptology ePrint Archive, Report 2011/626, 2011.

[11] Nicolas Courtois, Michał Misztal: *A*ggregated Differentials and Cryptanalysis of PP-1 and GOST, In CECC 2011, Periodica Mathematica Hungarica Vol. 65 (2), pp. 1126, 2012.

[12] N.T. Courtois and M. Misztal: *First Differential Attack On Full 32-Round GOST.* In ICICS'11, pp. 216-227, Springer LNCS 7043, 2011.

[13] N.T. Courtois and M. Misztal: *Differential Cryptanalysis of GOST.* In Cryptology ePrint Archive, Report 2011/312, 2011.

[14] I. Dinur, O. Dunkelman and A. Shamir:*Improved Attacks on Full GOST*. FSE 2012, LNCS 7549, pp. 9-28, 2012.

[15] GOST: *A Russian reference implementation of GOST implementing Russian algorithms as an extension of TLS v1.0. is available as a part of OpenSSL library. The file gost89.c contains eight different sets of S-boxes and is found in OpenSSL 0.9.8 and later: http://www.openssl.org/source/.* , 2005.

[16] L.R. Knudsen: Block Ciphers The Basics, Spring 2011, https://www.cosic.esat.kuleuven.be/ecrypt/courses/albena11/slides/LRK-basics.pdf.

[17] X. Lai and J. Massey:*Markov Ciphers and Differential Cryptanalysis*. In Eurocrypt'91, LNCS 547, Springer-Verlag, pp.17-38, 1991.

[18] A. Malchik and W. Diffie:*English Translation,Cryptographic Protection for Information Processing Systems, Government Standard of the USSR,GOST 28147-89.* autochthonous.org/crypto/gosthash.tar.gz, 1994.

[19] A. Poschmann, S. Ling and H. Wang: *256 Bit Standardized Crypto for 650 GE GOST Revisited.* In CHES 2010, LNCS 6225, pp. 219-233, 2010.

[20] V. Rudskoy and A. Chmora: *Working draft for ISO/IEC 1st wd of amd1/18033-3.* In Russian Block Cipher GOST, ISO/IEC JTC 1/SC 27 N9423, MD5=feb236fe6d3a79a02ad666edfe7039aa , 2011.

[21] M. J. Saarinen: *A chosen key attack against the secret S-boxes of GOST*. Unpublished manuscript, 1998.

[22] H. Seki and T. Kaneko: *D*ifferential Cryptanalysis of Reduced Rounds of GOST. In SAC 2000, LNCS 2012, pp. 315-323, Springer, 2000.

[23] B. Schneier: *Applied Cryptography*. Section 14.1 GOST, 2nd Edition, In John Wiley and Sons,1996.

[24] I.A. Zabotin, G.P. Glazkov and V.B. Isaeva: *Cryptographic Protection for Information Processing Systems, Government Standard of the USSR,GOST 28147-89.* Government Committee of the USSR for Standards, 1989.

 ISBN: 978-1-61208-298-1

PP-2 Block Cipher

Krzysztof Bucholc, Krzysztof Chmiel, Anna Grocholewska-Czurylo, Janusz Stoklosa
Institute of Control and Information Engineering
Poznan University of Technology
Poznan, Poland
{krzysztof.bucholc, krzysztof.chmiel, anna.grocholewska-czurylo, janusz.stoklosa}@put.poznan.pl

***Abstract*—The paper describes the rationale behind PP-2 block cipher and gives the details of the original design of this cipher. PP-2 block cipher is the result of continued development of the PP-1 block cipher, improving on performance, resistance against differential cryptanalysis and the speed of diffusion. Cipher structure is different for encryption and for decryption. Diffusion layer is based on multiple rotations. This paper shows that for block length n = 64 bits a global diffusion is reached after two rounds. Number of rounds is dependent on key size.**

Keywords - block ciphers; S-boxes; differential cryptanalysis; diffusion

I. INTRODUCTION

PP-2 cipher emerged as a continuation of PP-1 cipher development carried out by the Institute of Control and Information Engineering at the Technical University of Poznan. Initially, a 64-bit block length cipher PP-1 was designed, which was aimed at limited resource platforms, especially with very limited internal storage available for storing ciphers components [4]. Next, a scalable version of PP-1 cipher was developed, with block length being a multiple of 64 bits [3][6][8]. A distinctive feature of this cipher is that it constitutes an involutional substitution-permutation network. The same network, and particularly the same single S-box and the same single P-block, are used for both encryption and decryption process. PP-1 cipher is characterized by high resistance to differential and linear cryptanalysis and high performance [5][7].

Motivation for PP-2 block cipher was to develop a cipher which inherits advantages of PP-1 cipher, namely high resistance to cryptanalysis, while improving on performance. Limited resource platforms was no longer a target system for PP-2, as it was for PP-1. One of the aims for both PP-1 and PP-2 ciphers was an efficient implementation in both hardware and software.

Comparing PP-2 cipher to DES, PP-2 is characterized by higher resistance to linear and differential cryptanalysis, its software implementation is faster and it is scalable (variable block and key length). In comparison to Advanced Encryption Standard (AES) [1], PP-2 has roughly the same resistance to linear and differential cryptanalysis, similar software implementation efficiency and is scalable.

Main differences between PP-2 and PP-1 are that PP-2 is no longer involutional (to increase cipher efficiency dictated by higher number of rounds needed to resists Misztal's attack), an affine transform has been added to S-box construction, P-box based on multiple rotations further increases efficiency and the speed of diffusion, reducing the number of round keys and modifications to key generation algorithm increases efficiency even further.

In Section II, a general structure of PP-2 cipher is presented, which is an enhanced version of PP-1 cipher that further improves the performance, increases resistance to differential cryptanalysis [10] and speeds up the diffusion.

Section III describes the S-box designed for PP-2 cipher and gives the details of the method used to generate it.

Section IV describes the round key generation scheme. An important feature of round key generation algorithm is the independence of generated bit sequences forming the round keys, which improves cipher's security, as the cryptanalysis is more difficult for independent keys [2].

As for security considerations, in Section V, upper bounds of effectiveness of the PP-2 nonzero linear and differential approximations are given. Further and more detailed PP-2 security evaluation is planned.

Section VI gives the results of PP-2 cipher speed tests. Two compilers were tested: C++ compiler from MS Visual Studio 2008 package and Intel C++ Compiler Professional 11.1 for Windows. Application extensively utilizes 64-bits operations. For this reason both 32- and 64-bit code was generated for each compiler. Speed of cipher operation is roughly 50% higher for 64-bit version as compared to 32-bit version generated by Intel compiler. For VS2008 compiler, this difference was not that high, at roughly 33%.

II. GENERAL STRUCTURE OF PP-2

A. One round structure

PP-2 cipher is a symmetric scalable block cipher that processes data blocks of n bits in r rounds with the key k of the bit length $|k|$, such that $|k| = d \cdot n$, where $n = t \cdot 64$ and $d, t \in \{1, 2, 3, ...\}$. (Tab. 1).

1) Construction of the round

One round of the cipher is presented in Fig. 1. It is composed of $t = n/64$ parallel processing paths. In each path, a 64-bit nonlinear operation NL is executed. Furthermore, an n-bit permutation P is carried out. In each round, a round key k_i is used.

 ISBN: 978-1-61208-298-1

TABLE I. NUMBER OF ROUNDS DEPENDING ON THE BLOCK SIZE

Key length |*k*| \ Block size *n*	64	128	192	256	...
64	11				
128	13	22			...
192	15	24	32		...
256	17	26	34	43	...

The nonlinear element *NL* is presented in Fig. 2. A 64-bit subblock is processed as eight 8-bit subblocks by four types of operations: S-boxes *S* of the size 8×8, XOR (⊕) of respective bits, addition (⊞) modulo 256 and subtraction (⊟) modulo 256 of integers represented by respective bytes. Two of these operations (addition and subtraction modulo 256) are local nonlinear mappings.

2) The layer structure of a round

In round *i*, the following 3 layers can be distinguished: the key addition layer, the substitution layer and the permutation layer. Functions of the layers will be denoted by *KL*, *SL* and *P*, respectively.

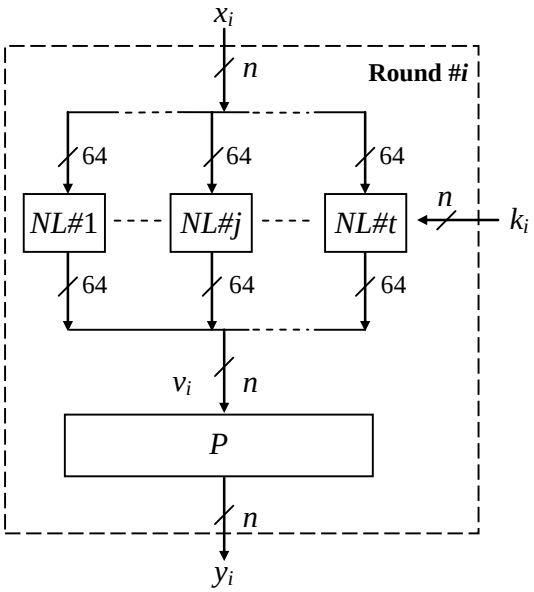


Figure 1. Processing in round #*i* (*i* = 1, 2, ..., *r*)

In Fig. 1, one round of the PP-2 cipher during encryption and decryption is presented. For decryption the order of the layers is inverted, and the appropriate (corresponding) inverse functions are used, i.e., P^{-1}, SL^{-1} and KL^{-1}.

During encryption, the round function *h* is the following composition (product) of the layer functions:

$$h = P \circ SL \circ KL. \quad (1)$$

For decryption, the inverse round function h^{-1} is calculated as follows:

$$h^{-1} = KL^{-1} \circ SL^{-1} \circ P^{-1}. \quad (2)$$

In layer *KL* of elements *NL* (Fig. 2), the following sequence of 8-bit operations is used: (⊞, ⊕, ⊟, ⊕, ⊕, ⊟, ⊕, ⊞), and in layer KL^{-1} – the sequence of their inverse operations, *i.e.* (⊟, ⊕, ⊞, ⊕, ⊕, ⊞, ⊕, ⊟). In layer *SL*, substitution *S* is used, and in layer SL^{-1}, substitution S^{-1}.

3) The cipher structure

In the PP-2 cipher, a different structure for encryption and decryption is used. The function *h* is used in each round *i* = 1, 2,…, *r*–1, while in the round number *r* (the output round) the function $\hat{h} = SL \circ KL$ is different – it is composed of *KL* and *SL*. Thus, the transformation for the encryption is as follows:

$$PP\text{-}2 = \hat{h} \circ h^{r-1} = (SL \circ KL) \circ (P \circ SL \circ KL)^{r-1} \quad (3)$$

Functions h^{-1} in rounds no. *i* = 2, 3,…, *r*, are the same, and function $\hat{h}^{-1} = KL^{-1} \circ SL^{-1}$ of round no. 1 (the input round) does not contain the permutation P^{-1}. The transformation for the decryption has, therefore, the following form:

$$PP\text{-}2^{-1} = (h^{-1})^{r-1} \circ \hat{h}^{-1} = (KL^{-1} \circ SL^{-1} \circ P^{-1})^{r-1} \circ (KL^{-1} \circ SL^{-1}) \quad (4)$$

.

Therefore, in the PP-2 cipher we have:

$$PP\text{-}2 \neq PP\text{-}2^{-1}, \quad (5)$$

and in consequence, a different network is used during encryption and decryption, and the round keys are applied in reverse order.

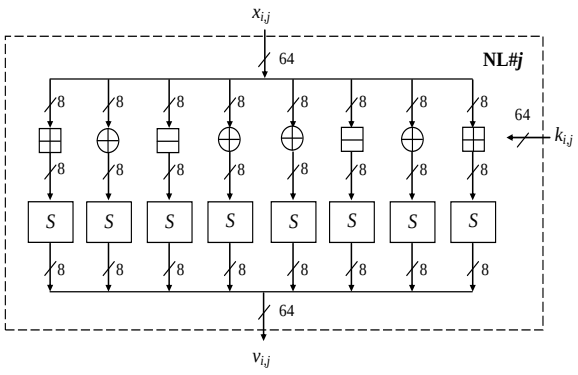


Figure 2. Nonlinear element *NL#j* of PP-2 (*j* = 1, 2, ..., *t*)

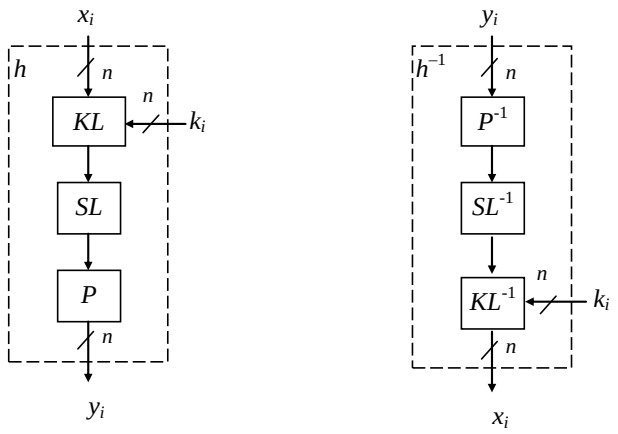


Figure 3. Round function h (encryption) and its inverse function h^{-1} (decryption)

B. Diffusion layer

1) Construction of multiple rotations

Assume that the bits of the input block and the output block are numbered successively, from 1 to n, from the left to the right. Consider permutation P of the PP-2 cipher as the transformation of bit numbers, i.e., as bijection P: {1, 2, ..., n} → {1, 2, ..., n}.

We define the rotation of bit i by b bits to the right as the following mapping:

$$ROR(b, i) = (i + b - 1) \bmod n + 1, \text{ for } i \in \{1, 2, ..., n\}. \quad (6)$$

For the set of indices, $I \subseteq \{1, 2, ..., n\}$ we define a rotation to the right as follows:

$$ROR(b, I) = \begin{cases} ROR(b, i) & \text{for } i \in I \\ 0 & \text{for } i \notin I. \end{cases} \quad (7)$$

For the PP-2 cipher with block length n = 64, the transformation of bit i with use of permutation P, called the multiple rotation to the right, is defined as follows:

$$P(i) = ROR(12, [1]) + ROR(28, [2]) + ROR(44, [3]) + ROR(60, [4]), \quad (8)$$

where $i \in \{1, 2, ..., n\}$ and:

$$[1] = \{1, 5, ..., 61\}, [2] = \{2, 6, ..., 62\}, [3] = \{3, 7, ..., 63\}, [4] = \{4, 8, ..., 64\}. \quad (9)$$

The sets defined by (9) are called the classes of bits. In the general case, *i.e.*, for $n = t \cdot 64$ (t = 1, 2, ...), the permutation P of the PP-2 cipher is defined identically (i.e., by Formula 8), and the classes of bits are defined as follows:

$$[1] = \{1, 5, ..., t\cdot64–3\}, [2] = \{2, 6, ..., t\cdot64–2\}, [3] = \{3, 7, ..., t\cdot64–1\}, [4] = \{4, 8, ..., t\cdot64–0\}. \quad (10)$$

The inverse permutation P^{-1} is defined as the multiple rotations to the left, in the following way:

$$P^{-1}(i) = ROL(12, [1]) + ROL(28, [2]) + ROL(44, [3]) + ROL(60, [4]), \quad (11)$$

where $i \in \{1, 2, ..., n\}$.

2) Diffusion for multiple rotations

In Fig. 4, a diffusion for (and introduced by) permutation P of the PP-2 cipher is presented, in the case of n = 64. For simplicity, the iteration function h of the cipher is restricted to SL and P layers. Bits dependent on bit number 1 after transformations in consecutive layers are denoted by dots. All bits of the output block are dependent on bit number 1 after 3 layers (i.e., after 2 rounds). Similarly, for the remaining bits.

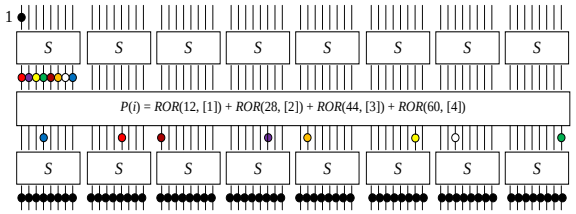


Figure 4. Diffusion for permutation P (n = 64)

Transformation P for bits numbered 1–8, dependent on bit number 1 after substitution S (see Fig. 4), is as follows (bits 1 and 5 are element of class [1], bits 2 and 6 belong to class [2], 3 and 7 – to class [3], while 4 and 8 – to class [4]):

$$P(1) = ROR(12, 1) = 13, P(5) = ROR(12, 5) = 17,$$
$$P(2) = ROR(28, 2) = 30, P(6) = ROR(28, 6) = 34,$$
$$P(3) = ROR(44, 3) = 47, P(7) = ROR(44, 7) = 51,$$
$$P(4) = ROR(60, 4) = 64, P(8) = ROR(60, 8) = 4. \quad (12)$$

3) Implementation of multiple rotations

Permutation P of the PP-2 cipher, which uses the multiple rotations, gives higher diffusion speed than involution P of the PP-1 cipher. Moreover, it is also much faster and easier in the software implementation. Isolation of the classes of bits can be obtained as the result of logical multiplication of the argument (block) by appropriate mask. Thus, calculation of P value, for an argument, requires at most four *AND* operations, four *ROR* operations and three addition operations, performed on n-bit words. Calculation of the inverse permutation P^{-1} during decryption is similar.

III. SUBSTITUTIONS BOXES

The S-box of the PP-2 cipher has been generated using the multiplicative inverse procedure with primitive polynomial \$87 defining the Galois field, so a procedure similar to that used to generate AES cipher S-box as well as that of PP-1 cipher. An important difference to AES cipher S-box is that PP-2 S-box does not have any affine equivalence between its component functions as in AES, which is always the case when an S-box is generated by an unmodified multiplicative inverse procedure.

Nonlinearity of this S-box is 110 and its nonlinear degree is 7. Individual Boolean functions that constitute this S-box S

= (F_0, F_1, F_2, F_3, F_4, F_5, F_6, F_7) have nonlinearities equal to 110 or 112.

Maximum value in the *XOR* profile is 4. For comparison, AES S-box [1] has nonlinearity equal to 112, and all other parameters equal to those of PP-2 cipher S-box.

S-box can be displayed as a 2-dimensional table (Fig. 5). The input is represented as a two digit hexadecimal number where the high order digit is read by giving the row number and the low order digit is read by giving the column number. For example, for an input value 86 the S-box output is FB.

S-box is said to exhibit an affine equivalence is any of its component functions can be mapped to another one using only affine transformations. This is an undesirable property as it theoretically could help to form an algebraic attack on a cipher. Removal of this affine equivalence has been achieved by switching one pair of elements of the S-box table between them. This procedure lowered S-box nonlinearity to 110 (from the initial 112). We consider this a minimal loss of nonlinearity, which in turn allowed the S-box to have that important characteristics missing from AES S-box, that is - no affine equivalence between any component functions.

The S-box of PP-2 cipher is the result of the extensive search of S-boxes generated with multiplicative procedure with randomly selected polynomials. In each of generated S-boxes, a pair of input was randomly searched, switching which would eliminate the affine equivalence present in the S-box. This random search continued until an S-box has been found fulfilling all the required parameters mentioned above.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	9E	81	F7	2F	CC	F1	46	6E	B7	CE	29	76	14	42	E6	BB
1	EC	39	B6	EF	A3	D5	EA	91	3D	37	F0	51	44	C6	0C	BC
2	41	80	AB	1D	6C	D2	C0	ED	00	FE	3B	F9	A4	24	FF	DB
3	4F	7A	4A	B8	A9	4E	79	A8	15	9B	B2	A7	31	52	69	58
4	71	DC	77	99	04	A6	B9	25	E7	92	5E	62	57	89	C1	61
5	D1	66	48	C2	AA	38	2D	8E	65	8C	C3	6A	C8	7B	DA	95
6	90	0E	0A	6B	F4	5B	8D	A1	05	0B	10	03	8B	9D	85	E1
7	BD	19	FA	6D	88	22	E4	4C	AF	49	F8	BE	83	07	FD	D3
8	8F	A5	BF	E8	C4	E5	FB	16	35	3C	64	2C	0D	C5	43	02
9	59	C7	7E	E3	18	CF	06	4B	9C	D0	F3	70	D7	33	87	B1
A	DF	20	E2	EE	F5	32	56	B3	84	74	2B	34	47	36	96	DD
B	63	0F	97	28	D6	5F	7D	9F	53	09	8A	12	5A	AE	1B	3A
C	7F	40	30	A0	D4	27	82	3E	4D	08	7C	1C	17	2E	01	CD
D	B5	CB	54	A2	D9	EB	50	93	F2	3F	1F	9A	13	CA	21	94
E	E9	23	5D	1A	AC	B0	67	86	73	1E	26	6F	45	98	11	BA
F	E0	D8	75	72	AD	55	68	78	F6	2A	B4	5C	C9	60	DE	FC

Figure 5. S-box *S*

PP-2 cipher's S-box is not its own inverse ($S^{-1} \neq S$). To avoid attacks based on algebraic properties of an S-box, after generating the table of inverses, an affine transformation is applied to each entry of the S-box, which introduces diffusion.

This transformation in case of PP-2 cipher is described by the following matrix transformation:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (13)$$

where $[x_7,x_6...,x_0]$ is an entry in the inverse table, $[y_7,y_6...,y_0]$ is S-box output value, and constant c = $[c_7,c_6...,c_0]$ = [10011110] = $9E.

In PP-2 cipher, the constant *c* of the affine transformation has been selected in such a way that the S-box does not have any fixed points (more precisely, the value of *c* has been selected by random search until a suitable value was found that resulted in the absence of fixed points in the S-box). S-box *S* does not have any fixed points when $S(a) \oplus a \neq 00$ and $S(a) \oplus a \neq$ FF for every value of *a*.

IV. KEY SCHEDULE

A. Round key generation

A key schedule of the cipher is an algorithm that, given the key *k*, calculates the subkeys K_i for each of its *r* rounds, and consequently the round key key_i. The length of the block is a multiple of 64, i.e. $n = t \cdot 64$, and the length of the key *k* equals $d \cdot n$, where $d, t \in \{1, 2, 3, ...\}$.

The scheme of one iteration consists of *P* - the same permutation as in the processing path of the cipher (Fig. 1) and *S* - the same S-box as used in nonlinear element *NL* (Fig. 6).

In the *i*-th round of key schedule, the auxiliary key K_i is generated. As the result of each iteration sch_i we obtain key_i.

Let

$c_0 = RR(0, (E3729424EDBC5389)) \| RR(1, (E3729424EDBC5389)) \| RR(t–1, (E3729424EDBC5389))$, (14)

$c_1 = RR(0, (59F0E217D8AC6B43)) \| RR(1, (59F0E217D8AC6B43)) \| RR(t–1, (59F0E217D8AC6B43))$, (15)

where $RR(b, x)$ is the *b*-bits right rotation of a binary word *x* and || denotes the concatenation.

Furthermore, let for $t = n/64$

$$K_i = K_{i,1} \| K_{i,2} \| \ldots \| K_{i,j} \| \ldots \| K_{i,t} \quad (16)$$

(similarly for $K_{i,j}$ in each block *KS#j*),

$$X_{i-1} = X_{i-1,1} \| X_{i-1,2} \| \ldots \| X_{i-1,j} \| \ldots \| X_{i-1,t} \quad (17)$$

(similarly for $X_{i-1,j}$ in each block *KS#j*),

$$V_{i-1} = V_{i-1,1} \| V_{i-1,2} \| \ldots \| V_{i-1,j} \| \ldots \| V_{i-1,t} \quad (18)$$

(similarly for $V_{i-1,j}$ in each block *KS#j*), where

$$V_{i-1,j} = KS\#j(K_{i,j}, X_{i-1,j}) \text{ for } j = 1,2,\ldots,t, \quad (19)$$

$$Z_{i-1}=P(V_{i-1}), \quad (20)$$

$$key_i=X_{i-1} \oplus Z_{i-1}. \quad (21)$$

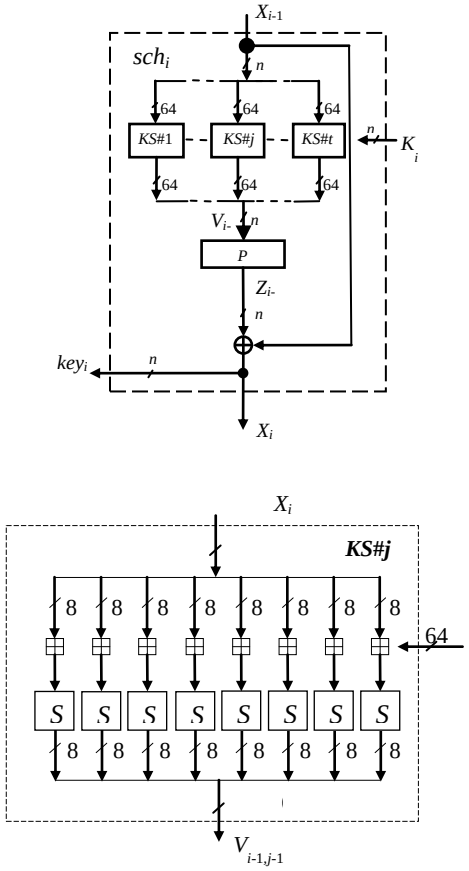


Figure 6. One iteration of the key schedule

B. *Creating auxiliary keys*

For constants c_0 and c_1, and $i = 1, 2, \ldots, \lceil d \rceil \cdot t + r$ we calculate

$$K_i = K_i^* \oplus RR(i-1, c_1),$$

where

$$\left(K_i^*\right)_{i=1}^{\lceil d \rceil t+r} = \left(\kappa_1, \underbrace{0,0,\ldots,0}_{t}, \kappa_2, \underbrace{0,0,\ldots,0}_{t}, 0, \ldots \kappa_{\lceil d \rceil}, \underbrace{0,0,\ldots,0}_{t}, \underbrace{0,0,\ldots,0}_{r-\lceil d \rceil}\right).$$

The round key

$$k_i = \begin{cases} key_{i(t+1)}, & \text{dla } i = 1,2,\ldots,\lceil d \rceil \\ key_{\lceil d \rceil (t+1)+i}, & \text{dla } i = \lceil d \rceil + 1, \lceil d \rceil + 2, \ldots, r \end{cases}$$

The scheme of the generation of round keys for n = 128 and $|k|$ = 256 is presented in Fig. 7.

V. RESISTANCE AGAINST CRYPTANALYSIS

To evaluate the resistance of the PP-2 cipher against the differential and the linear cryptanalysis let us apply the rough method, described in [5] and [7]. The main idea of the rough method is to evaluate the best nonzero approximation of a cipher by a composition of the best nonzero approximation of a single iteration. In the case of the PP-2 cipher, we assume a single active S-box in each round

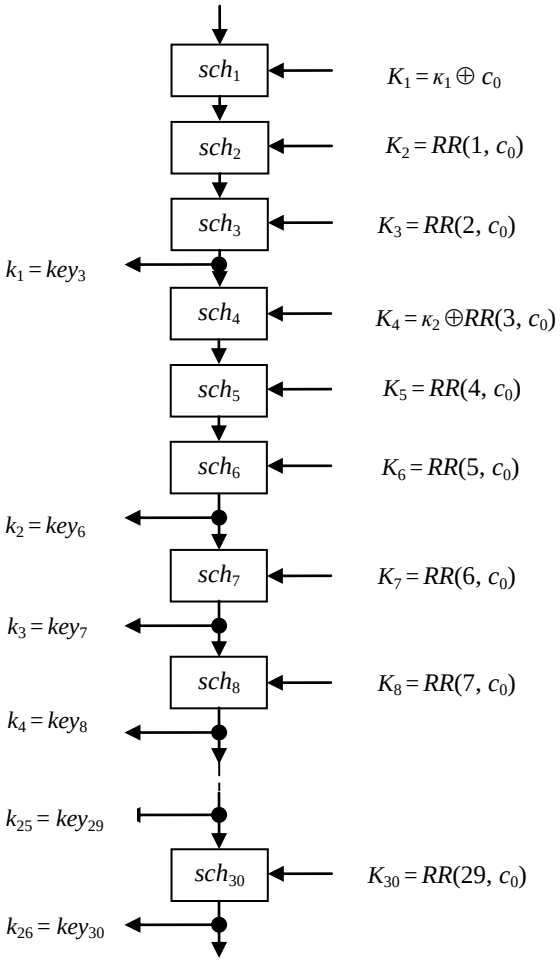


Figure 7. The scheme of the generation of round keys for n = 128 and $|k|$ = 256

The best nonzero linear approximation of the S-box S of PP-2 has the effectiveness $|\Delta p_S^+|$ = 18/256. The effectiveness of the best nonzero differential approximation of the S-box is π_S^+ = 4/256. Assume that for the round function h the same values for effectiveness of the best linear and differential approximation are obtained, i.e. $|\Delta p_h^+|$ = 18/256 and π_h^+ = 4/256. Assume moreover that the best nonzero approximations of PP-2 are composed of r best nonzero approximations of function h. Then the values of effectiveness $|\Delta p_a^+|$ and π_a^+ obtained for the best nonzero approximations of PP-2 are presented in Table II.

TABLE II. UPPER BOUNDS OF EFFECTIVENESS OF THE PP-2 NONZERO LINEAR AND DIFFERENTIAL APPROXIMATIONS

(n, r)	(64, 11)	(128, 22)	(192, 32)	(256, 43)
$\lvert\Delta p_a^+\rvert$	$1.83/2^{33}$	$1.67/2^{64}$	$1.35/2^{92}$	$1.24/2^{123}$
π_a^+	$1/2^{66}$	$1/2^{132}$	$1/2^{192}$	$1/2^{258}$

The best nonzero linear approximation of PP-2 is evidently more effective than the differential one. It does not mean, however, that the linear attack is less complex than the differential one. In the two attacks the number of required texts is of order of 2*n*, where *n* is the block size.

VI. TESTS RESULTS

In this section, we present tentative results of PP-2 performance evaluation. We have measured the encryption speed for three algorithms: PP-1, PP-2, and AES. The reference (unoptimized) implementations of PP-1 and PP-2 were used for this purpose. Therefore, we have chosen partially optimized implementation of AES based on the implementation developed by Brad Conte [9] for comparison. (We plan, in the future, to compare optimized version of PP-2 with best optimized implementations of AES.)

Exemplary implementations use 128-bit key. The data block size is 64-bit for PP-1 and PP-2. All implementations were written in C and compiled using Intel C++ Compiler Professional 11.1 for Windows.

For each algorithm, two versions were generated: 32-bit code and 64-bit code. Experiments were performed using PC computer with AMD Phenom II X4 965 3.4 GHz processor, 8 GB of RAM, Windows 7 operating system (64-bit version).

Tables III and IV present processing speed for data block size 32, 64, 512, and 1024 bytes for 32-bit code and 64-bit code.

TABLE III. THE ENCRYPTION SPEED [MB/S] 32-BIT CODE

	Processed data blocks [bytes]			
	32	64	512	1024
PP-1	3.03	4.08	6.08	6.31
PP-2	12.69	14.23	16.21	16.36
AES	17.80	18.57	19.08	19.29

TABLE IV. THE ENCRYPTION SPEED [MB/S] 64-BIT CODE

	Processed data block [bytes]			
	32	64	512	1024
PP-1	3.38	4.50	6.43	6.60
PP-2	19.35	21.47	25.06	25.16
AES	21.26	22.5	23.31	23.54

As we can see, the PP-2 performs much better than PP-1. For big data blocks encrypted with the same key, the PP-2 outperforms PP-1 by more than 2. For short data blocks, the difference is even greater.

TABLE V. ROUND KEYS GENERATION TIME 32-BIT CODE

	Time [μs]	Processor cycles
PP-1	2.49	8467
PP-2	0.55	1866
AES	0.14	492

There are no big differences in performance between the PP-2 and the AES. For long blocks of data encrypted with the same key AES performs better in 32-bit version, whereas PP-2 slightly outperforms AES in 64-bit version. For short blocks, AES is better in all cases. It means that the round keys generations takes more processor cycles in PP-2 than in AES.

Tables V and VI present measured round keys generation time for PP-1, PP-2, and AES for 32-bit and 64-bit code respectively

TABLE VI. ROUND KEYS GENERATION TIME 64-BIT CODE

	Time [μs]	Processor cycles
PP-1	1.56	5313
PP-2	0.36	1227
AES	0.15	507

As we can see in tables IV and V, the round keys generation requires more time in PP-2 than in AES. On the other hand, from tables III, IV, V and VI, we can deduce that generation of the round keys requires the same amount of time as encryption of about 9.5 bytes of data. It means that performance of PP-2 round keys generation algorithm is fairly good.

VII. CONCLUSION

PP-2 cipher construction details were described in this paper. Cipher structure is different for encryption and for decryption. In particular, a round function construction was presented as well as diffusion layer based on multiple rotations. It was shown, that for block length $n = 64$ bits a global diffusion is reached after two rounds. Number of rounds is dependent on key size.

New S-boxes were designed, S and S^{-1} with very good cryptographic characteristics.

Round key generation algorithm for PP-2 cipher has been considerably simplified in comparison to that of PP-1 cipher. This allowed lowering cipher initialization time. Round keys remained independent, which improves cipher quality when it comes to differential cryptanalysis (such cryptanalysis is more difficult).

ACKNOWLEDGMENT

The paper is a scientific work financed from science research fund in years 2010-2013 as a research project and partially by the grant DS-PB/45-085/12.

REFERENCES

[1] Advanced Encryption Standard (AES), NIST, FIPS Pub. 197, November 26, 2001.

[2] E. Biham and A. Shamir, Differential Cryptanalysis of the Data Encryption Standard, Springer-Verlag, New York, 1993.

[3] K. Bucholc at al., Scalable PP-1 block cipher, International Journal of Applied Mathematics and Computer Science, vol. 20, no. 2, 2010, pp. 401−411.

[4] K. Bucholc, K. Chmiel, A. Grocholewska-Czuryło and J. Stokłosa, PP-1 Block Cipher, Polish Journal of Environmental Studies, vol. 16, no. 5B, 2007, pp. 315−320.

[5] Chmiel K., Methods for differnetial and linear cryptanalysis of block ciphers (in Polish), PUT Press (Wydawnictwo Politechniki Poznańskiej), 1–212, Poznań 2010.

 ISBN: 978-1-61208-298-1

[6] K. Chmiel, A. Grocholewska-Czuryło, P. Socha and J. Stokłosa, Involutional scalable block cipher, Metody Informatyki Stosowanej, nr 3/2008 (tom 16), 2008, pp. 65−75.

[7] K. Chmiel, A. Grocholewska-Czuryło and J. Stokłosa, Evaluation of PP-1 Cipher Resistance against Differential and Linear Cryptanalysis in Comparison to a DES-like Cipher, Fundamenta Informaticae, vol. 114(3–4), 2012, pp. 239–269.

[8] K. Chmiel, A. Grocholewska-Czuryło and J. Stokłosa, Involutional block cipher for limited resources, 2008 IEEE Global Telecommunications Conference, Computer and Communications Network Security Symposium, IEEE eXpress Conference Publishing, ISBN 978-1-4244-2324-8 (CD ROM), New Orleans 2008, pp. 1852−1856.

[9] B. Conte, Implementation of AES in C, http://bradconte.com/aes_c, [retrieved: July, 2013]

[10] M. Misztal, Differential Cryptanalysis of PP-1 Cipher, Proceedings of International Cryptology Conference – Recent Advances in Cryptology and National Telecommunication Security Systems, Wojskowa Akademia Techniczna, Warszawa 2011, also in Annales UMCS Informatica AI XI, 2, 2011, pp. 9–24.

 ISBN: 978-1-61208-298-1