Distributed Identity Management Model for Digital Ecosystems

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

Mihaela Ion
CREATE-NET
Via Solteri 38, Trento (Italy)
Email: mihaela.ion@create-net.org

Luigi Telesca
CREATE-NET
Via Solteri 38, Trento (Italy)
Email: luigi.telesca@create-net.org

***Abstract*— Digital Ecosystems is the new paradigm for dynamic IT business integration. A Digital Ecosystem consists of institutions that compete, collaborate, and form stable or unstable federations. Such a dynamic environment becomes a bottleneck for identity management solutions. Existing solutions 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 and Medium-size Enterprises (SMEs).**

This paper presents an identity management model for automated processing of identity information between distributed ecosystem partners. The model emphasizes on its practical, clear and easy to deploy framework. The model is based on the new OASIS SAML 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 SMEs to use and enhance their current identity technology with a practical and easy to implement identity management solution that scales up to the dynamic and distributed nature of digital ecosystems.

I. Introduction

A Digital Ecosystem (DE) consists of diverse institutions which sometimes compete against each other and other times collaborate and form stable or unstable federations. Digital ecosystems are interconnected by a network to form a complex and dynamic environment.

Managing identities in such a distributed system poses many challenges. First of all, institutions use different types of certificates and identity technologies (e.g. X.509, SPKI and Kerberos) which are not always compatible with each other. Secondly, users often need to access applications, services or a composition of services located on different administrative domains. Finally, because of the dynamic nature of the environment, federating and sharing of identities becomes a complex task. A pure federating approach is viable only when there is a stable relation. In Digital Ecosystems, federation does not scale up because of the unstable and ad-hoc coalitions. Institutions cooperate some times and are rival to each other other times.

We need to provide ways for exchanging identity information between companies independent of the standards they use and to share user identity between different domains which could be federated or have no direct trust.

WS-Policy [14], WS-Trust [15] and WS-Federation [13] cover a wide range of requirements and at the same time are difficult to suit immediately for small and medium size enterprises (SMEs). What SMEs in DEs need is a targeted model that is easy to understand and straightforward to implement and put in practice. Existing standards are heavy and difficult to understand and implement and therefore suitable for large enterprises.

A. Paper Contribution

Our model 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 OASIS Security Assertion Markup Language (SAML v2.0) standard for providing proper identification. SAML faces interoperability on the message level and helps to automate and converge when 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.

II. Available Standards and Approaches

There are several technologies and standards used for managing distributed identities. The most mature and widely deployed solutions for federated identity are the SAML and Liberty Alliance standards. SAML [9], 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. It also provides standards for single sign-on of identity management. 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) [3] and Web Services Framework (ID-WSF) [4]. 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.

WS-Trust and WS-Federation define standards for federating identities by allowing and brokering trust of identities, attributes and authentication between participating Web services. However, though partially inspired by the two standards, our model aims at simplifying them and targeting environments composed of unstable coalitions which is often the case for SMEs.

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 [17]. The entity issuing the SAML assertions is called the asserting party or identity provider (IdP). The party which makes use of the assertions made by the IdP to control access and provide services is called relaying party or service provider (SP). A SP trusts an IdP if it relies on assertions issued by the IdP.

Identity federation means sharing of identity information between domains who have a trust relationship or agreement. SAML and Liberty Alliance define standards for federating identities and single sign-on (SSO). SSO allows users to get identified once and then move across domains within a federation with this identity.

The SSO use case requires a Service Provider (SP) and an Identity Provider (IdP) which have a trust relationship (SP relies on assertions from IdP). The SSO can be initiated by any of the two parties. Below we describe the SP-initiated SSO case.

1) The user tries to access a service on SP without any login information.
2) The user is not recognized and gets redirected to the IdP.
3) The user signs on by providing the required credentials to the IdP (e.g. username and password).
4) IdP provides a SAML web SSO assertion for the user's federated identity back to SP.
5) The user is identified at SP and can access the service. The redirection is done automatically and the user is not aware of being forwarded to a different domain.

X.509[16] and SPKI[10] are the main standards for identity certification used by IT business companies now-a-days. The standards are not compatible since they were designed to address different issues.

X.509 is the widely used standard for Public Key Infrastructure (PKI). It specifies a standard format for certificates and a certification path validation algorithm. An X.509 certificate binds a public key to a global name. A Certification Authority (CA) is a trusted third party which creates and signs off-line digital certificates. X.509 assumes a hierarchy of CAs. X.509 was designed for providing authentication and attribute information.

SPKI provides a standard for using digital certificates to provide authorization and authentication for using resources in a peer-to-peer fashion. SPKI allows principals to define local names and their namespaces can be linked to define trust (also called Web-of-trust). There is no hierarchical global infrastructure in contrast to X.509. Each public key (entity) is a certificate authority and can issue certificates on the same basis as any other principal thus making it suitable for decentralized and unstable coalitions.

OpenID[1] is a decentralized framework for digital identity. The underlying idea is that users can identify themselves on the web like websites do with URIs. OpenID allows a username/password login. The username is the personal URI and the password is safely stored on the OpenID Provider. To login to an OpenID-enabled website, 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 website with the required identity information to logon.

III. The Basic Identity Management Model

We start by defining the key entities in 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) *Credential Provider (CP)*: 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.
5) *Federation*: Stable coalition of companies which have a cooperative relation.

We propose an identity management model for decentralized peer-to-peer ecosystem domains. All users are considered equal and there is no hierarchy of ecosystems. Any peer can be a Credential Provider or a Service Provider, or both. Each user can issue a certificate to other users. Each user has a list of trusted Credential Providers. Each Credential Provider has a list of acceptable security tokens. A Credential Provider issues certificates to users either *(i)* based on secure tokens issued by the provider itself or *(ii)* based on trusted secure tokens (from Credential Providers with whom it has trust relationships) or *(iii)* based on user registration information.

A. Managing user identities

In a system 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 username and password, ad hoc secure tokens or adoption of OpenID for Web-based access.

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

[1]http://openid.net

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. Section IV describes in details the use cases regarding the model.

After joining a Digital Ecosystem, users (partners) obtain a variety of certificate tokens issued (transformed) by Credential Providers for particular business needs. Possible secure tokens considered in the system are X.509, SPKI, Kerberos and SAML identity assertions. However, partners that already have OpenID or ad hoc identity tokens (or username/password) can use them in the system but only for the purpose of providing identity information to CPs that are to certify partners' identity. All the subsequent certificates issued by CPs in a DE are bound to one of the standards mentioned above. The reason for that is to unify and simplify identity management between DEs to well-defined identity standards.

Each CP has the responsibility to provide proper pseudonimity to end users. Typically a CP either provides a user pseudonym on its own or allows users to define it and then certifies the pseudonym in a trusted secure token to a Service Provider. We note that a SP explicitly asks a CP to reveal user identity in case of user misbehavior. So, each CP maintains a database mapping user's pseudonimity with user's real identity.

B. Managing user profile

Having multiple identity certificates issued by different CP, it becomes difficult for a user to manage and allocate 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, username/passwords and public/private key pairs. For that purpose we adopted the use of *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 a CP by username/password or via LDAP[2] storage etc).

Here, 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 disclosure (no one except the owner of the file) and at the same time must be 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.

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. So, we decided to provide a sticky policy with each profile that encodes who can use the profile and under what conditions. The sticky policy is optional and if not explicitly specified it has the default value of read and write permissions only for the profile's owner. A good solution for a sticky policy model is the use of Access Control Lists (ACL).

As we mentioned before, we adopted the concept of peer-to-peer trusted network to replicate and provide service availability when locating and loading user profiles. As of time being, we leave the problem of how to establish a proper methodology for data replication specific to a particular application scenario and assume that there are available models, e.g. [5], [12], that address this issue.

A user is required to remember only a username and password in order to login into a DE. The username and password are obtained once when the user initially registers to a DE. Then, whenever the user logs in another (or the same) Digital Ecosystem by presenting its name and password (just authentication), the DE takes the responsibility to allocate and retrieve the encrypted user profile. Each DE has its predefined trusted node(s) which stores information on DE users and addresses of other trusted nodes residing in other DEs.

When a user starts a new session, his profile is downloaded on a secure memory (e.g. browser s-box) in its Web browser or local client and then decrypted in the memory. Once decrypted the profile is ready to be used and processed by the Web browser client or the local client. At the end of the session, the user's profile is encrypted and updated on the associated trusted node (peer) and then replicated on the 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 the other trusted peers in order to provide availability and actualization if shared among multiple users.

C. Managing passwords and secure tokens

Each user has a pair of private/public keys used in all certificates issued by different authorities[3]. User identity information is stored in a user profile that is encrypted and replicated on trusted peers. The user profile contains information about available certificates, public/secret key pair and user authentication information needed to access and obtain secure tokens.

User identity information obtained outside DEs should be updated in the user profile so that it can be used when the user does business interactions with partners within DEs. Especially, when a user first registers to a DE and creates its initial profile, it decides whether to import the already available identity information. However, a user can start from

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

[3]For the sake of simplicity of the framework we assume one key pair. However, multiple key pairs are also possible providing that there is a proper mapping between available certificates and used key pairs in the user profile.

no identity information and collect it on a step-by-step basis when interacting with service providers (and their CP).

After each interaction with a CP, the user's client (web or local) automatically records the information on the new identity token for subsequent use. The information stored depends on the settings the user specified and the CP's policy. For example, an identity token may only be issued to be presented to a SP without being stored on the client profile. In this case, we record only the information on how to obtain a new token (e.g., by presenting another token or by username/password). In other cases, the identity token could be stored on a secure LDAP server trusted by the CP who issued the token so that it can be automatically obtained by a user via authentication to the server. In any case, after each interaction with a CP, the client profile stores the necessary information on what identity, what validity and how to obtain such.

The last issue left to be examined is how to keep user profiles encrypted. A user profile is encrypted with a long master password (usually key phrase) that is never stored and known only to the user. The master password must 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 best security when using a DE.

D. Use of SAML in the model

Having designed the identity model, we faced the problem of incompatibility of different identity standards. X.509 and SPKI, the certificate standards most widely cited in the literature, are designed to be different. We had to provide ways for a client identified with one standard to be able to use his identity information when communicating with a SP using another standard.

Another issue we had to take into account was that SMEs often adopt their own (ad hoc) certificate tokens or different identity mechanisms (such as OpenID) to manage identities of their employees.

To cope with this wide range of identity mechanisms we have to make the following assumption. Each SP adopts the identity standard best suiting its needs but its related CPs should support by default the SAML standard (especially v2.0). It means that any SME could preserve its existing identity management infrastructure but should enhance its trusted CP with the ability to understand SAML. Furthermore, each CP must be able to issue SAML assertions derived (transformed) from any of the standards the CP supports. Refer to the example given below.

With the new version of SAML, the standard allows to express identity information (in SAML assertions) within a context of any type of authentication (e.g. X.509, SPKI, Kerberos tickets, username/password etc). Thus, the model uses SAML assertions to bridge different identification information and standards.

For example a CP that supports X.509 and username and password authentication to be functional/compatible in our framework it has to also support the conversion:
– X.509 $\leftrightarrow$ SAML assertion
– Username & password $\leftrightarrow$ SAML assertion

SAML assertions are used when accessing or negotiating with different ecosystem domains. Once we unify the identity and authorization representations between CPs we can accommodate any identity model/requirements particular to a service provider.

Example *(X.509 and SPKI identity exchange)*
Let us suppose that a SP_1 only accepts X.509-based authentication to identify entities and that SP_1 trusts CP_1 to validate X.509 tokens. Now, if a user has a SPKI certificate issued by CP_2 that has a trust relationships with CP_1 then the user is able to identify itself to SP_1 by use of our model.

To do so, the user has to contact his CP_2 and request for a SPKI to SAML assertion transformation in order to identify itself to CP_1. Since CP_1 has trust in CP_2 for proper entity identification CP_1 accepts the SAML assertion and issues (transforms it to) an X.509 identity certificate that is forwarded to SP_1. SP_1 trusts its CP_1 for identifying entities and provides access to the desired service.

E. The Identity Management Model

So far, we have presented all we need to state our identity management model. Figure 1 shows the basic model and workflow of messages between the main actors. The message flow of the model is the following:
1. An entity (web browser user or local client) makes a request

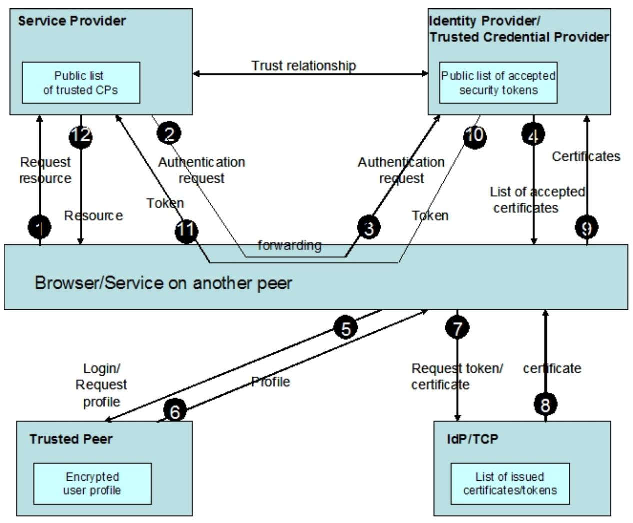
Service Provider
Public list of trusted CPs
Trust relationship
Identity Provider/ Trusted Credential Provider
Public list of accepted security tokens
Request resource
Resource
Authentication request
Token
forwarding
Authentication request
Token
List of accepted certificates
Certificates
1
2
3
4
5
6
7
8
9
10
11
12
Browser/Service on another peer
Login/ Request profile
Profile
Request token/ certificate
certificate
Trusted Peer
Encrypted user profile
IdP/TCP
List of issued certificates/tokens

Fig. 1. The Basic Identity Management Model

to a Service Provider.
2-3. The Service Provider redirects the user to a Trusted Credential Provider (TCP).
4. The user has no credentials issued by the TCP. The TCP sends a list of accepted certificates with a list of its trusted CPs (TCPs) to the user.
5. The user requests its profile from a trusted peer storing it and uses username/password for authentication. Information

for ecosystem trusted peers is obtained (possibly publicly available) when users join the ecosystem.
6. The trusted peer sends the encrypted user profile.
7. After the profile is decrypted, the user checks if it has the right credentials, i.e. processes his profile for matching of credentials (issued by any of the TCPs obtained in step 4). If no credential is matched then the user has to register to the TCP to obtain an identity token. If one of the credentials requested in step 4 is found, then the user extracts it either from the profile or requests it from the remote TCP that issued it. Important issue here is that the user identifies whether the certificates match by type and TCP name or only by TCP name. The first case requires that the user just presents the certificate as it is, while the latter case requires that the user requests the TCP for credential transformation.
8. The TCP authenticates the user and then returns either the requested certificate or its transformation to a SAML assertion.
9. The user forwards the certificate/SAML assertion to the TCP.
10. The TCP verifies and validates the certificate and issues (transforms if needed) a new one that is to be forwarded to the SP.
11. The user is redirected to the Service Provider which accepts tokens from its TCP.
12. The Service Provider verifies the new certificate and provides the requested resource to the user.

We note that in step 10, TCP transforms the certificate presented in step 9 to an identity token type acceptable by the SP. Even if the identity token type accepted by the SP is the same with the one presented by the user, the TCP still has to issue a new identity token (even of the same type), but signed by the TCP (thus validating the user's identity).

The only case in which the TCP does not issue a new token is, for example, when the user has been already in contact with the SP and presents the same identity token that has been issued by the TCP last time. In this case, the TCP does only certificate verification and validation.

F. Available technology providers

Having shown the basic model and its functionalities, the next step is to identify the technology providers and possible component implementations supporting our framework.

Below we list the technology implementations with a brief description of their use in the framework.

– OpenSAML [7]: the most widely used SAML v1.1 technology implementation. It is available in Java and C++ libraries.
– OpenPMI [6]: an open source project targeting an open platform for providing PKI/PMI based solutions. Especially interesting for use is XSAML library for conversion of X.509 to SAML assertions and vice versa,
– OpenSSO [8]: provides widely use of the SSO concept based on SAML,
– Sun Java System Federation Manager [2] and Sun Java System Access Manager [1]: useful tools for digital identity generation, management and sharing (based on SSO model) within a circle of trusted partners and across federations. It is based on the SAML and Liberty standards.

IV. Basic use cases and scenarios

There are several basic use cases that the model needs to consider:

- Accessing a resource in the same ecosystem: corresponds to the SSO use case.
- Accessing a resource in a different ecosystem: requires a new login if the user is recognized in the new ecosystem.
- Transforming secure tokens from one type to another: the user is not recognized in the new ecosystem.
- Services composition: one service depends on other services.

A. Transforming of secure tokens from one type to another

An identity can be trusted in one domain of the federation and not in another one. To access resources in another domain, an identity mapping is needed. Identity mapping means converting a digital identity from one domain to a digital identity valid in another domain. The conversion is made by a credential provider that trusts the starting domain and is trusted by the ending domain (the end domain accepts its assertions).

Example *(Identification during negotiation)*
Figure 2 shows an example of identification in interconnected digital ecosystems for the purpose of a business negotiation. CP1 is the trusted credential provider of B1 and N which are in the same ecosystem and CP2 is the trusted credential provider of B2 which belong to a different ecosystem. N is the entity (peer) that hosts the business negotiation. The identification between B1 and N is done using SSO. B1 is simply redirected to CP1 to login. For the identification between N and B2, a transformation of credentials is needed from CP2 credentials to CP1 credentials.
1. N starts a negotiation

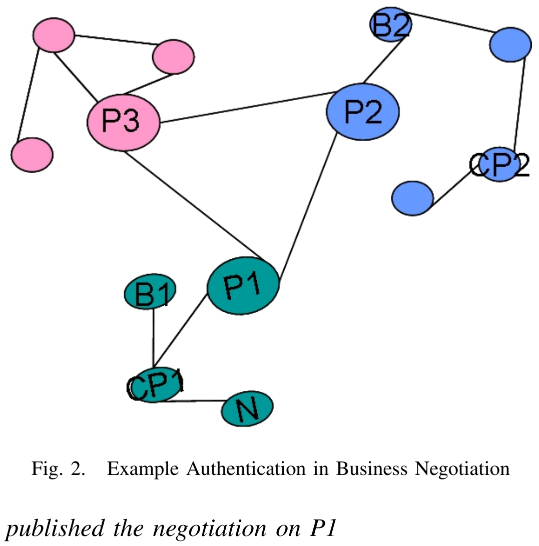

Fig. 2. Example Authentication in Business Negotiation

2. N published the negotiation on P1
3. B2 finds out about the negotiation through P2

4. B1 and B2 bids
5. Mutual authentication of B1 and N: SSO using CP1
6. Mutual authentication of B2 and N: transforming of CP2 credentials to CP1 credentials and vice versa.

V. Model extension to service composition

Digital Ecosystems allow companies to cooperate with each other and compose services. An important requirement for an identity management model for DEs 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.

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 [11] 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 proxy certificate.

There are three important requirements specified in the policy of a proxy certificate that reflect our identity model. 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 by the main service (e.g. aggregation of aggregation) should not consider the PC as a valid identity certificate (on behalf of a client).

To solve the issue of complex aggregation of services that aggregate other services we propose the second requirement, that is the level of aggregation. The purpose of this level of aggregation is to restrict the use of a PC in a chain of service aggregations.

The third policy requirement is the validity period of the 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.

The level of aggregation should be interpreted as not to use the PC as deep as the level is, but to indicate whether a new PC could be obtained from the original one. That is, 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 signs a new PC to the second SP but with level of aggregation decreased with one unit and scope of PC the scope of the second SP aggregated service. Additionally, the validity period of the new PC is the remaining validity period of the PC that signs it.

Thus, the second SP can use the new PC only for the sake of execution of its aggregated service as requested by the first SP and within the validity time frame as specified originally by the user.

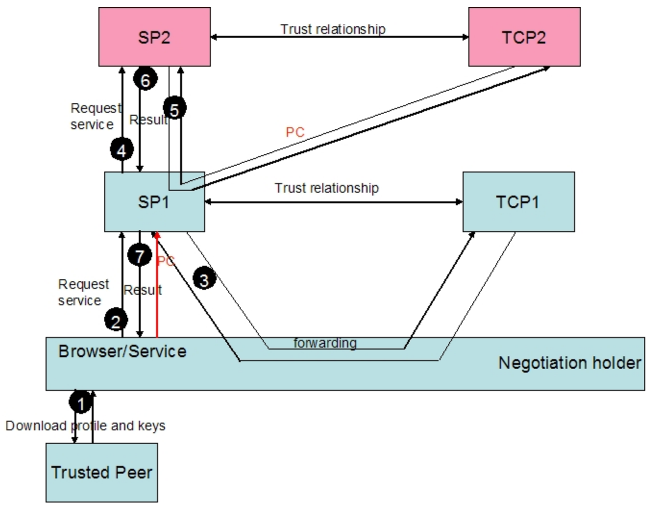

Fig. 3. The Extended Identity Management Model

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 that stores it.
2. The user requests a composite service from SP1.
3. The user is redirected to TCP1 to logon (SSO use case). SP1 indicates that the requested service is an aggregation of services together with a list of the services to be used. The user identifies itself to the TCP1 and then issues a PC to SP1 with policy that the proxy certificate will only be used for the scope of this service request and specified level of further aggregations.
4. SP1 requests a service to SP2.
5. SP2 redirects SP1 to TCP2 for authentication. SP1 authenticates the user with TCP2 using the proxy certificate (the PC obtained in step 3).
6. SP2 runs the service and provides the result to SP1.
7. SP1 completes the service execution and provides the result to the user.

The extended model scheme can be (recursively) applied in case SP2 needs to contact SP3 as next level aggregated service provider. Then SP2 takes the role of SP1 in the model.

VI. Conclusions and Future Work

In this paper we have presented a new identity management model for Digital Ecosystems. The model bridges main identity standards by using SAML as a unified message-level protocol for quering and obtaining authentication attributes. By using SAML we are able to automate the process of identifying entities in a distributed environment. We adopted the use of user profiles to keep an abstract view of user's identity information such as certificates, username/passwords, public/private keys etc. The user profile is encrypted and replicated on trusted peers. To scale to service compositions we adopted the use of proxy certificates with two main policy settings limiting service scope and level of aggregation.

We have presented the extension of the model to address the problem of identity management and control for service compositions. The extended model provides the end-user with the ability to control the use of its identity information in case of service aggregations.

Ongoing work is providing a practical implementation of the described model. Several steps are needed to be taken into account. First one is defining a standard semantics of user profiles allowing for automated processing and querying of identity information. Second step is defining interfaces and APIs for credential providers and local/web-based user clients to allow automatic queries and transformations of different identity tokens to SAML assertions. The second step will be based on SSOs.

The last main step is enhancing the model with management of authorization information between distributed ecosystem partners. Especially, we want to target the two standards, X.509 and SPKI, that play a major role in certifying authorization decisions.

Acknowledgment

This work was partly supported by the projects: 038978 EU-MarieCurie-EIF-iAccess, 034744 EU-INFSO-IST ONE and 034824 EU-INFSO-IST OPAALS.

References

[1] Sun Java System Access Manager. www.sun.com/software/products/access_mgr.

[2] Sun Java System Federation Manager. www.sun.com/software/products/federation_mgr/index.xml.

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

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

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

[6] OpenPMI. Open Privilege Management Infrastructure (OpenPMI). openpmi.sourceforge.net.

[7] OpenSAML. Open Source SAML Implementation. www.opensaml.org.

[8] OpenSSO. Open Web SSO (OpenSSO). https://opensso.dev.java.net/.

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

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

[11] Steven Tuecke, Von Welch, Doug Engert, Laura Perlman, and Mary Thompson. RFC3820: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, 2004. www.ietf.org/rfc/rfc3820.txt.

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

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

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

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

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

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