Towards User-centric Identity Interoperability for Digital Ecosystems

Hristo Koshutanski
Computer Science Department
University of Malaga (Spain)
Email: hristo@lcc.uma.es

Mihaela Ion
CREATE-NET Research Center
Via alla Cascata 56/D, Trento (Italy)
Email: mihaela.ion@create-net.org

Luigi Telesca
CREATE-NET Research Center
Via alla Cascata 56/D, Trento (Italy)
Email: luigi.telesca@create-net.org

***Abstract*— Digital Ecosystem is a new paradigm for dynamic IT business integration. Its main focus is to provide micro- and small enterprises with technological solutions bootstrapping their growth and cooperation. In a Digital Ecosystem, institutions compete in some business aspects and collaborate in others, and thus form stable and unstable coalitions. Such a dynamic environment becomes a bottleneck for identity management solutions. Existing and well-researched solutions for identity federation are either too restricting and not flexible enough to support the dynamic nature of ecosystems or they are too complex and difficult to adopt by small enterprises.**

In this paper we present a model targeting cross-domain identity interoperability between distributed ecosystem entities. The model is based on the recent OASIS SAML v2.0 standard to provide interoperability and convergence between existing identity technologies. The paper presents the basic and extended identity models for single services and service compositions. The aim of this research is to allow small and medium companies to use and enhance their current identity technology with a practical and easy to adopt identity management solution that scales up to the dynamic and distributed nature of digital ecosystems.

Keywords: Identity management, Single-sign on, Digital ecosystems, Identity interoperability, User-centric identity profile.

I. Introduction

Digital Ecosystem (DE) [14] is an innovative multidisciplinary concept that explains how dynamic business coalitions can be supported through an open IT environment. DE are a set of open standards, joint infrastructure and advanced services supporting the dynamic evolutions of business relations and virtual organizations over time.

DE complements current Service Oriented Architectures (SOA) with a sustainable approach that overcomes the limitations of SOA. SOA provides service composition opportunities only though a high level ”centralized” architecture and broker/integrator coordination. SOA/WS standards can, in fact, facilitate services integration only in the context of a well defined business domain where a big player can dictate certain rules, standards (even proprietary) and/or basic communication conditions.

However, software interoperability and integration are totally different in the context of a whole industry, where a role of business broker is not well specified in advance and can change over time based on market conditions and target markets. In this context, establishing a certain level of control and coordination, even in modest amounts, is very difficult. The justification for this non efficient behavior is motivated by the fact that usually integrators try to lock in other players (usually suppliers) in order to avoid vertical and even horizontal competition controlling the supply chain and the evolution of the target markets.

Digital Ecosystem aim to overcome those limitations. DE provides a Peer to Peer (P2P), interoperable, service infrastructures supporting the dynamic nature of the business ecosystems. DE is therefore an open, decentralized, communication and service infrastructure populated by networked agents (big and small and medium enterprises, service brokers, public bodies, end users), data, knowledge models and software services supporting the interaction of the above mentioned ”species” and the evolution of the open ecosystem. Thanks to this open (joint ownership of the infrastructures) and friendly approach DE provides new infrastructure enablers that can facilitate the fast deployment of services by Small and Medium companies (SMEs) or even individuals.

In such dynamic environment agents are able to evolve dynamically through incessant transactions, alliances, adaptation and composition of service offerings. They negotiate (cooperating and competing) with the final objective to survive to market competition while increasing their wealth (not only increasing the profits) and competitive advantage. Thanks to the DE approach economic actors can perform different roles (services producers and consumers) over time and with their active participation they can open new market opportunities, business models and service deployment methods. Figure 1 shows the high-level stack view of a Digital Business Ecosystem [15].

Current closed federation approaches are too restrictive and not sustainable over time to support unstable alliances and virtual coalitions. Those solutions are also very complex and not affordable by SMEs. In this dynamic context, identity management solutions need to be more open and easy to use and they should be able to connect entities coming from different business domains and using different certification schema.

Ecosystem-oriented architectures. Today users and organizations employ a broad set of digital components, such as software products, business services, knowledge (documents,

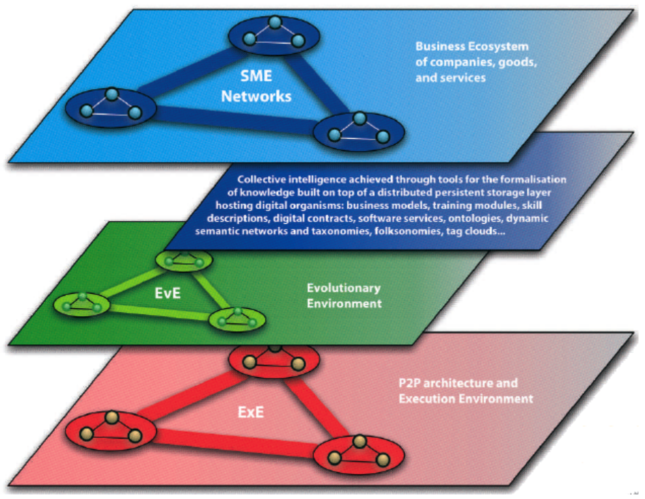
SME
Networks
Business Ecosystem of companies, goods, and services
Collective intelligence achieved through tools for the formalisation of knowledge built on top of a distributed persistent storage layer hosting digital organisms: business models, training modules, skill descriptions, digital contracts, software services, ontologies, dynamic semantic networks and taxonomies, folksonomies, tag clouds...
EvE
Evolutionary Environment
ExE
P2P architecture and Execution Environment

Fig. 1. The stack view of the Digital Business Ecosystem [15]

e-mails, portals, wikis, etc.) and data structure representing business objects. An Ecosystem Oriented Architecture (EOA) [4] can be defined as a meta-level architecture for DE, allowing for the description of digital components and processes that are involved. The idea behind EOA is the extension of the classical SOA in a distributed and semantic rich architecture designed to support the interoperability and the integration of the different processes that characterize a DE. In an EOA all the components interact together, crossing organizations' boundaries and forming a DE that connects different systems, and exchange information using common data representations, like XML and other standard formats. All the EOA services are deployed on a distributed, peer-to-peer platform and described by business and functional models, using Unified Modeling Language (UML), adding in this way semantic to the service description.

This decentralized architecture defines a topology and a replication schema that depend on a set of collaborative peer nodes. A peer-to-peer network supports this topology, and the data replication across the network is guaranteed by a *Distributed Knowledge Base* (DKB) that stores and retrieves contents in a smart way. The final picture is a peer-to-peer and service oriented architecture with high integration capabilities offered by the adoption of open standards where the gap between business abstraction and software implementation is bridged by the adoption of model driven methodologies.

Identity technologies. Institutions use different types of authentication and and identity certificate technologies such as X.509 [26], SPKI [17], Kerberos [11], SAML [16] etc, which are not always compatible and interoperable with each other. Users often need to access applications, services or a composition of services located at different administrative domains.

WS-Policy [24], WS-Trust [25] and WS-Federation [23] cover a wide range of requirements and at the same time are difficult to suit immediately for small and medium size enterprises (SMEs). Existing standards are heavy and difficult to understand, and require a high-cost and longer term deployment, and therefore suitable for large enterprises.

What SMEs in DEs need is a simple and easy do adopt model that allows them to enhance their current identity technology with an extension to identity interoperability management [12].

Paper contribution. Our approach aims at automating the process of identification between ecosystem partners. We emphasize on practical solutions which are clear and easy to implement. The model is based on the new SAML (v2.0) standard [16] for providing proper identification. SAML faces interoperability on the message level and helps to automate and converge when the technologies are not compatible. We face distributed identity storage by the use of user profiles. A user profile is an abstract view of a client's identity information that is stored in a decentralized manner. Decentralization is faced by use of peer-to-peer replication of user profiles on trusted nodes, part of DKB.

The paper is organized as following. Section II defines the core model functionalities scaling to DE's nature. Section III presents the model architecture with its message flow, and the model extension to service compositions. Section IV presents details of a possible user profile structure. Section V discusses the concept of token transformation for interoperability with its inherent SAML-based functionality. Section VI overviews current identity management standards, and Section VII concludes the paper.

II. Identity Model Functionality

Let start by summarizing the main functionality an identity management model for DEs should cover.

1) Dynamic trust relationship establishment and management between identity providers across administrative domains. Identity providers should flexibly define (new) trust relations with other identity providers and the relations should be easy to discover (by end-users) to allow for dynamic, on the fly, trust discovery. The dynamic definition of trust relationship will allow identity providers to maintain and update when their trust relations change, which is often the case in DEs.
2) Enable single sign-on mechanism on top of the established trust relations for decoupling a service provision logic from an authentication (identification) process.
3) Allow user-centric identity management of available credentials for easy discovery and identification to third party service providers/administrative domains.

A. Single sign-on mechanism for identification abstraction

Single sign-on (SSO) mechanism has been developed to provide separation and abstraction between a service provision logic, and an identification process to service users. The abstraction aims at encapsulating the management of an identification process by a third-party called an identity provider (IdP). By adopting an SSO mechanism a service provider (SP) offloads the burden for a proper user identification to a trusted IdP.

When more than one SPs share a common IdP they form a simple form of identity federation where a user with a single sign-on can access all services under the federated SPs.

Although a SP defines a trusted IdP, still a SP provides access to resources based on a simple form of user authentication. The SP's authentication process is a verification process of whether a user has been identified by a trusted IdP. Generally, an SSO mechanism can be initiated by both a SP or a user accessing a service. The user-initiated SSO allows users to select a user-trusted IdP that a SP should contact for a user identification. Even though the user-initiated SSO has several application scenarios for Web-based authentication (e.g., OpenID SSO mechanism[1]), it is not suitable for our purposes.

Our targeted SSO is the SP-initiated SSO that allows a SP to define and maintain its own (federated on not) IdPs. Below we describe the SP-initiated SSO case.

1) A user is accessing a service under a SP without any login information.
2) The user is not recognized and gets redirected to a SP-trusted IdP.
3) The user is required to sign on by providing required credentials to the IdP (e.g., user name and password or an identity certificate).
4) If successful authentication, the IdP redirects the user back to the SP including information about the user authentication, often in a form of a security token.
5) The SP, in its turn, verifies if the user authentication has been done by the IdP (a simple authentication process if an IdP has digitally signed user authentication information), and gives access to the requested service. A security context (session) is created based on the user authentication.

Let us start by defining the key components of the model.

1) *User*: any entity that can be identified in the network (peer or web browser user, institution or person)
2) *Service Provider (SP)*: any identifiable entity that has one or more services or resources available to other entities.
3) *Identity Provider (IdP)*: any entity that is able to provide digitally signed credentials to other entities.
4) *Digital Ecosystem (DE)*: distributed digital environment where both partners and competitors are present and where stable and unstable coalitions are created; coalition of digitally represented partners with few or no a priori established trust relations. Thus the notion of ecosystem comprises cooperative and competitive relations.

We target an identity management model for decentralized peer-to-peer ecosystem domains. All entities are considered equal and there is no hierarchy of ecosystems. Any peer can be an IdP or a SP, or both. Each user can issue a certificate to other users. Each user has a list of trusted IdPs. Each IdP has a list of acceptable security tokens. An IdP issues certificates to users based on:

- security tokens issued by the provider itself, or
- security tokens issued from IdPs with whom it has trust relationships, or
- user registration information.

B. Multiple user identities and technology standards

In a network of interconnected digital ecosystems, users and companies use different kinds of certificates obtained from outside the system. Companies have own X.509 certificates issued by Certification Authorities outside the system and which they are obliged by law to use when doing online transactions. SMEs often have their own proprietary solutions for identification of their employees such as user name and password, ad hoc secure tokens or adoption of OpenID for Web-based access.

After joining a DE, users (partners) obtain a variety of certificate tokens issued by IdPs for particular business needs. However, partners that already have ad hoc identity tokens or user name/passwords authentication should be able to use them for the sake of providing identity information to IdPs that are to certify partners' identity. The reason for that is to unify identity management between partners with already existing identity token standards.

Each IdP has the responsibility to provide proper pseudonymity to end users. An IdP either issues a user pseudonym on its own or allows users to define it and then certifies the pseudonym in a security token to a SP. A SP explicitly asks an IdP to reveal user identity in case of user misbehavior.

C. User-centric identity profile

Having multiple identity certificates issued by different IdPs, it becomes difficult for a user to manage and locate all of them when needed to access a service, especially in the case of distributed services.

Users connect to a DE either via a portal (a Web browser) or via a rich client system installed on their computers. In either of the cases a user needs a way to manage its credentials, user name/passwords and public/private key pairs. For that purpose we adopted the use of a *user profile*. A user profile contains all available information about user's identity obtained from the user's interactions within DEs. Its main purpose is to provide an abstract view of what identity credentials are available, where they are available (e.g. local or remote storage) and how to obtain them (e.g. via authentication to an IdP by user name/password or via an LDAP[2] storage etc).

An important issue is how to allocate, store, and retrieve the user profile. The profile contains sensitive information that is necessary when communicating with entities in a DE. So, the profile must be protected from unauthorized access (no one except the owner of the file) and at the same time must be

[1]http://www.openid.net

[2]http://www.openldap.org

available on demand (avoid denial of service/availability). To address these issues we adopted to keep the profile encrypted and replicated on trusted peers. The encrypted profile is only meaningful to its owner and reliably obtained via a trusted peer-to-peer network as part of the DKB in DEs.

Another issue worth mentioning is the availability of a profile to be shared (used) by multiple entities. This may often be the case for SMEs where selected employees are allowed to use the profile and therefore represent the company in on-line business negotiations. A possible approach is to define an access policy for each profile that encodes who can use the profile and under what conditions. The access policy is optional and if not explicitly specified it should have the default value of only read and write permissions for the profile's owner. A simple and yet effective solution for the policy model is the use of Access Control Lists[3].

We adopted the concept of peer-to-peer trusted network, as provided by the DKB of DEs infrastructure, to replicate and provide service availability when locating and loading user profiles. The problem of how to establish a proper methodology for data replication is beyond the paper scope, and some works can be found in [13], [22].

A user is required to remember a user name and a password in order to login into a DE. The user name and password are obtained on initial user registration to a DE. Once registered, whenever the user logs in another (or a same) DE with its login information, the DE's infrastructure takes the responsibility to allocate and retrieve the encrypted user profile.

When a user starts a new session, its profile is to be downloaded on a secure memory (e.g. browser s-box) of its Web browser or a local client and then decrypted. Once decrypted the profile is ready to be used and processed by the Web browser client or the local client. On end of a session, the user profile is encrypted again and updated on the associated trusted node (peer) and then replicated on other trusted peers.

In the case of a local client installed on user's own machine, the profile could be locally copied and stored so that it could be loaded from the client's machine next time. However, in this case the profile must also be stored and replicated on other trusted peers in order to provide availability and actualization if shared among multiple users.

The user profile is encrypted with a long master password, usually a key phrase, known only to a user. The master password should be different from the user password needed for user authentication to a DE. Thus, a user has to remember one login information and one master password in order to facilitate a secure profile storage.

D. *Identity profile evolution over time*

A user profile contains information about available identity certificates, public/private key pairs and user authentication information needed to access and obtain security tokens.

User identity information obtained outside DEs should be updated (imported) in the user profile so that it can be re-used when the user does business interactions with partners in DEs. When a user first time registers to a DE and creates its initial profile, it is requested to import the already available identity information. However, a user can start from no identity information and collect it on a step-by-step basis when interacting with SPs and their trusted IdPs.

An important aspect here is the possibility of evolving user's identity token information dynamically, during normal user interactions with ecosystem partners. After each interaction with an IdP, the user's client (web or local) automatically records the information on the new identity token for subsequent use. The information stored should detail the new token, issuer, authentication process used to obtain the token (e.g., by another token authentication or by user name/password), token type, validity and location of token retrieval, refer also to Section IV. This would allow users to dynamically discover new trust relationships between IdPs and obtain the respective identity tokens for proper authentication, as discussed later in Section V-C.

E. *Token transformation for interoperability: SAML approach*

To approach proper identity management first we need to define a way to cope with the incompatibility of the variety of standards and solutions. Here we borrow the concept of credential transformation from one type to another as already introduced in the WS-Trust standard. To address the problem we have to convert identity information from one certificate technology to another one compatible with the current domain of business.

We have to provide a way for a client identified within one standard to be able to use its identity information when communicating with a SP using another identity representation standard. The issue we take into account is that SMEs may adopt their own (ad hoc) certificate tokens or mechanisms to manage identities of their employees.

To cope with this wide range of identity mechanisms we make the following assumptions.

- Each SP adopts the identity standard best suiting its needs but its trusted IdP should support as a default authentication the SAML standard (especially v2.0). It means that a SME could preserve its existing identity management infrastructure but should enhance its trusted IdP to be SAML-aware, i.e. the IdP should issue SAML authentication assertions derived (transformed) from any of the standards the IdP already supports.
- In order to provide a correct semantic identification and processing between different identity technologies we also impose a SP to be SAML-aware (as a default setting) for its interactions with a trusted IdP. In this way, each SP adopts SAML identity assertions as means of proof of identification between the SP and its trusted IdPs. For example, a transformation of SPKI to SAML and then of SAML to X.509 may be semantically incorrect due to the different design goals behind the X.509 and SPKI standards.

[3]http://en.wikipedia.org/wiki/Access_control_list

With the new SAML release, the standard allows to express identity assertions within a context of many types of authentication, such as X.509, SPKI, Kerberos tickets, user name/password, etc. Thus, SAML becomes a suitable message format standard for unifying identification information of different identity standards. SAML authentication assertions are used when accessing or negotiating with different ecosystem domains.

For example an IdP that supports X.509 and user name & password authentication to be functional/compatible in our framework it has to also support the following authentication to SAML-based conversion:

- X.509 token-based authentication to SAML identity assertion
- User name & password authentication to SAML identity assertion.
- SAML-based authentication to SAML identity assertion.

Example 1: Let us suppose that a SP_1 only accepts X.509-based authentication to identify entities and that SP_1 trusts IdP_1 to validate and authenticate users based on X.509 tokens. Now, let IdP_2 identifies users based on SPKI tokens, and let a user has a SPKI certificate issued by the IdP_2. If IdP_1 and IdP_2 have bilateral contractual relationship of sharing users' identities for the sake of common service usage, following our model assumptions, the following conversions are to be provided:

IdP_1: X.509 token-based authentication to SAML identity assertion.
IdP_1: SAML-based authentication to SAML identity assertion.
SP_1: SAML-aware for a proof of authentication from IdP_1.
IdP_2: SPKI token-based authentication to SAML identity assertion.
IdP_2: SAML-based authentication to SAML identity assertion.

With the above assumptions the user is able to automatically identify itself to SP_1. To do so, the user has to contact its IdP_2 and request for a SPKI-based authentication to SAML identity assertion transformation. Based on the model assumption, IdP_2 provides a remote authentication (e.g., SPKI-based with challenge/response) in order to properly authenticate the user. Based on the authentication information and user identity in the SPKI certificate, the IdP_2 digitally signs a SAML authentication statement with the result of the authentication, and returns it back to the user. The user forwards the newly obtained token to IdP_1.

Since IdP_1 has a contractual trust relationship with IdP_2, IdP_1 accepts the SAML assertion, by verifying its signature, and issues a new SAML assertions to SP_1 for proof of entity identification. SP_1 is SAML-aware and trusts IdP_1 for identifying entities and provides access to the desired service.

III. Identity Management Model Architecture

Figure 2 shows the basic model architecture and workflow of messages between the main actors. The message flow of the model is the following:

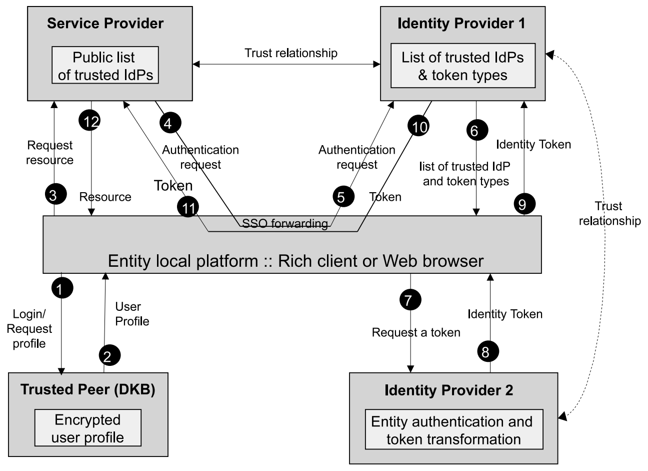

Fig. 2. Model architecture and communication scheme

1: A user requests its profile from a trusted peer storing it by authenticating himself with the user name/password from the registration process. Information of ecosystem trusted peers is obtained (possibly publicly available) when users join the ecosystem.
2: On successful authentication, the trusted peer retrieves and sends the encrypted user profile.
3: The user decrypts the profile (with the master password) and starts using ecosystem services. It makes a request to a SP to access a service.
4–5: The SP redirects the user to a trusted IdP_1 (SSO use case).
6: The user has no credentials issued by the IdP_1. The IdP_1 sends a list of its trusted IdPs and the accepted token types to the user.
7: The user queries the profile if it has available tokens. The profile is processed to match if there are tokens (information) issued by any of the IdPs from step 6. If no credential is matched then the user (possibly) has to register to IdP_1 to obtain an identity token. If an identity token is found then the user extracts it either from the profile, or requests it from the remote IdP that issued it. If a match of IdP and token type then the user just presents the certificate as it is. In case of more than one possible matches the user is prompt to choose which token to use. We note that the user can perform steps 8 and 9 even if it has the right credential match of IdP and token type but does not have the token locally in the profile. In such a case, the user obtains it via a remote authentication (e.g., LDAP server storage).
8: If a credential match, e.g., of IdP_2 but with different token type then the user requests IdP_2 for authentication and transformation to a SAML identity assertion. On successful authentication, IdP_2 issues a SAML assertions and returns it to the user.
9: The user forwards the certificate/SAML assertion to IdP_1.

10: IdP_1 verifies and validates the certificate and issues a SAML assertion to be forwarded to the SP.
11: The user is redirected to the SP which accepts tokens from IdP_1.
12: The SP verifies the new certificate and provides the requested resource to the user.

We note that in step 10, IdP_1 certifies the authentication outcome in step 9 with an identity token acceptable by the SP. The only case in which IdP_1 does not issue a new token is when the user (already been in contact with the SP) presents a same identity token issued by the IdP_1 last time. In this case, after certificate verification and validation, IdP_1 forwards the token to the SP.

The SAML standard is to be used when user authentication format with an IdP is different than the one agreed between the SP and the IdP. For example, in case of X.509 user authentication to IdP_1 and a same X.509 format agreed between the SP and IdP_1 then, IdP_1 may not issue a SAML authentication statement to SP but use the X.509 format.

Step 11 is a point where the user profile records and stores the new identity token and associated information for a subsequent use.

A. Model extension to service composition

Digital Ecosystems allow companies to cooperate with each other, form coalitions, and thus use service compositions suitable for their business models. An important requirement for an identity management model is to support composition of services. We extend the basic model presented above to cope with the case in which one service relies on services from other providers. We assume that the service composition model occurs between SPs having contractual trust relationships.

In a service composition scenario, the service provider aggregating services from other service providers needs to run the services on the name of the user and, as so, he has to authenticate the user to the other providers. To solve this problem we adopted the use of *Proxy Certificate* that the client issues to the provider of the composite service.

A Proxy Certificate [18] is derived from and signed by a normal X.509 public key end-entity certificate or by another Proxy Certificate (PC). The identity of the new PC is derived from the identity that signed it. A PC has its own public and private key pair. A PC is identified as such by its extensions. Any X.509 certificate has extension fields to encode different certificate characteristics. A PC has a policy that specifies what conditions must be respected when an entity is using it. Another important issue is that a PC can only sign another PC.

SAML standard v2.0 defines a rich set of subject classification, as part of the SAML identity assertion, that allow entities to be bound to a public-key information. In this case, the real use of proxy certificate in our model is in the SAML context definition. Thus, the message encoding of identity information in a proxy certificate becomes as a SAML assertion and is compatible (message-level interoperable) with the SPs of composite services.

There are two important requirements specified in the policy of a proxy certificate that reflect our identity model.

- *Service scope.* The first requirement is the scope of a PC. We identity the scope of a PC to be the scope of the service being requested by a client. Scope of service means any aggregated service that is directly used for the sake of proper execution of the main service. In other words, any service that is not directly aggregated within the main service (e.g. aggregation of aggregation, or third-party services not part of current aggregation) should fall beyond the PC scope, i.e., not considered as a valid identity certificate on behalf of a client.
- *Level of service aggregation.* To solve the issue of complex aggregation of services that aggregate other services, we propose as a second requirement the level of service aggregation. The purpose of the level of aggregation is to restrict the use of a PC in a chain of service aggregations. Often a client may wish to restrict not only the scope but also the re-generation of next level PCs. For example, to restrict the use of a service of selling books, a user may use level of aggregation 1 indicating that the at most one level of aggregation is allowed, expecting only a product shipping service to be used and not further delegation of PC usage. The level of aggregation should be interpreted as not to derive more PCs longer in chain than the specified level.

Another requirement is a validity period of a PC. Usually, this depends on the particularity of the main service being executed (i.e., the validity of the service transaction). The client obtains such information from the SP hosting the main service. This parameter plays an important part of PC usage. A client may restrict the use of a PC according to his expectations or familiarity with a given SP. If a distrusted SP a client may wish to generate a PC with short validity period to reduce potential misuse of it.

When a SP contacts another SP to execute an aggregated service, the second SP specifies that it needs a PC to execute other services within its aggregated service. To do so, the first SP issues and signs a new PC to the second SP with the following restrictions:

- The derived PC has as service scope the aggregated service to be executed,
- The derived PC has a level of aggregation the level of the predecessor PC decreased with 1 (if not zero),
- The derived PC has a validity period, the remaining validity period of the predecessor PC that signs it (if not expired).

In this way, a next level SP can use the newly derived PC only for the sake of execution of its service. To validate a PC an IdP needs the set of all PCs derived from the client's generated one, and validates if they follow a correct PC derivation as described above.

Figure 3 shows the extended identity model for service compositions. The steps behind the model are the following:

1: The user downloads the profile from a trusted peer.

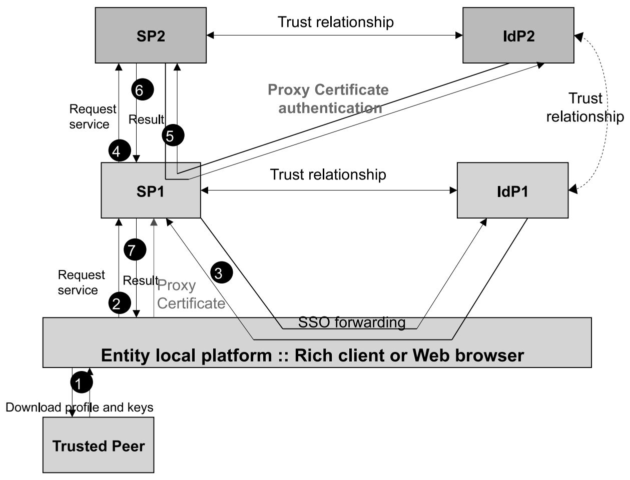

Fig. 3. Service composition using proxy certificates

2: The user requests to access a composite service of SP_1.
3: The user is redirected to IdP_1 to login (SSO use case). On successful authentication following the message flow of the main model, IdP_1 issues a new (SAML) identity token forwarding it back to SP_1. SP_1 indicates to the user that the requested service is an aggregation of services together with a list of the services to be used. The list of aggregated services is an optional (but recommended) element and it serves to precisely define in a PC the service scope. The default value of the service scope is the current service the user requests with a level of aggregation 1. The user issues a PC to SP_1.
4: SP_1 requests a service to SP_2.
5: SP_2 redirects SP_1 to IdP_2 for user authentication. SP_1 authenticates to IdP_2 on behalf of the user using the proxy certificate obtained in step 3. We note that a new PC has its own private/public key pair used for an authentication process. For successful authentication, SP_1 sends to IdP_2: *(i)* the PC issued from the user, *(ii)* the identity token received from IdP_1 for user authentication, and *(iii)* the user original certificate that signed the PC. The last certificate is essentially the one the user has authenticated with to IdP_1.
We made the assumption that service aggregation occurs between contractual trust relationships, in this case, IdP_2 has a trust relationship with IdP_1 and can validate that the original certificate which signed the PC has been used for authentication by IdP_1, by analyzing the identity token issued from IdP_1, and that the PC remains valid according to its specified policy. If successful verification, IdP_2 issues an identity token to SP_2 (binding the original user identity) authorizing SP_1 as running on behalf of an authenticated user.
6; SP_2 runs the service and provides the result to SP_1.
7: SP_1 completes the service execution and provides the result to the user.

In case of more than one SPs on a next level aggregation, e.g., $SP_2, SP_3, ..., SP_n$, the aggregator SP issues a new PC for any of the SPs on the next level, by repeating the extended model n-times. The extended model scheme can be recursively applied in case SP_2 needs to contact SP_3 as next level aggregated service provider. In such a case, SP_2 takes the role of SP_1 in the extended model.

B. Model integration in DE

Ecosystem oriented architectures [5] provide specific mechanisms for peer-to-peer decentralized communications. There is an abstraction communication layer that, close to Grid communications, defines seamless and platform independent service provisioning and execution. In this way ecosystems' services interact with each other transparently of the communication layer and regardless whether service provisioning and execution takes place on a remote or local platform.

Each SP, providing services via the DE's infrastructure, defines a trusted IdP (or a list of them), as a dedicated DE's service so that DE users will be forwarded to it. On the other side, each DE user will benefit of the DKB part of the DE's infrastructure for a distributed storage and retrieval of its profile. The DKB accessibility service requires a user to be a registered DE user, which will bootstrap the identity management model with a token availability from the initial registration. This is especially useful for those IdPs that accept DE's registration tokens for possible user authentication.

An SSO is the main interaction mechanism for a user authentication. An IdP has two main services relevant to the proposed model: an authentication service and a transformation service. Since the services an IdP provides are DE's dedicated services, the SSO mechanism (between an SP and an IdP) is to be based on top of the DEs communication layer. Thus, a user will use the DE's standard mechanism for service accessibility in both when requesting a secure token transformation and when authenticating to an IdP.

A comprehensive identity management solution for DEs is tightly bound to the definition of suitable trust and reputation mechanisms that benefit (are based upon) the identity model. The work in [9] defines a peer-to-peer reputation framework for quantifying trust on different levels of DEs stepping on the identity model in [12]. The work in [10] complements [9] by presenting an agency-based reputation model as a more reliable trust quantification schema. The agency reputation model defines an interoperation schema between agencies to provide a scalable reputation solution to DEs. Our aim with the above approaches is to define a targeted trust and identity management framework for DEs that scales to the needs of SMEs.

IV. User-centric Identity Profile

In this section we will describe the structure and syntax of a user profile.

User profile structure. The user profile is built using RDF (Resource Description Framework) meta-model and XML syntax [20]. RDF provides a language for representing information and information modeling. RDF works on the basis of making statements about resources. These statements about resources are given in the format of subject-predicate-object [19]. The subject refers to the resource, predicate refers to a property or aspect of the subject, and object assigns a value to this predicate.

RDF identifies entities using Web identifiers (called Uniform Resource Identifiers, or URIs), and describes resources in terms of simple properties and property values. This allows RDF to describe statements about resources as a graph of nodes and arcs representing the resources, and their properties and values.

The user profile has a generic structure of:

- A standard v-Card format, and
- A listing of relevant identity token information available to a user.

The user profile lists all available security tokens together with relevant token information. The user profile provides a unified view of the user's identity information. Users get certificates from interactions with different DE domains. The user profile will be referenced across the DKB as an I-name XRI reference [2]. The following example of an RDF graph depicts the structure of the user profile:

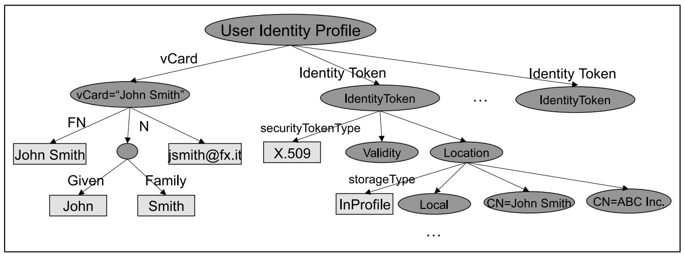

Fig. 4. User profile example

The JENA framework[4] provides a programmatic environment for reading, writing, querying and updating RDF documents in several formats such as RDF/XML and Turtle. We will overview the functional description of the core classes used in the profile structure.

- `UserData` A class encapsulating basic user information provided during a registration process. It corresponds to the vCard information in the profile.
- `Profile` A class encapsulating the identity profile that contains information about all credentials a user has.
- `IdentityToken` A class encapsulating/corresponding to one credential entry in a user profile.
  - `TokenType` – X.509, SPKI, SAML assertion, user name & password, etc.
  - `Subject` – subject name of the user in a given token, i.e. how the user is known to a given IdP.
  - `Issuer` – issuer distinguished name as defined in an identity token.
- `Location` A class encapsulating a location of a certificate. Token availability in a profile and how to access it. If the token type is user name & password, then the token will be contained in the profile. If a different token type, the token will be either stored in an PKCS#12 attached to the RDF profile or will be stored on an external LDAP (Lightweight Directory Access Protocol) server.
- `Validity` A class containing a validity period of an identity token, in a format `notBefore` and `notAfter` dates.
- `AccessInfo` A class encapsulating the information on how to access/retrieve a certificate. Information about the location of the server (URI), the location of the certificate (distinguished name: DN) and possibly user name and password information for accessing the token will be available in the profile.

[4]http://jena.sourceforge.net

Based on the core classes the following methods/interfaces are provided for a user profile management.

- `createProfile(UserData)` Creates a user profile based on the information in class UserData. It creates a registration data such as name, address, email, etc. It creates also the vCard as described in the RDF schema.
- `addIdentityToken(IdentityToken)` Adds an identity token information to a current user profile.
- `deleteIdentityToken(IdentityToken)` Deletes a credential information from a current profile. This usually happens because a certificate has expired or an Identity Provider leaves a network or is no longer trusted.
- `matchIdentityTokens(List_of_Trusted_IdPs)` Returns a list of matched IdentityToken elements. It queries a current user profile for all credentials matching an IdP and a TokenType from the input list. Later in the section we will describe what data structure an IdP returns to a client for a list of trusted IdPs used for the query process.

Identity credential token schema. While the vCard structure is obviously a syntax supported by a standard schema [21], the identity token syntax needs to be defined within the RDF structure. The RDF graph for an identity credential token is depicted in Figure 5.

We will use Turtle [3] for expressing the structure of the RDF schema. Turtle allows RDF graphs to be written in a compact and natural text format. The listing below shows the identity token schema:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@base <http://www.one-node.org/2008/04/profile> .

  :IdentityToken a rfds:Class .
  :Validity a rfds:Class .
  :Location a rfds:Class .
  :AccessInfo a rfds:Class .

  :securityTokenType a rdf:Property ;
        rdfs:domain :IdentityToken ;
        rdfs:range [
          a rdf:Alt;
          rdf:_1 rdfs:datatype("X509", xsd:string) ;
          rdf:_2 rdfs:datatype("SAML", xsd:string) ;
          rdf:_3 rdfs:datatype("SPKI", xsd:string) ;
          rdf:_4 rdfs:datatype("UsrnPswd", xsd:string) .  ] .
```

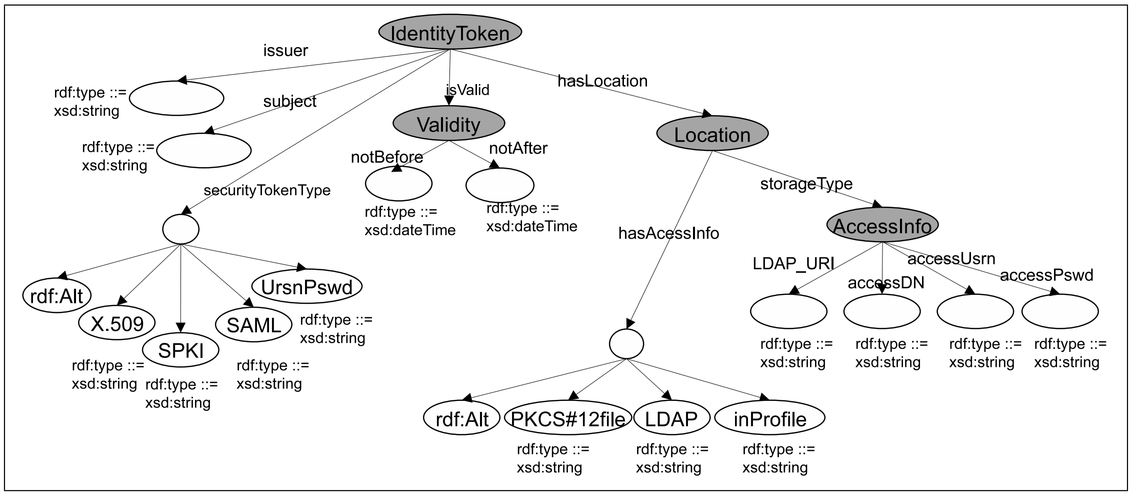

Fig. 5. User profile identity token schema: an RDF graph

```
:isValid a rdf:Property ;
      rdfs:domain :IdentityToken ;
      rdfs:range :Validity .
:notBefore a rdf:Property ;
      rdfs:domain :Validity ;
      rdfs:range xsd:dateTime .
:notAfter a rdf:Property ;
      rdfs:domain :Validity ;
      rdfs:range xsd:dateTime .
:issuer a rdf:Property ;
      rdfs:domain :IdentityToken ;
      rdfs:range xds:string .
:subject a rdf:Property ;
      rdfs:domain :IdentityToken ;
      rdfs:range xds:string .
:hasLocation a rdf:Property ;
      rdfs:domain :IdentityToken ;
      rdfs:range :Location .
:storageType a rdf:Property ;
      rdfs:domain :Location;
      rdfs:range [
        a rdf:Alt;
        rdf:_1 rdfs:datatype("PKCS#12file", xsd:string) ;
        rdf:_2 rdfs:datatype("LDAP", xsd:string) ;
        rdf:_3 rdfs:datatype("inProfile", xsd:string) .   ] .
:accessInfo a rdf:Property ;
      rdfs:domain :Location ;
      rdfs:range :AccessInfo .
:LDAP_URI a rdf:Property ;
      rdfs:domain :Location ;
      rdfs:range :AccessInfo .
:ND a rdf:Property ;
      rdfs:domain :AccessInfo ;
      rdfs:range xds:string .
:accessUsrn a rdf:Property ;
      rdfs:domain :AccessInfo ;
      rdfs:range xds:string .
:accessPswd a rdf:Property ;
      rdfs:domain :AccessInfo ;
      rdfs:range xds:string .
```

V. Identity Token Transformation for Interoperability

The main SAML objective is the ability of expressing assertions about a subject in a portable fashion so that other applications across domain boundaries can trust it.

Authentication statements assert to the service provider that the principal did indeed authenticate with the identity provider at a particular time using a particular method of authentication. Other information about the authenticated principal, called the authentication context, can be inserted in an authentication statement.

SAML authentication statement. A SAML authentication statement defines the following triple: ⟨Issuer, Subject, Validity_Period⟩. Interactions between a user and an IdP for a SAML identity assertion transformation occur within a SAML context, i.e. using the SAML authentication request/response protocol.

The authentication process will be based either on an identity token issued by the IdP or a user name and password authentication. For example, if a user has a SPKI token issued by an IdP and the user needs to have a corresponding SAML identity assertion, the user will initiate a SAML authentication request to the IdP. The authentication process will be based on the SPKI token the user has from the IdP (via challenge/response for authenticity). On successful authentication, the IdP will issue a SAML authentication statement with a userID taken from the SPKI token. We note that an optional input to the transformation interface can be provided allowing a user to specify the need of a pseudonym to be used in the new SAML authentication token.

SAML authentication context. A relying party (a SP's trusted IdP) may require information additional to the assertion itself in order to assess its level of confidence in that assertion. SAML does not prescribe a single technology for authentication and it may vary from an IdP's to IdP's policy. For that case a SAML authentication context is provided to specify additional information, to the authentication process generating a current SAML token, such as what authentication mechanism or method (e.g., password or certificate-based SSL) was used.

Thus, in our example, the IdP issuing the SAML authentication token will additionally specify an authentication context as SPKI-based SSL authentication. Based on that information, the relying IdP can infer what authentication took place and generate the SSO token response (to the SP) with longer/shorter session validity period, or even refuse to accept the SAML token.

```
          <List_of_TIdP> ::=  <IdP_def> | <IdP_def> <List_of_TIdP> .
               <IdP_def> ::= <IdP_id> <IdP_accepted_tokens> .
                <IdP_id> ::= [<Public_key_certificate>] <Distinguished_name> [<List_of_TIdP_URL>] .
    <Distinguished_name> ::= <IdP_name_type> <IdP_name_value> .
         <IdP_name_type> ::= "X500" | "I-Name" | "String" .
        <IdP_name_value> ::= <string_value> .
      <List_of_TIdP_URL> ::= <string_value> .
   <IdP_accepted_tokens> ::= <Token_type> | <Token_type> <IdP_accepted_tokens> .
            <Token_type> ::= "X509" | "SPKI" | "SAML" | "UsrnPswrd" .
<Public_key_certificate> ::= <Token_type> <Token_encoding> <Token_value> .
        <Token_encoding> ::= "Base64" | "Binary" .
```

Fig. 6. List of trusted IdPs structure: BNF notation

Some of the possible context authentication schemes relevant for our scope are: SPKI, X.509, Kerberos, PGP, SSL certificate, password, previous session.

A. Token transformation services

We have two main token transformation functionalities. They represent a remote invocation from a user to a trusted IdP server. Essentially, the two functionalities provide a user authentication process via either an identity token (e.g., SSL-based, challenge/response-based) or via a user name&password login.

Token-based authentication to SAML transformation. An interface that transforms from available token formats to a SAML identity token. A user is authenticated based on its available certificate token. On successful authentication the interface transforms the user authentication information to a digitally signed SAML authentication assertion. The interface (optionally) should allow a user to specify an alternative user identity (user-chosen pseudonym) to be bound to the new SAML identity token. This would allow a user to have privacy (to some extend anonymity) in a given domain. If a pseudonym is used in a SAML token the user should not re-authenticate with that token and request for a new pseudonym, i.e. derivation of a pseudonym from a pseudonym should not be allowed.

User name-based authentication to SAML transformation. An interface that transforms a user name to a SAML assertion. If a user name&password match those of IdP's internal database then a SAML assertion is generated with the user name as a user identity in the SAML token. An optional pseudonymity input should allow a user-chosen identity name to be used instead of his original user name in the new SAML assertion. Note that this should not change the original user name of the user but only bound the new user name in the SAML token.

B. Defining a List of Trusted IdPs

Figure 6 shows the core structure used for representing a list of trusted IdPs. A list of trusted IdPs is a set of tuples each identifying an IdP authority. An IdP authority is identified by (optionally) its public-key certificate and by its distinguished name. For each IdP authority identifier we assign a list of accepted security token types from that authority. Note that Figure 6 describes the data structure and not the representation of a list of trusted IdPs.

A suitable representation of the shown structure is in an XML-based format. We assume that there is a commonly shared dictionary between entities for unambiguous processing of the above (labeled) information. If using Web Services technology, a suitable ground for setting up a list of trusted IdPs is the use of WS-Policy framework[5]. WS-Policy provides a set of basic constructs for defining requirements (basic assertions) about service accessibility.

Example 2: (List of Trusted IdPs):

```
<List_of_TIdP>
  <IdP_def>
     <IdP_id>
        <Public_key_certificate>
           <Token_type> X509 </Token_type>
           <Token_encoding> Base64 </Token_encoding>
           <Token_value>
           -----BEGIN CERTIFICATE-----
           MIIB+jCCAWOgAwIBAgICAfQwDQYJKoZIhvcNAQ...
           EENhbGlmb3JuaWEgU3RhdGUxHDAaBgNVBAMT...
           MTQ0NjQxWhcNMDkwOTAxMTQ0NjQxWjA/MQs...
           cmtlcnMgT3JnLjESMBAGA1UEAxMJSm9obiBDb3V...
           -----END CERTIFICATE-----
           </Token_value>
        </Public_key_certificate>
        <Distinguished_name>
            <IdP_name_type>X500<IdP_name_type>
            <IdP_name_value>
              CN=ABC CA Class-1,O=ABC Inc.,C=US
            </IdP_name_value>
        </Distinguished_name>
     </IdP_id>
     <IdP_accepted_tokens>
         <Token_type> X509 </Token_type>
         <Token_type> SAML </Token_type>
     </IdP_accepted_tokens>
  </IdP_Def>
</List_of_TIdP>
```

The example shows an XML representation of a list of trusted IdPs with only one certification authority. The IdP is identified with an X.500 distinguished name, and the accepted security tokens are X.509 and SAML.

C. User profile evolution: Dynamic token discovery

The main feature of a user-centric identity profile is the possibility of dynamic evolution over time. On each interaction with a SP the user profile will update user identity token

[5]http://www.w3.org/Submission/WS-Policy

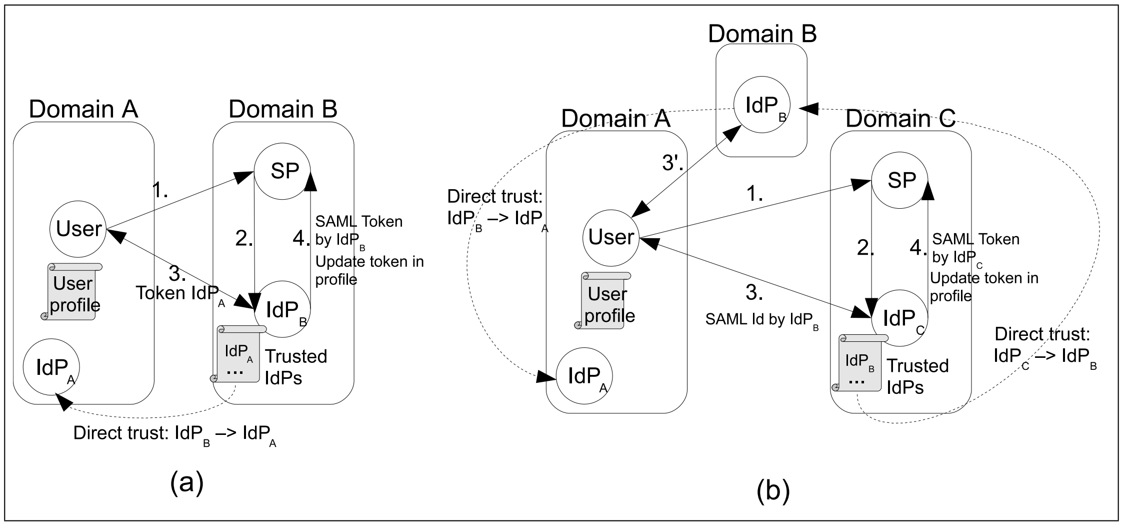
Domain A
Domain B
User
User profile
IdP_A
SP
IdP_B
IdP_A ...
Trusted IdPs
1.
2.
3. Token IdP_A
4. SAML Token by IdP_B Update token in profile
Direct trust: IdP_B –> IdP_A
(a)
Domain B
Domain A
Domain C
3'.
Direct trust: IdP_B –> IdP_A
1.
2.
3. SAML Id by IdP_B
4. SAML Token by IdP_C Update token in profile
IdP_C
IdP_B ...
Trusted IdPs
Direct trust: IdP_C –> IdP_B
(b)

Fig. 7. User profile evolution scenario

information with a new token information obtained from the interactions with the IdP, trusted by the SP. Figure 7 shows the main envisaged scenario.

Figure 7(a) illustrates a basic scenario of a direct trust relationship between a SP's IdP_B and an IdP_A that has issued a token to a user. We represent the two IdPs as belonging to different administrative domains A and B, respectively. The SSO between the user and IdP_B of domain B will authenticate the user by using the existing token information, as already shown in the model. Let assume that the existing token information is of SPKI format and IdP_B accepts only SAML tokens (the default format in the model). After an authentication and a transformation process with IdP_A via the existing token, the user obtains a SAML authentication token that it forwards to IdP_B. Next, on successful authentication, the IdP_B generates an SSO response token (in a SAML format as a default format) forwarding it to the SP (step 4). At this point of the SSO, the user agent updates the user profile with the new SAML token as signed by IdP_B. This will allow a user profile to dynamically evolve as the user interacts with SPs of different DE's domains.

Figure 7(b) illustrates the case of dynamic token discovery when the same user interacts with a SP of domain C. The SP's trusted IdP_C has a trust relationship with the IdP_B of domain B, but has no (direct) trust with the IdP of domain A. Now, since the user has updated its profile with the identity token of the last SSO interaction, the same can discover that it has an identity token signed by IdP_B of domain B. Since the token is in a SAML format the user can directly provide it for an authentication with IdP_C. After a successful authentication the IdP of domain C issues a new SAML token to the SP in response to the SSO authentication process. Again, the new token information is stored in the user profile for a subsequent usage.

In case the SAML token from the scenario in Figure 7(a) is expired at a time of a next user interaction, the user profile, storing the IdP_B location of service authentication and (optionally) what token was used for authentication, the user can, first, request a token transformation (step 3') to IdP_B (by obtaining the list of trusted IdPs and) presenting the identity token issued by IdP_A. Second, on successful authentication, the user presents the new identity token to IdP_C of domain C (step 3).

The above scenarios can be generalized to an N-step authentication process where a user starts with the list of trusted IdPs of a given IdP and continues with the respective lists of trusted IdPs for any IdP in the main list, thus forming a graph (Web) of trusted IdPs (we included an optional link to IdP's respective list of trusted IdPs for each IdP definition, refer to Figure 6). In this case, an algorithm for finding a matching token is a breadth first search algorithm with no loops.

VI. Identity Management Standards

In a distributed environment, users access in one session services located on different administrative domains and need to be authenticated by each of them. If users would have to sign in each time a different domain is accessed and to remember and manage all the different security credentials, the system will not be scalable and become almost impossible to use with a big number of players. In order to allow users to sign in just once and then access services on other domains (single sign-on), organizations establish trust relations between them (on a contractual basis) and allow access to their resources to users which have been authenticated by one of their trusted partners. This is know as *identity federation* and many specifications and implementations are dedicated to it.

Identity federation means sharing of identity information between domains which have a trust relationship or agreement. Once a federation is established, users can experience single sign-on (SSO) inside the circle of trust. SAML and Liberty Alliance define standards for federating identities and single sign-on (SSO).

SAML [16], developed by OASIS, is an XML-based framework for communicating user authentication, authorization and attribute information. SAML provides XML formats and protocols for encoding and exchanging identity information. SAML assertions allow principals to make statements about a subject's authentication, attribute, or authorization details. A subject is uniquely referred to by using an Identifier which can be a real name or a pseudonym. SAML focuses on authentication and attribute statements while authorization statements are the focus of XACML [27]. SAML assertions provide a good way of exchanging authentication information between parties using different and incompatible authentication technologies. Because of this, we are going to use SAML in our model to achieve interoperability between different standards.

SAML also provides standards for federation creation and SSO. However, though SAML v2.0 is very flexible and offers many choices, in practice it is yet hard to establish identity federations with it [6]. Some of the reported reasons are listed below:

1) Long deployment times. For example, deploying SAML-based projects can take weeks or even months with a single partner. One reason for that is the lack of standardized mechanisms for meta-data exchange and trust establishment.
2) Administrators need to familiarize themselves with the details of SAML v2.0 and have a deep understanding of the way federations are secured.
3) SAML 2 has many choices (for profiles and bindings, attributes and identifiers etc.) but lacks guidance on what is the most appropriate to choose.
4) The implementations available today require administrators to provide answers to fundamental questions that require deep insight into the SAML 2 standard: how to manage trust between providers and metadata describing them,which SAML profiles and bindings to use, which messages and what part of each message should be signed, which identifiers and attributes should be exchanged and how, etc.
5) Administrators need to establish point-to-point federation connections with each new partners. This connections take time and affect the scalability of the system when moving from just a few partners to hundreds or thousands.
6) In order to allow small organizations with fewer resources and technically unsophisticated administrators to deploy these standards, the implementation should be easy to deploy and to configure.

To overcome the above shortcomings, Ping Identity[6] and their partners have been working on developing dynamic SAML [6] which should minimize the steps administrators must perform to configure SAML connections securely.

Liberty Alliance provides open SAML based standards for federated network identity. The most relevant technology specifications developed by the Alliance are Identity Federation Framework (ID-FF) [7] and Web Services Framework (ID-WSF) [8]. As of the new SAML version (v2.0) the OASIS technical committee has unified the Liberty standards within one SAML identity framework with a rich set of identity profiles.

[6]Ping Identity Corporation http://www.pingidentity.com

Liberty ID-FF defines identity federation as the linking of distinct user's accounts at the Service Provider and Identity Provider sites. The account linking (or identity federation) is done with the user's consent and must be audited. Liberty ID-FF defines the following required steps for setting up a federation:

1) First of all, businesses form circles of trust based on Liberty architecture and operational agreements that define trust relationships between them.
2) Users federate the isolated local accounts they have with the businesses from the circle of trust. When this happens, the local identifiers (e.g. usernames) of the user are not exchanged between the sites, but instead they exchange opaque user handles.

After this, the users can experience SSO and login at the IdP site and then gain access to the SP sites federated with the IdP. The user needs to allow introductions such that sites of the federation can discover when the user recently accessed a site in the circle of trust and ask the user to federate the accounts. The user can also find a link to trusted SPs from a web site of the IdP. Liberty Alliance specifications are difficult to understand and use for mainly the same reasons we mentioned in the above subsection for SAML. Although business could benefit form deploying Liberty Alliance identity federation solution, the standard is too heavy and organizations face implementation hurdles.

Moreover, it is not always easy for users to discover which accounts they can federate or for SPs to discover which IdP a user is using. This is the case in bigger circles of trust with several IdPs. Liberty ID-FF specifies an optional introduction profile based on cookies which could potentially solve this problem. The idea is to set up a common domain for the circles of trust and to use a common domain cookie accessible by all parties (user, SPs, IdPs). This solution has many shortcomings because it relies on cookies and because common domains need to be updated when trust relations change.

WS-Trust [25] and WS-Federation [23] define standards for federating identities by allowing and brokering trust of identities, attributes and authentication between participating Web services. WS-Trust defines a service model called the Security Token Service (STS), and a protocol for requesting and issuing security tokens. The kind of tokens that a Web Service accepts are described using WS-SecurityPolicy. WS-Federation defines federation as a collection of domains that have established relationships for securely sharing resources. WS-Federation builds on the STS service of WS-Trust and provides mechanisms that simplify interactions between users, IdP (or STS) and SPs. WS-Federation allows to determine policies for obtaining services and cross organizational identity mapping.

OpenID[7] is a decentralized framework for digital identity. The underlying idea is that users can identify themselves on the web like Web sites do with URIs. OpenID allows a user name/password login. The user name is the personal URI and the password is safely stored on the OpenID Provider. To login to an OpenID-enabled Web site, the user is required the OpenID URI and then gets redirected to the OpenID Provider to authenticate. After authentication, the OpenID Provider sends back the user to the web site with the required identity information to login.

CardSpace[8] [1] is an identity selector for Microsoft Windows. It allows users to have different identities, each represented by a card. When a users needs to authenticate to a web site or a web service, CardSpace pops up a set of suitable information cards for the user to choose from. Each card has some identity data associated with it, though not stored actually in the card. The cards can be issued either by an Identity Provider or by the user himself (self-signed).

The CardSpace model is close to our user profile functionality with the difference of having static updates and with no additional information for a token transformation service. However, CardSpace is a suitable underlying technology for a user-centric profile management.

VII. Conclusion

We have presented an identity management model targeting identity interoperability for DEs. The model bridges main identity standards by using SAML as a unified message-level protocol for querying and obtaining authentication assertions. By using SAML one can automate the process of identifying entities in a distributed environment. We adopted the use of a user-centric profile to keep an abstract view of user's available identity information such as identity certificates, user name/passwords, public/private keys, etc. The user profile is replicated and encrypted on trusted peers.

We presented the core interoperability model, its architecture and message flow. Then, we presented the extension of the model to service compositions. To scale to service composition, we adopted the use of proxy certificates with two main policy settings: limiting a service scope and a level of aggregation. The extended model provides the end-user with the ability to control the use of its identity information in case of service aggregations.

Acknowledgements

Hristo Koshutanski was supported by the Marie Curie EIF iAccess (#038978) fellowship of the 6th Framework Program of the European Commission. Mihaela Ion and Luigi Telesca were supported by the project EU-INFSO-IST ONE (#034744) of the 6th Framework Program of the European Commission.

[7]http://openid.net

[8]http://www.microsoft.com/net/cardspace.aspx

References

[1] CardSpace documentation and resources. http://msdn2.microsoft.com/en-us/netframework/aa663320.aspx.

[2] OASIS Extensible Resource Identifier (XRI). http://www.oasis-open.org/committees/xri.

[3] Rdf primer - turtle version. http://www.w3.org/2007/02/turtle/primer/.

[4] P. Ferronato. Architecture for digital ecosystems, beyond service oriented architecture. In *Proceedings of the 1st IEEE Conference on Digital EcoSystems and Technologies (DEST'07)*, 2007.

[5] P. Ferronato. *Digital Business Ecosystems*, chapter Ecosystem oriented architecture (EOA) vs SOA. European Commission, 2007. http://www.digital-ecosystems.org/book/de-book2007.html.

[6] Patrick Harding, Leif Johansson, and Nate Klingenstein. Dynamic security assertion markup language. simplifying single sign-on. *Security & Privacy, IEEE*, 6(2):83–85, March-April 2008.

[7] ID-FF. Liberty Identity Federation Framework (ID-FF), 2007. http://www.projectliberty.org/resources/specifications.php.

[8] ID-WSF. Liberty Identity Web Services Framework (ID-WSF), 2007. http://www.projectliberty.org/resources/specifications.php.

[9] M. Ion, A. Danzi, H. Koshutanski, and L. Telesca. A peer-to-peer multidimensional trust model for digital ecosystems. In *Proceedings of IEEE International Conference on Business Ecosystems and Technologies (IEEE-DEST'08)*. IEEE press, February 2008.

[10] M. Ion, H. Koshutanski, V. Hoyer, and L. Telesca. Rating agencies interoperation for peer-to-peer online transactions. In *In proceedings of the 2nd International Conference on Emerging Security Information, Systems and Technologies (SECURWARE'08)*, pages 173–178. IEEE Computer Society, 2008.

[11] Kerberos. The kerberos network authentication service (v5), 2005. IETF RFC 4120.

[12] H. Koshutanski, M. Ion, and L. Telesca. A distributed identity management model for digital ecosystems. In *Proceedings of the 1st International Conference on Emerging Security Information, Systems and Technologies (SECURWARE'07)*, Valencia, Spain, October 2007. IEEE press.

[13] T. Loukopoulos and I. Ahmad. Static and adaptive distributed data replication using genetic algorithms. *Journal of Parallel and Distributed Computing*, 64(11):1270–1285, 2004.

[14] F. Nachira, P. Dini, A. Nicolai, M. Le Louarn, and L. Rivera Leon, editors. *Digital Business Ecosystems*. European Commission, 2007. http://www.digital-ecosystems.org/book/de-book2007.html.

[15] Francesco Nachira, Paolo Dini, and Andrea Nicolai. *Digital Business Ecosystems*, chapter A Network of Digital Business Ecosystems for Europe: Roots, Processes and Perspectives. European Commission, 2007. http://www.digital-ecosystems.org/book/de-book2007.html.

[16] SAML. Security Assertion Markup Language (SAML), 2005. http://www.oasis-open.org/committees/security.

[17] SPKI. SPKI certificate theory, 1999. IETF RFC 2693.

[18] S. Tuecke, V. Welch, D. Engert, L. Perlman, and M. Thompson. RFC3820: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, 2004. http://www.ietf.org/rfc/rfc3820.txt.

[19] W3C. RDF Primer, W3C Recommendation, 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[20] W3C. RDF/XML Syntax Specification, W3C Recommendation, 2004. http://www.w3.org/TR/rdf-syntax-grammar/.

[21] W3C. Representing vCard Objects in RDF/XML, 2004. http://www.w3.org/TR/vcard-rdf.

[22] O. Wolfson, S. Jajodia, and Y. Huang. An adaptive data replication algorithm. *ACM Transactions on Database Systems*, 22(2):255–314, 1997.

[23] WS-Federation. Web Services Federation Language (WS-Federation), 2006. http://www-106.ibm.com/developerworks/webservices/library/ws-fed.

[24] WS-Policy. Web Services Policy Framework (WS-Policy), 2004. http://www-106.ibm.com/developerworks/library/specification/ws-polfram.

[25] WS-Trust. Web Services Trust Language (WS-Trust), 2005. http://www-106.ibm.com/developerworks/library/specification/ws-trust.

[26] X.509. The directory: Public-key and attribute certificate frameworks, 2005. ITU-T Recommendation X.509:2005 | ISO/IEC 9594-8:2005.

[27] XACML. eXtensible Access Control Markup Language (XACML), 2005. http://www.oasis-open.org/committees/xacml.