OPAALS PROJECT

Contract n° IST-034824

WP4: Workpackage 4: Distributed Accountability, Identity, and Trust

Del 4.5 – Final Report

Project funded by the European Community under the "Information Society Technology" Programme

Contract Number: IST-034824

Project Acronym: OPAALS

Deliverable N°: 4.5, Workpackage 4 Final Report

Due date: M18

Delivery Date: M33

Short Description: This deliverable provides a summary of the work done in Workpackage 4 during Phase I of OPAALS. No new additions to the tasks are introduced here. Further work on these tasks will be performed in Workpackage 3 in Phase II of the project and further developments will be reported there.

Author: Paul Malone (WIT), Mark McLaughlin(WIT), Jimmy McGibney (WIT), Brendan Jennings(WIT), Mihaela Ion (CN), Jesus Gabaldon (TI), Amir Razavi (UniS)

Partners contributed: WIT, CreateNET, TechIdeas, UniS

Made available to: Public

Versioning

Version	Date	Name, organisation
0.1	06/07/08	Paul Malone (WIT)
0.2	08/09/08	Mihaela Ion (CN), Jesus Gabaldon (TI), Amir Razavi (UniS)
0.3	27/09/08	Jimmy McGibney (WIT), Brendan Jennings(WIT)
0.4	21/11/08	Paul Malone (WIT), Mark McLaughlin(WIT)
0.5	18/03/09	Paul Malone(WIT)
1	24/0309	Paul Malone(WIT)

Quality check

Internal Reviewers: Gary Gaughan (UL), Sotiris Moschoyiannis(Surrey)

Dependences:

Achievements	Our identity model aims at automating the process of identification between ecosystem partners. We focus on practical solutions which are clear and easy to implement. The model is based on the new SAML (v2.0) [1] standard for providing proper identification. SAML provides interoperability on the message level and helps to automate and converge when the technologies are not compatible. We manage 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 decentralised manner in the peer-to-peer network. We provide a model for reputation based trust in distributed environments based on an overlay of trust managers. Though we primarily took a computer science approach, in order to solve concrete issues related to the open distributed platform of the DE, we also relied on concepts and methods from ecology, economy and sociology. The model is suitable for the decentralised and dynamic nature of DEs. We base our model on standards and consider scalability, availability and ease of putting into practice as the most important characteristics of the model. The model has four main components: distributed identity management, peer-to-peer reputation, trusted rating agencies, and learning. These components address different security, social and evolutionary aspects of the DE and create a complete model. The identity management model enables creating a reputation framework by providing ways for securely identifying entities. The reputation framework has two main components: a peer-to-peer reputation model and a decentralised trusted rating agencies model. DEs evolve in time in order to respond to changing conditions. We describe a decentralised Accountability model is designed which considers service composition, scalability, security as well as contract exchange. The model under development is a combination of the PeerMint model published by Hausheer and and Stiller [2] combined with the SOA Accountability Architecture desigend by Zhang et al [3]. Our model takes the concept of the Accounting Authourity introduced by Zhang and to deploy this as a distributed service and combine this with the distributed PeerMint model. The contribution of the collaborative knowledge sharing work performed provides a framework for collaborative knowledge sharing, novel tools for collaborative knowledge sharing and new algorithms of distributed knowledge sharing for digital ecosystems

Work Packages	The identity work described in this document and in deliverable D4.1 [4] will be augmented with further work in Workpackage 3. This will take the form of further refinement of the Identity Operations and a close integration with the Trust Model described in [5]. Issues such as a possible requirement for global uniqueness versus the lack of single point of control need also to be addressed. Also implementation plans for deploying the Identity Model in the planned infrastructure. Currently the proposed infrastructure will be based on JXTA[1]. The Trust model described in [5] and in this document rely on experience reports generated internally within a digital ecosystem in the creation of trust ratings and also on those trust ratings being shared among Trust Managers to create a community driven reputation based trust system. In the next period of the work (in Workpackage 3) we will examine the idea of institutional trust, i.e. ratings made about entities by external trusted institutions. An example of this type of institutional trust would be a driving licence verified by a state Department of Transport. In addition to this we will address integration points for our trust overlay model in other aspects of the peer-to-peer network being developed in Workpackage 3. We will look at how data generated by the rating agencies and distributed transaction facility can be used in creating the experience reports which provide input into the trust update process. The Accountability model described in [6] and in this document provides most of the functionality required to allow for distributed accountability in peer-to-peer networks where dynamic service composition is the enabler for multi-peer collaboration. However, some requirements are not satisfied fully by this model, namely security issues. Privacy is a security aspect which also needs to be addressed in a future evolution of this model. In the next evolution of this model an access control mechanism needs to be added to overcome this issue. Furthermore, decisions on hashing methods, routing tables, leaf-sets and selection of account and session holders need to further considered as the network design decisions emerge from Workpackage 3 and will be reported in Deliverable D3.8. The techniques of data mining and FCA have many potential applications for collaborative knowledge sharing of the ecosystem. We will develop suitable models and algorithms to analyze, extract, acquire, deliver and share knowledge of the ecosystem. This development can be integrated with other WPs in Phase 2 of the project such as WP5 (Integration with the Digital Ecosystem platform) and WP10 (Sustainable Research Community Building in the Open Knowledge).
Partners	IPTI (platform development), TechIdeas (integration), Surrey (p2p

[1] JXTA Community Project, https://jxta.dev.java.net/

	platform)
Domains	Identity, Trust, Accountability, Knowledge Sharing, Distributed Computing, Security
Targets	Other Researchers, System Implementers, SMEs, Public Administrators, Social Scientists
Publications*	The identity model was published in Koshutanski, H., Ion, M. and Telesca, L., 2007, *A distributed identity management model for digital ecosystems*, in Proceedings of International Conference on Emerging Security Information, Systems and Technologies (SECURWARE'07), IEEE Press The trust overlay Model was published in McGibney, J. & Botvich, D., 2007. A Trust Overlay Architecture and Protocol for Enhanced Protection against Spam. In Proceedings of *The Second International Conference on Availability, Reliability and Security*. IEEE Computer Society, pp. 749-756. and McGibney, J. & Botvich, D., 2008. A trust based system for enhanced spam filtering. *Journal of Software*, 3(5), 55-64. The accountability model published: Malone, P. and Jennings, B., 2008, Distributed Accountability Model for Digital Ecosystems, *2nd IEEE International Conference on Digital Ecosystems and Technologies*, Phitsanulok, Thailand, February 2008. The collaborative knowledge algorithms: Fu, H., Scalable Conceptual Hierarchy Based Algorithm for Knowledge Sharing in Digital Ecosystem, *2nd IEEE International Conference on Digital Ecosystems and Technologies*, Phitsanulok, Thailand, February 2008.
Outstanding features*	The use of a modeling framework to build generic identity protocols (operations), with the possibility of multiple, re-usable bindings and integration with the trust, represents a significant advance in the state of the art beyond a SAML-inspired, identity federation approach. The trust overlay approach represents an incremental advance to the state of the art in trust evaluation for entity-centric distributed systems. The work provides an incremental change in the state of the art by providing a model and protocol to enable a service composition capable accountability framework to operate in a distributed and private manner. The algorithms for knowledge sharing provides a similar advancement in the area of knowledge sharing. Each area of the work has been published to communities outside the digital ecosystems community (See Publications above).

Disciplinary domains of authors*	Mark McLaughlin, WIT, Computer Science Mihaela Ion, CN, Computer Science Paul Malone, WIT, Computer Science Jimmy McGibney, WIT, Computer Science Dmitri Botvich, WIT, Computer Science Jesus Gabaldon, TI, Computer Science Brendan Jennings, WIT, Computer Science

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License. To view a copy of this license, visit : http://creativecommons.org/licenses/by-nc-sa/3.0/ or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Table of Contents

List of Figures

1 Introduction

Workpackage 4 was responsible for the creation of models for distributed identity, trust and accountability as well as distributed collaborative knowledge sharing. Each of these tasks have been reported in their respective deliverables. This document is a report of the work performed in those tasks and much of the content contained in this report is derived from previous deliverables from workpackage 4, namely [4] ,[6], [5] and [7]. The work performed in the workpackage will feed into work in Phase 2 via Workpackage 3 with a view to integrating these models more closely with the rest of the work being performed at the peer-to-peer network layer and also with the distributed transactions model.

A Summary of the work on a per-task basis is given in the following chapters.

2 Distributed Identity Model for Digital Ecosystems

Much of the content in this chapter is derived from Deliverable D4.1[4] and contains details of a work published in [8].

A Digital Ecosystem (DE) consists of diverse organisations 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, organisations 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.

Companies co-operate some times and are rivals at 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-Trust[9], WS-Federation[10] and WS-Policy [11] cover a wide range of requirements and at the same time are difficult to suit immediately for small and medium sized 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.

Our model aims at automating the process of identification between ecosystem partners. We focus on practical solutions which are clear and easy to implement. The model is based on the new SAML (v2.0) [1] standard for providing proper identification. SAML provides interoperability on the message level and helps to automate and converge when the technologies are not compatible. We manage 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 decentralised manner in the peer-to-peer network. We refer to the following key entities in our model:

- **User**: any entity that can be identified in the network (peer or web browser user, institution

or person)

- **Service Provider (SP)**: any identifiable entity that has one or more services or resources available to other entities.
- **Credential Provider (CP)**: any entity that is able to provide digitally signed credentials to other entities.
- **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 little (or no) previously established trust relations. Thus the notion of ecosystem comprises co-operative and competitive relations.
- **Federation**: stable coalition of companies which have a co-operative relation.

We propose an identity management model for decentralised 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.

2.1 Multiple user identities

In a system of interconnected digital ecosystems, users and companies use different kinds of certificates obtained from outside the system. Companies have their 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 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.

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 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 misbehaviour. So, each CP maintains a database mapping user's pseudonimity with user's real identity.

2.2 User profile

Having multiple identity certificates issued by different CPs, 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 which 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 unauthorised 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

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

issues we opted for keeping 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).

When a user starts a new session his profile is downloaded on a secure memory (e.g. browser sandbox) 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 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 actualisation if shared among multiple users.

2.3 Passwords and secure tokens

Each user has a pair of private/public keys used in all certificates issued by different authorities. 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. 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/private 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 enters a DE and creates its initial profile, it decides whether 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 Service Providers (and their CPs).

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. Thus, a user has to remember one login information and one master password in order to facilitate best security when using a DE.

2.4 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[1] 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. Depending on the standards used by their SPs, CPs should be able to support different assertions such as:

- X.509 → SAML assertion
- SPKI → SAML assertion
- Kerberos → SAML assertion
- Username & password → SAML assertion

SAML assertions are used when accessing or negotiating with different ecosystem domains. Once we unify the identity and authorisation 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 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 CP_1 for identifying entities and provides access to the desired service.

2.5 Model communication scheme

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.

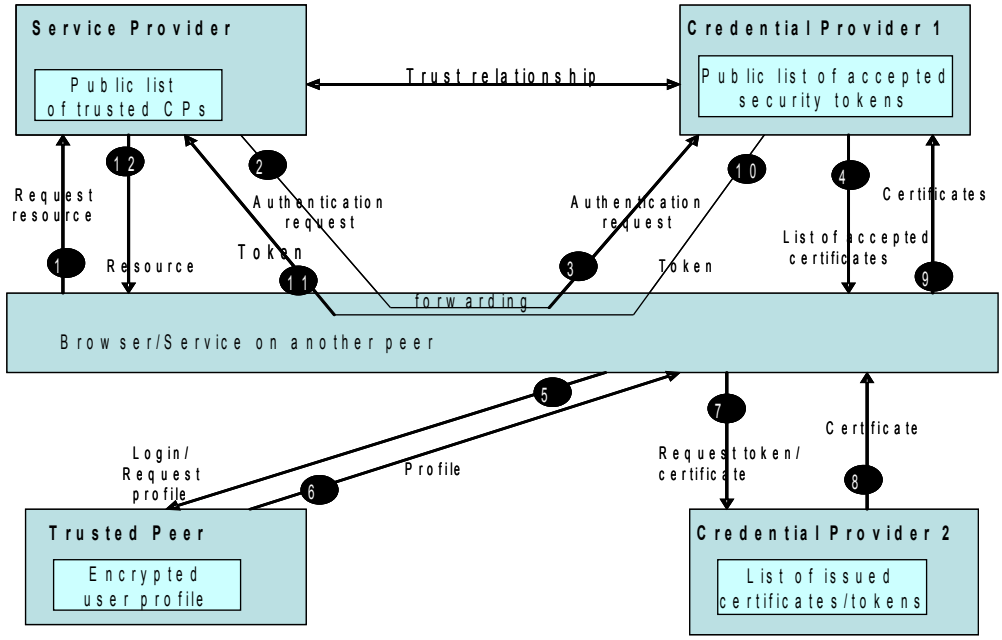

Figure 1: Model communication scheme

The message flow of the model is the following:

- 1. An entity (web browser user or local client) makes a request to a Service Provider.
- 2-3. The Service Provider redirects the user to a trusted Credential Provider (CP1).
- 4. The user has no credentials issued by CP1. CP1 sends a list of accepted certificates with a list of its trusted CPs 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 its profile for matching of credentials (issued by any of the CPs obtained in step 4). If no credential is matched, then the user has to register to CP1 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 CP2 that issued it. Important issue here is that the user identifies whether the certificates match by type and CP name or only by CP 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 CP for credential transformation.

- 8. CP2 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 CP1.
- 10. CP1 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 CP1.
- 12. The Service Provider verifies the new certificate and provides the requested resource to the user.

The only case in which CP1 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 CP1 last time. In this case, CP1 does only certificate verification and validation.

2.6 Model extension to identity propagation in service composition

Digital Ecosystems allow companies to co-operate 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 may need to run the services in the name of the user and as therefore has to authenticate the user to the other providers. To solve this problem we adopted the use of *Proxy Certificate* [12] that the client issues to the provider of the composite service.

A Proxy Certificates 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 use a second policy: 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.

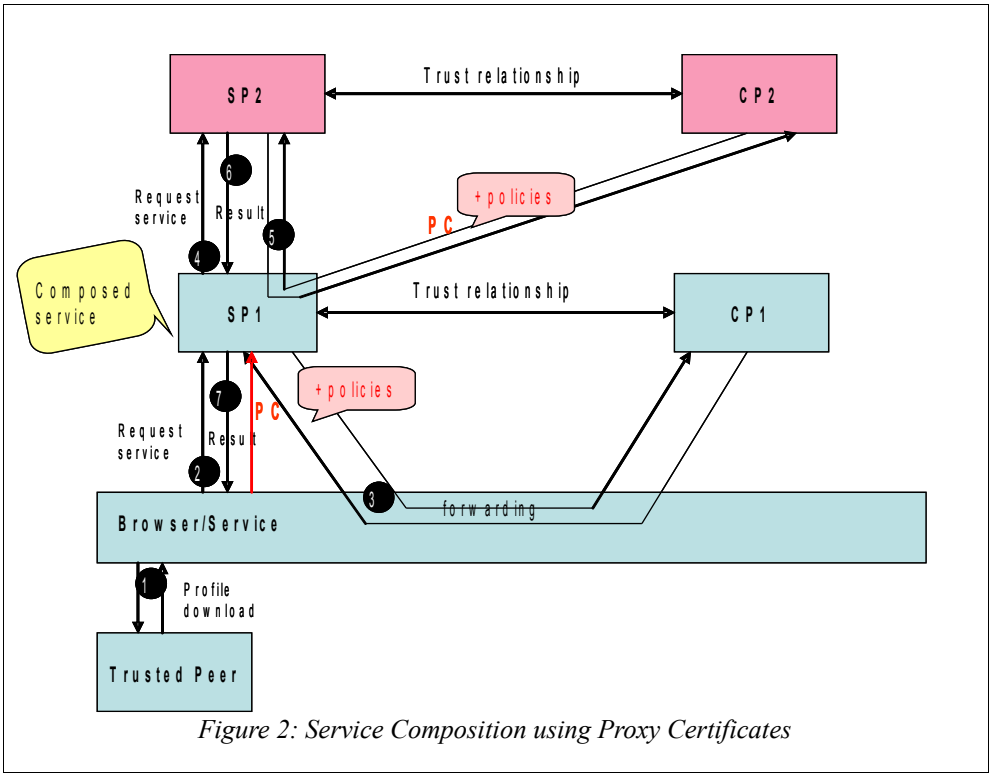

Figure 2: Service Composition using Proxy Certificates

Figure 2 shows the extended identity model for service composition. 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 CP1 to login (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 CP1 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 CP2 for authentication. SP1 authenticates the user with CP2 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.

2.7 Model Implementation

We produced an implementation that incorporates the core functionality of the model. We adopted a flexible approach to design, that would allow us to make significant changes to the comunication scheme, or protocol, used to establish identity. We have endeavoured to make our design agnostic of specific technologies or underlying protocols, where possible, in order to make our implementation as broadly useful as possible.

2.7.1 Technologies used and design decisions

- Java is our choice of programming language. Java is widely used, has a rich set of class libraries, and is well suited to network applications.
- SAML v2.0 is the standard we use to communicate identity information between the entities in our DE. The second version of the protocol is more flexible and powerful than the first one. The SAML Single Sign-On (SSO) profile, using a combination of the HTTP GET and POST redirect bindings proved a good fit for our model.
- OpenSAML v2.0 beta (Java) is the SAML implementation we are currently using. Since SAML v2.0 is a new standard, implementations are naturally at an immature state. However, a full OpenSAML v2.0 release is due to be released in the near future.
- Although the model and implementation are agnostic of network protocols, we focussed on directly supporting thin-client (i.e. browser based) user agents and web (HTTP) based SPs and CPs. This scenario is highly restrictive in terms of the communication schemes we can use, and is therefore an excellent demonstration of the flexibility of our model and implementation.
- No user profile implementation. A pure SAML based implementation of the model (see below) obviates the strict requirement for a user profile. Therefore we do not give an implementation here.

2.7.2 Refining the model

The identity model is flexible and general enough to allow for many potential implementations. It is

non-prescriptive in terms of the authentication mechanisms supported and the format of the security tokens that are passed between entities in the system. We chose to adopt a 'pure SAML' approach in our implementation (see Figure 3 below). Using this approach, all identity information in the DE after the point of authentication is passed in the form of a SAML assertion. We use SAML assertions, passed between trusted entities, in place of arbitrary security tokens to “assert” identity, rather than passing the credentials themselves.

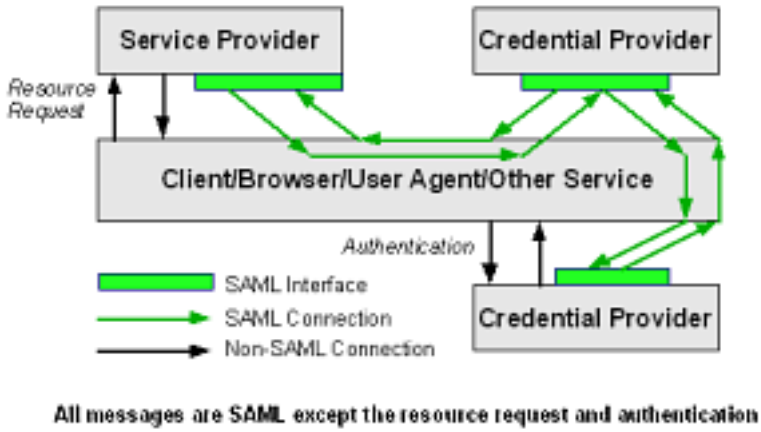

Figure 3: Implementation using a 'pure SAML' approach
œœœ

This gives us the benefits of dealing with a single, simple “token”, rather than pandering to particular SPs and CPs by supporting the arbitrary tokens they might ordinarily use. The cost of adopting this approach is that a “SAML interface” must be exposed by SPs and CPs in order to interpret these SAML assertions and take the appropriate actions. We consider the benefits of a pure SAML approach, in terms of simplicity and extensibility, to be worth this cost. The cost itself can be reduced by providing powerful, generic interfaces for use with SPs and CPs that can be adapted easily to support a wide range of applications.

It is also worth mentioning that it may be of dubious benefit to support the generation and manipulation of a number of different types of security tokens, since there would be a continuous requirement to support new types of tokens, and no strong incentive to converge on a standard.

A significant real world use-case in identity is the scenario where a 'dumb' client such as a browser is the user agent whose identity an SP is attempting to establish. Such clients do not accept incoming requests (only responses to requests) and generally speaking do not have the intelligence

to be anything more than passive entities in the protocol flow used to establish identity. Our model must be flexible enough to support such clients. If dumb clients are supported, we can guarantee that all clients that match or exceed a browser in sophistication will also be supported.

We use HTTP redirects in all instances where manual input from the user is not required. These redirects are illustrated by arrows that pass through, but do not terminate at, the user agent, as illustrated in Figure 5 below.

2.7.3 Actors and Connections

In our implementation, we model DE entities as Actors that participate in the protocol flow that establishes identity. We refer to a message transfer between two Actors as a Connection. We can describe any communication scheme used to establish identity in terms of these two elements. In terms of the identity model, there are three main Actors: the user agent (UA), the service provider (SP) and one or more identity or credential providers (CPs). Examples of Connections used to pass messages between Actors might be, *DirectHTTPConnection* or *GETRedirectHTTPConnection*. Actors and Connections can be extended arbitrarily.

In a real DE, Actors may co-exist on the same node (P2P), or machine. That is to say that an entity in a DE could behave as a UA, an SP and/or a CP at the same time, or at different times and/or in different contexts.

2.7.4 Building Operations from Profiles

We define an Operation as some protocol flow that establishes an identity related function on a network. Examples include Web Single Sign-On, Single Sign-Off, or an attribute request by an SP to a CP concerning a UA. Our identity model is concerned mainly with conducting SSO in a DE, but we use the more general concepts of Operation and Profile to make our model and implementation as flexible and extensible as possible.

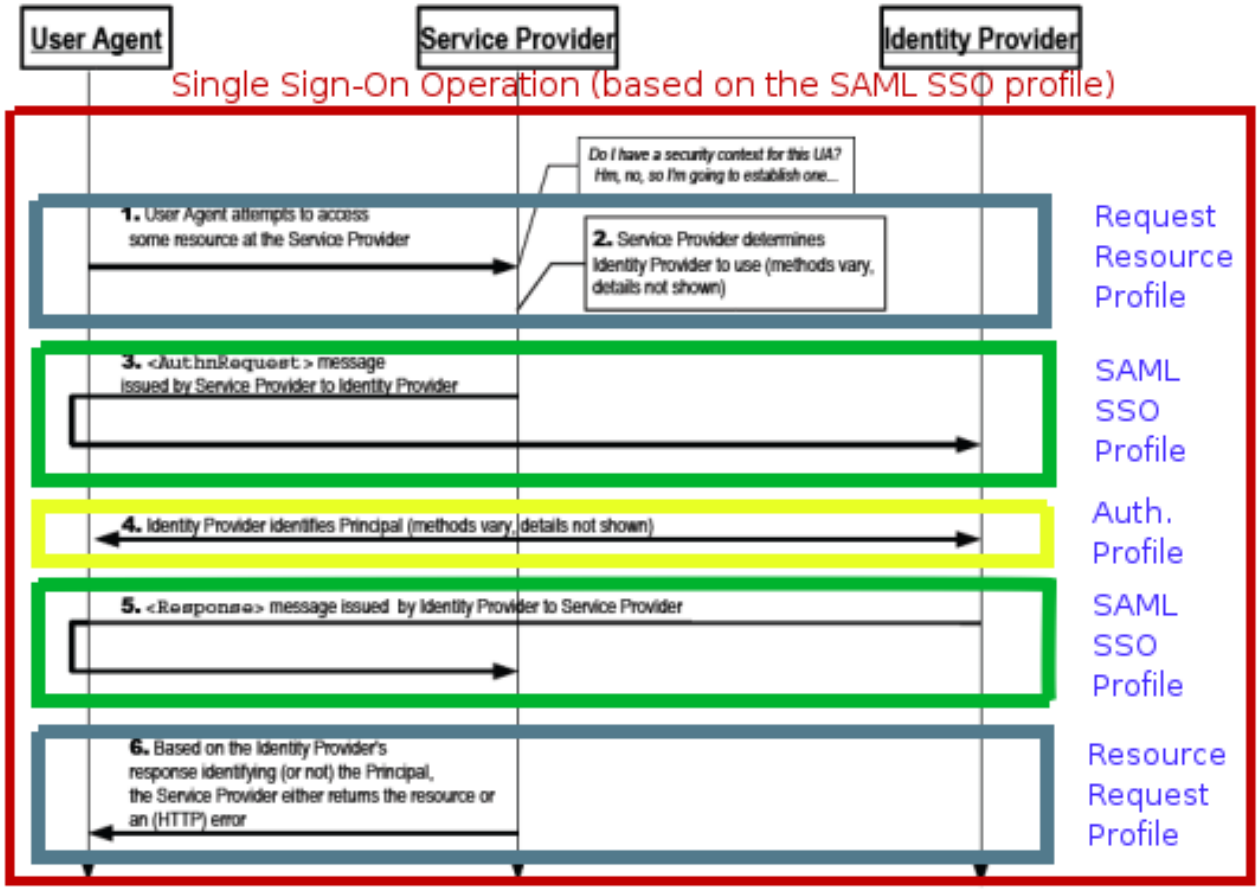

Figure 4.: Single Sign-On Operation based on the SAML SSO profile

Profiles are analogous to, and inspired by, SAML profiles. SAML profiles specify a skeleton protocol flow for identity operations. In our model, Profiles specify portions of the overall protocol flow specified by an Operation. Therefore an Operation is built from a number of Profiles. Profiles are themselves build from Connections. And example of an Operation is given in Figure 4 below. Some of the Profiles in this Operation specify all their Connections, such as the SAML profile (since this Figure is derived from the SAML v2.0 profiles document), while other Profiles, such as the Authentication Profile have not been expanded.

The flow of a protocol specified by an Operation cannot be pre-determined ahead of time, because its next step will often be contingent on events that occur during the protocol execution itself. For example, a SSO Operation will be conducted differently if a user enters a correct password to when a user enters an incorrect password. In the former case, the Operation may attempt to return a 'sign-on successful' type of result to the requester, while in the latter the Operation may attempt to divert the user to a 'try again' password screen. CPs have to make decisions as to whether they can assert a user's identity or not. Again the Operation will progress differently depending on whether the

answer is a yes or a no. Because we must allow for dynamically determined contingencies in the protocol flow, Operations cannot be constructed from a simple series of Profiles (or Profiles from Connections) prior to protocol execution. They must be constructed instead from a contingent interconnection of Profiles, which are in turn constructed from a contingent interconnection of Connections.

This approach, although somewhat involved, allows us to represent an Operation as something that is reasonably easy to understand and design on the high level (since it lends itself to modelling), while encapsulating the resulting complexity at a lower level. It also gives us the maximum flexibility possible, since it does not tie us down to any particular protocol flow.

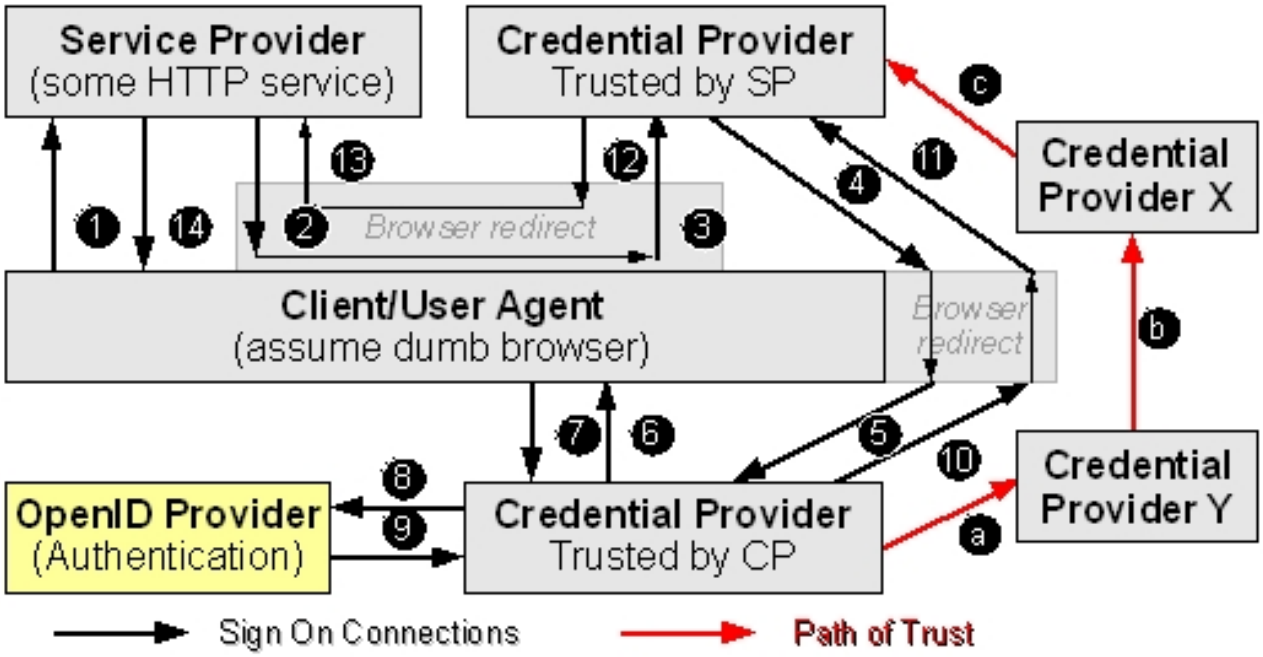

Figure 5.: Single Sign-On Operation (with simplified view OpenID authentication)

The most important Operation required by our identity model implementation is the Single Sign-On scenario. See Figure 4 above for a diagrammatic view of this Operation. In this implementation of the identity model, OpenID backs the UA's CP authentication, and trust is established dynamically between the UA's CP and the SP's CP by passing a token (potentially a SAML artifact) across a 'trusted path' of Cps.

2.7.5 Actors and State

Operations define the communication scheme or protocol flow. In order to make practical use of

Operations we must express them from the point of view of the participating Actors.

Each Actor supporting a given Operation constructs a state machine representing the States (in the protocol flow) the Actor may occupy. Actors, with the exception of Actor initiating the Operation, begin in a 'waiting state', and progress from one state to another depending on dynamic factors determined during protocol execution.

The Operation, and hence the protocol flow, will come to an end when Actors reach pre-determined States.

The Single Sign-On Operation begins when the SP requests its CP to assert the UA's identity and ends when the SP receives a response from the CP. A broader view of the same Operation begins when a resource is requested that triggers an identity request, and ends when access to the resource is granted or refused.

2.7.6 Open Source Project – *IdentityFlow*

Software supporting the identity model implementation, as well as an SSO Operation and a demonstration, are being developed as part of an open source project. Open source development is consistent with the spirit of the OPAALS project and helps facilitate the development of identity model implementations by making the source available to a wider community.

The software is written in Java with Servlet/JSP extensions as an example web interface to accept Actor Connections (SOAP/web services could be used, or an alternative mechanism).

A sourceforge.net project called *Identityflow*[3,4] has been established to host the project. The project contains a home page outlining a description of the project and useful links to project resources, a CVS repository containing code that is under development, a Maven 2 repository containing custom dependencies required to build the code, and immediately deployable software releases containing the identity model software implementation and demonstrations. The software currently consists of five sub-projects: Identity Model SAML, libraries for integrating the OpenSAML libraries with the identity model; Operation Builder, libraries for building Operations; Operation Request Handler, libraries for intercepting and handling incoming connections, including Servlet/JSP integration; Single Sign-On Operation, an implementation of an SSO Operation based on SAML profiles and HTTP GET/POST bindings; and Actors using SSO Example, which is a demo illustrating the use of

[3] http://identityflow.sourceforge.net/

[4] http://sourceforge.net/projects/identityflow/

the provided SSO Operation to identify a user agent to a service provider.

The build system used to build all of the software provided is maven[5]. With maven installed it is trivial to check out the latest code from the sourceforge.net CVS, build each sub-project and produce a Java web archive (war) file containing the demo application. The war file can be deployed immediately on a servlet container implementing the Servlet 2.5/JSP 2.1 specifications, such as Tomcat 6.0.

The demo, or sample application, provided, simulates a typical user-agent – service provider interaction, where the identity of the user-agent must be established by the service provider. The service provider initiates an SSO Operation in order to accomplish this. Each Actor is represented by a simple JSP page that redirects incoming Connections to an Interceptor. The Interceptor determines which Operation the Connection pertains to and forwards it to its OperationHandler. The OperationHandler then determines, based on the current state of the Actor and the incoming Connection, what it should do to continue protocol execution, and afterwards what outgoing Connections to other Actors it should initiate.

In the demo, the service provider's IdP cannot establish the identity of the user-agent so it refers to the user-agent's IdP. SAML assertions are then passed back to the service provider as directed by the SSO Operation, which establish the user-agent's identity to the service provider. OpenID is used as the backend identity provider for the user-agent's IdP. This demo, and the SSO Operation, will become more sophisticated over time as they exhibit a greater range of functionality.

[5] http://maven.apache.org/

3 Distributed Trust Model for Digital Ecosystems

In Deliverable 4.3[5], we proposed an evolutionary trust model for digital ecosystems based on current security technologies and reputation mechanisms. Though we primarily took a computer science approach, in order to solve concrete issues related to the open distributed platform of the DE, we also relied on concepts and methods from ecology, economy and sociology. The model is suitable for the decentralised and dynamic nature of DEs. We base our model on standards and consider scalability, availability and ease of putting into practice as the most important characteristics of the model. The model has four main components: distributed identity management, peer-to-peer reputation, trusted rating agencies, and learning. These components address different security, social and evolutionary aspects of the DE and create a complete model. The identity management model enables creating a reputation framework by providing ways for securely identifying entities. The reputation framework has two main components: a peer-to-peer reputation model and a decentralised trusted rating agencies model. DEs evolve in time in order to respond to changing conditions.

Experiments and simulations are also required to evaluate convergence of trust towards stability, and to test performance and scalability of various approaches taken. It is also necessary to evaluate robustness of the system in the face of attempts to corrupt it, including collusion between malicious entities (issuing false ratings about each other, for example).

Because of the constantly evolving and dynamic nature of relations in a DE, our goal is to model an adaptive multidimensional reputation-based trust. Trust will be primarily established based on recommendations expressed by users not only about other users, but also about data/knowledge, services and infrastructure. Hence, we tackle *multiple levels* of trust. Moreover, in a DE, users organise themselves in *social networks* and trust values are, therefore, local and relative to each user and their neighbourhood or social network. Because users interact in *different contexts* (e.g. automotive, construction, IT), the model allows for the expression of trust recommendations relative to a user-defined keyword (folksonomy). Contexts could be complex and expressed using a taxonomy. By using different contexts to express the trust level in a certain entity, we model multidimensional values for trust. For example, a service could be rated differently based on availability, response time, memory usage, result accuracy, etc.

3.1 Reputation system overview

For the purposes of this work, we adopt a two layer model for communications between peers. Peers can either interact for a transaction or to exchange trust information. For modelling purposes, each service usage interaction is a discrete event. A logically separate trust management layer handles threat notifications and other pertinent information.

We mimic social trust by setting a fuzzy trust level. Each different service can then make an appropriate decision based on this trust level – e.g. certain actions may be allowed and others not. In our system, we model trust as a vector. In the simplest case, at least if there is just one service, this can be viewed as a simple number in the range (0,1). Each peer may then maintain a local trust score relating to each other peer of which it is aware. If peer A's trust in peer B is 1, then peer A completely trusts peer B. A score of 0 indicates no trust. If trust is to be a probability measure, then the (0, 1) range is natural.

3.1.1 Trust representations

We will represent trust as a probabilistic measure in the range (0, 1). Zero means no trust or no information about trust and 1 means complete trust.

If we want to represent trust as a complex value such that it is more meaningful and conveys more information about the trustee, each dimension of trust will be represented as a probabilistic value from 0 to 1. For example, if agent A wants to express the trust it has about service S, it can use the following structure:

(A, S, availability, 0.9)
(A, S, response_time, 0.3)
(A, S, result_accuracy, 0.8)
(A, S, memory_usage, 0.2)
(A, S, security, 0.6)

Users or companies could be trusted in a certain context, but not in another one. For example, A can trust B about car repairs, but not about baby sitting. In order to express trust in a particular context, an agent can use the following structure: (*TrusterID, TrusteeID, Context, trust_value*). Trust_value can either be simple or multidimensional as showed above. For defining contexts, we will allow users to build a folksonomy by defining own keywords for expressing contexts.

3.1.2 Architecture

We propose the overlay of a distributed trust management infrastructure on top of the service delivery infrastructure (which may itself contain multiple layers).

Figure 6 illustrates the relationship between underlying services and this new infrastructure. With our proposed trust overlay architecture, a trust management layer operates separately from the mechanics of the transactions themselves. Two message passing interfaces are defined between the transaction layer and the trust management layer and another between the trust managers of individual peers. The interfaces are as follows (Figure 6):

1) Experience reports: Transaction engine → Trust manager
2) Trust recommendations: Trust manager ↔ Trust manager
3) Policy updates: Trust manager → Transaction engine

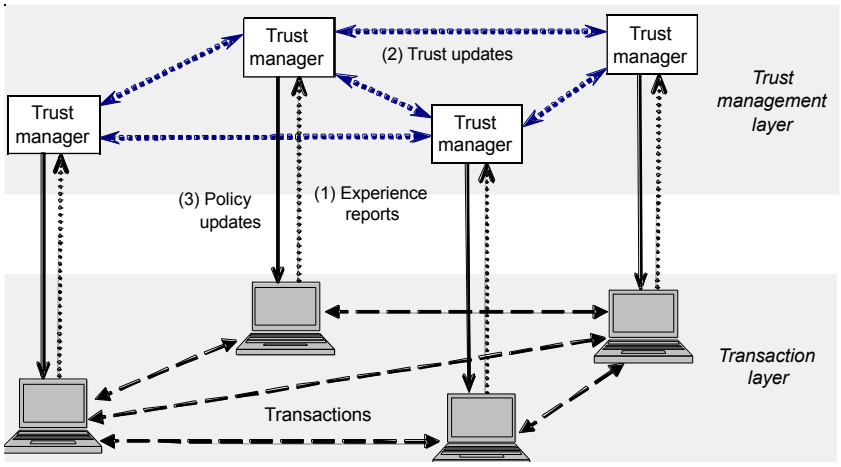

Figure 6: Trust overlay helps to secure transactions in a digital ecosystem.

Transactions are as normal, with the trust information just influencing protection mechanisms. The trust manager gathers transaction experience and uses this together with the experience of collaborators to inform its trust in other peers.

In the initial case, all trust values are set to a low value. Having initial values set to zero would prevent the so called *Sybil Attack*[13] , whereby attackers can take advantage of a default trust level by maintaining multiple identities. This is impractical though as we need to have some low-risk services enabled in order to get experience of other peers.

We then need to have a system to build up trust as peers gain experience of each other or learn of each other's reputation. In our system, trust can be updated in two ways:

1) *Direct experience:* On completion of a transaction between two peers, each peer updates its trust in the other based on a measure of satisfaction with the transaction.

2) *Reputation:* Peer A notifies other peers in its *neighbourhood* of the trust score that it has for peer B. This will change significantly following a security-related event.

How this neighbourhood is defined is significant. The neighbourhood of a peer is the set of peers with which it can communicate or with which it is willing to interact. The choice of neighbourhood peers is up to each individual peer to decide, and can be viewed as the peer's social network. In reality, choice of neighbours may depend on physical geography, network topology, frequency of contact or even trust level.

A benefit of this system is in using these trust scores to tune security settings. Trusted peers can be dynamically provided with more privileges than untrusted peers. In our system, as mentioned, we model all interaction between peers in terms of services. Each peer then sets a threshold trust level for access to each service in which it participates. If the trust score of a peer decreases, for example due to detected suspicious activity by that peer, services available to that peer are reduced.

How this neighbourhood is defined is significant. The neighbourhood of a peer is the set of peers with which it can communicate or with which it is willing to interact. The choice of neighbourhood peers is up to each individual peer to decide, and can be viewed as the peer's social network. In reality, choice of neighbours may depend on physical geography, network topology, frequency of contract or even trust level.

The main benefit of this system is in using these trust scores to tune security settings. Trusted peers can be dynamically provided with more privileges than untrusted peers. In our system, as mentioned, we model all interaction between peers in terms of services. Each peer then sets a threshold trust level for access to each service in which it participates. If the trust score of a peer decreases, for example due to detected suspicious activity by that peer, services available to that peer are reduced.

3.1.3 Rating Agencies

The trust values computed by the P2P reputation system are *subjective* and in some situations may not be acceptable. For example, certain business transactions could have higher constrains. Because of that, we see the need of using Rating Agencies which issues certificates to registered users. Certificates issued by the agencies should be *objective* and hence more trustworthy. Unlike the P2P

reputation system which uses recommendations expressed by users, the rating agencies make use of authorisation certificates issued by different domains. A Rating Agency is a dedicated service that could be offered by each of the peers. Agencies specify predefined criteria on which users are registered (i.e. necessary credentials). Each entity decides on its own to register or not with an agency. Each SP or user decides on its own which agencies to trust. Similar to CPs in the Identity Management model, rating agencies are distributed across the system, establish trust relationships between each other and translate certificates issued by trusted agencies. Moreover, agencies across the system cooperate with each other to retrieve information about unknown users.

3.1.4 Learning

The trust model we propose for DE is based on a multidisciplinary framework first introduced in [14] and adapted in Figure 7. The right side column represents possible technology platforms suitable for DE service execution management. The left side column represents the trusted environment that SMEs use to perform their business goals.

Trust can be provided in a DE in two ways:

- Through experiences inside the system

- Through institutional trust: an institution CA initially decides to trust a user by examining existing institutional trust relations with the institutions who know the user.

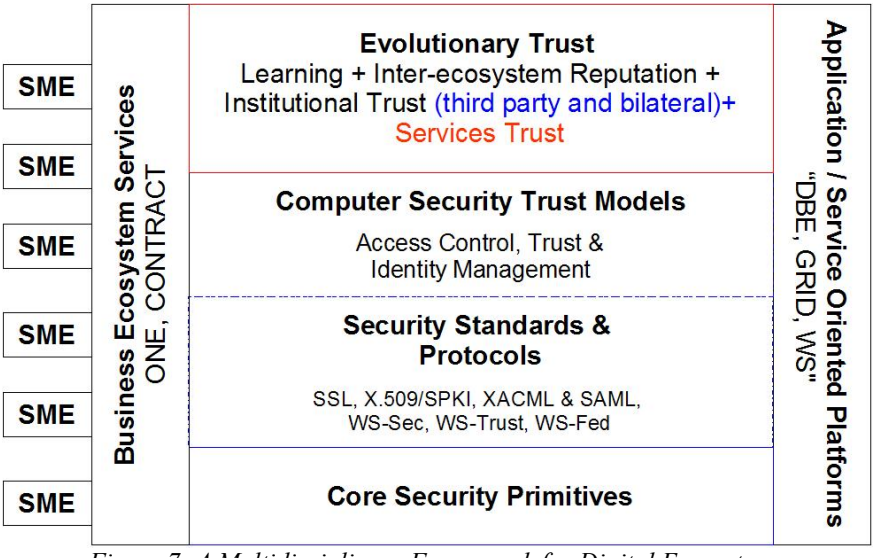

Figure 7: A Multidisciplinary Framework for Digital Ecosystems

This will create an independent and evolutionary platform capable to adapt and evolve on the basis of the evolution in social institutional trust.

3.2 Experiments

Initial simulation work has been done to assess the dyamics of our peer-to-peer reputation system.

3.2.1 Single-service scenario

The objective of this experiment is to see if we can get an improvement in the effectiveness of service access filtering by applying trust scores. The figures below show how this has been achieved in an illustrative case. In this experiment, a network of 50 peers is simulated; also there is a single "bad guy" who is responsible for 50% of all activity in the system. A default initial trust score of 0.2 is used for illustration purposes (to distinguish the bad from the simply unknown). Trust convergence for normal peers is a moderately fast exponential average. The neighbourhood consists on average of one-seventh of all peers, so we have a sparse, but still connected, topology.

We choose relatively flat (but distinct) probability density functions for bad activity indicators for both good and bad activity. Both have the Gaussian (normal) distributions shown in Table 1. Note the overlap implied by the relatively large standard deviation value for bad service usage.

Table 1. Parameters for Gaussian (normal) distributions used in experiments.

	Mean	Std Deviation
Benign service usage	8.0	4.0
Malicious service usage	1.0	2.0

Spam filters, for example, combine a variety of measures into a suspicion score and compare this score with a pre-defined threshold. For our experiments, a fixed threshold of 5.0 is chosen (*SpamAssassin* default) and used as a benchmark. As can be seen in Figures 5 and 6 below, a significant reduction in both false positives and false negatives can be achieved with auto-tuning of the threshold (based on trust values). Auto-tuning is of course most effective in a steady-state situation when trust values are quite stable. A range of other predefined threshold values were also tried, but with no better results than the value of 5.0 shown. Choosing a higher predefined threshold causes an increase in false negatives and choosing a lower predefined threshold causes an increase in false positives.

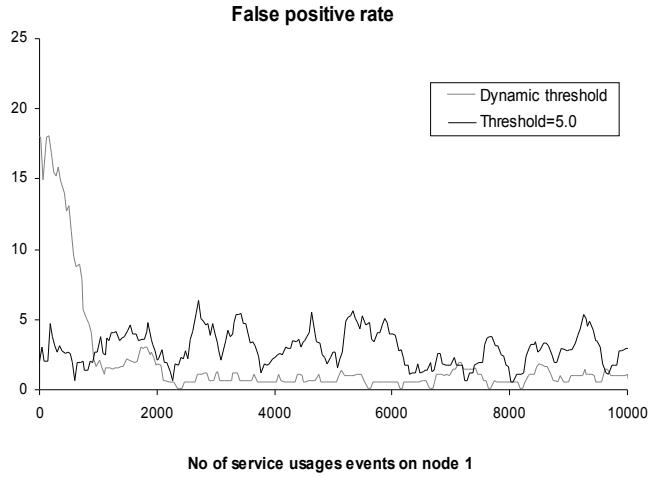

Figure 8. Comparison of dynamic vs. fixed threshold: impact on rate of false positives.

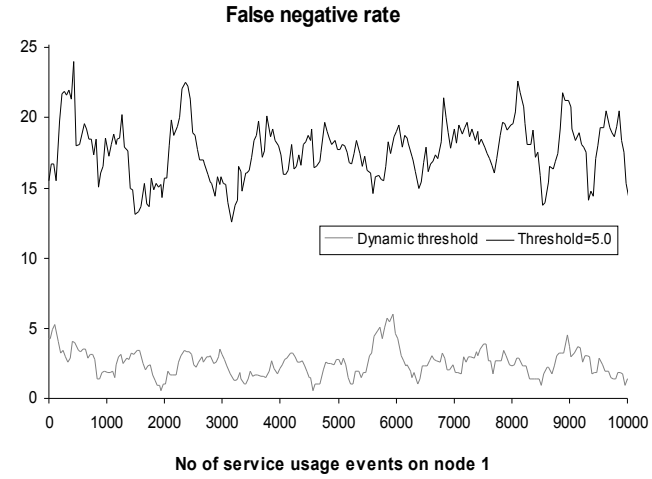

Figure 9. Comparison of dynamic vs. fixed threshold: impact on rate of false negatives.

4 Distributed Accountability Model

Deliverable 4.2[6] provided the initial model for accountability in digital ecosystems. Much of this section is a summary of that report and a short section on what is planned for the work in future iterations of the accountability model.

This initial model for accountability is largely based on adapting two of the works researched in D4.2 (PeerMint [2] and SOA Accountability Architecture[15]). The limitations of this model are discussed with a view to integrating a suitable solution in the next iteration. It should be noted that this model is an initial general approach to providing accountability in autopoietic peer-to-peer networks and its evolution is dependent on decisions of network design coming from WP3.

For the purpose of a digital ecosystem, the primary concern is that of accountability within distributed digital communities for the purposes of knowledge sharing, service provision and consumption, and collaboration. A useful definition of accountability in this context is given by Schedler [16] as *"A is accountable to B when A is obliged to inform B about A's (past or future) actions and decisions, to justify them, and to suffer punishment in the case of eventual misconduct"*. Within our context *A* and *B* could represent individual players in the ecosystem, or *A* could represent a participant and *B* represent the ecosystem or community, or indeed both could represent ecosystems and their accountability to each other.

4.1 Accountability and Peer-to-Peer Systems

Peer-to-peer networks operate through the leveraging of resources made available by its participants rather than relying on a set of servers providing a set of centralised finite resources. A pure peer-to-peer network operates on the basis that all peers are equal and each operates as both a client and a server in terms of providing and consuming services and/or content. These types of (pure peer-to-peer) networks have become the subject of much research and commercial interest in recent times due to their inherent lack of a single point of failure as resources are replicated across nodes, low cost of deployment in large-scale roll-out (due to the non-requirement of centralised management systems) and the fact that as more resources join the network, the greater the capacity of the network (due to the fact that it is the participants who provide resources). Another element of peer-to-peer technology that is of benefit on a social level is that, by its nature, it empowers its users who

in effect collectively own the network and its resources. It is partly this reason that peer-to-peer networks have become the technologies of choice in the area of digital ecosystems.

In terms of accountability, peer-to-peer networks bring the same challenges that are in place for accountability in traditional client server models, i.e. Non-repudiation, fairness, etc. In fact, in peer-to-peer the requirements could be considered more important due to the fact that participants now operate in entirely untrusted environments without any centralised authentication mechanisms providing trust and strong user identity. In addition, the lack of centralised control brings its own challenges in terms of replication of (sometimes sensitive) data and resources across peers in untrusted environments.

Two previous works on peer-to-peer accountability of particular interest are that of *Karma* and *PeerMint*. A brief discussion of these is included here.

4.1.1 Karma

Karma [17] is a framework for sustaining resource sharing peer-to-peer networks using a secure economic model. The motivation for the work is the avoidance of freeloading is resource sharing networks. The system s economic in that it keeps track of the resource purchasing power of peers in the network.

The framework provides properties of non-repudiation, certification and atomicity, protecting against both malicious providers and consumers. In addition, periodic correction to the outstanding karma in the system is performed in order to reduce the effects of inflation or deflation. Inflation can occur when peers use up their balance to consume resources and then leave the system. Deflation can occur when peers accrue large balances and then leave the system. Karma makes use of a value/resource transfer protocol which is shown in Error: Reference source not found in Figure 8 and described in more detail in D4.2 [6].

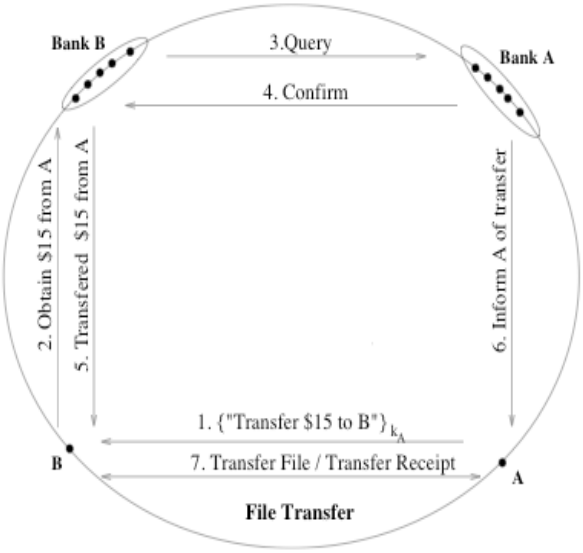

Figure 8: KARMA resource/value exchange protocol

.

4.1.2 *PeerMint*

In 2005 Hausheer and Stiller published a decentralised scheme for accountability in peer-to-peer networks which they called *PeerMint* [2]. The work was performed as part of the MMAPPS[6] project and was concerned with accountability mechanisms for commercial peer-to-peer applications. The provided scheme can be used as a means of ensuring fair sharing of resources or as a means of applying charging and payment mechanisms to peer-to-peer applications. The work introduces the concept of session peers in addition to account peers in order to reduce significantly the possibility of collusion between peers. The session peers are responsible for maintaining balances and updates for the current session and for validating actions against a predetermined Service Level Agreement (SLA) between the peers.

[6] *MMAPPS (Market Management of Peer-to-Peer Services),* http://www.mmapps.org *(domain has lapsed)*

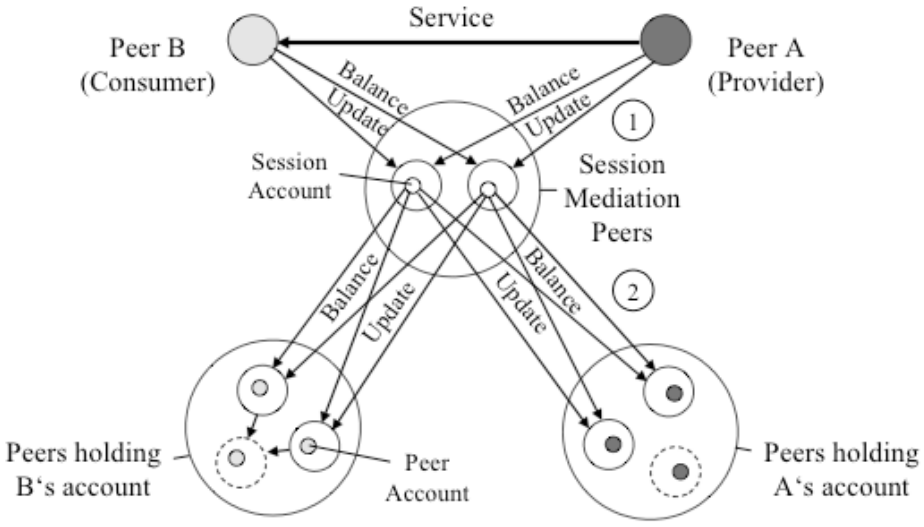

Figure 9: PeerMint Distributed Redundant Accounting

PeerMint takes a redundant approach to both session peers and account peers through the replication of data across a set of peers. The selection of account peers is made in a similar way to to the hashing method of *Karma* in order to optimise routing. The selection of session peers also uses this approach but the hash is performed on a combination of the peer IDs and a timestamp denoting the point of session initiation. In both cases (account and session peers), when a peer goes offline a new peer is selected to take its place. The new peer obtains the balance from the remaining peers associated with the account or session. The model is shown below in Figure 9. New peers are shown as a dashed circle.

4.2 Accountability Model for Digital Ecosystems

Before determining a model for accountability it was necessary to define a set of requirements for providing accountability in peer-to-peer digital ecosystems and to determine if previously developed frameworks could provide this functionality.

Below is a list of the technical requirements for accountability in peer-to-peer deployed digital ecosystems.

Decentralised

One of the fundamental requirements for digital ecosystems is that there is no single point of failure. If any one node (or any sets of nodes) becomes unavailable, the system must be capable of recovering completely without any external intervention. A suitable accountability solution requires

therefore that there is no dependence on a single centralised accountability manager or co-ordinator and that all functionality is distributed across the system.

Service Composition

A crucial aspect of digital ecosystems is collaboration between participants. In this sense it is important that services and resources can be combined to create new services. This service composition needs to be dynamic and cannot be known in advance. An accountability solution requires that such a dynamic composition of services can be seamlessly and efficiently accounted for.

Scalability

The number of peers in a digital ecosystem is arbitrary, ranging from a handful to several thousand. A suitable solution needs to work for large sets of peers as well as large service compositions.

Security

The implications for security when providing accountability in dynamic composing in digital ecosystems are wide ranging. Integrity of accounted data, availability of accounted data, confidentiality of accounted data, privacy of transactions al need to be considered when designing a suitable accountability model.

Contracts

Exchange of contracts or service level agreements prior to service consumption is a vital aspect of a commercial application of digital ecosystem deployment. It is desirable that aspects of these agreements can be assessed at runtime to ensure that parties are operating within the bounds of the agreement.

The model under development is a combination of the *PeerMint* model published by Hausheer and and Stiller [2] combined with the *SOA Accountability Architecture* desigend by Zhang et al [3]. The idea here is to take the concept of the *Accounting Authourity* introduced by Zhang and to deploy this as a distributed service and combine this with the distributed *PeerMint* model. The model is shown in Figure 10. Although the diagram shows a simple transaction between 2 services, the introduction of the *Accounting Authority* provides the required functionality necessary for composed services. The role of the accounting authority in a composed service scenario is provided by the super-set of mediation peers.

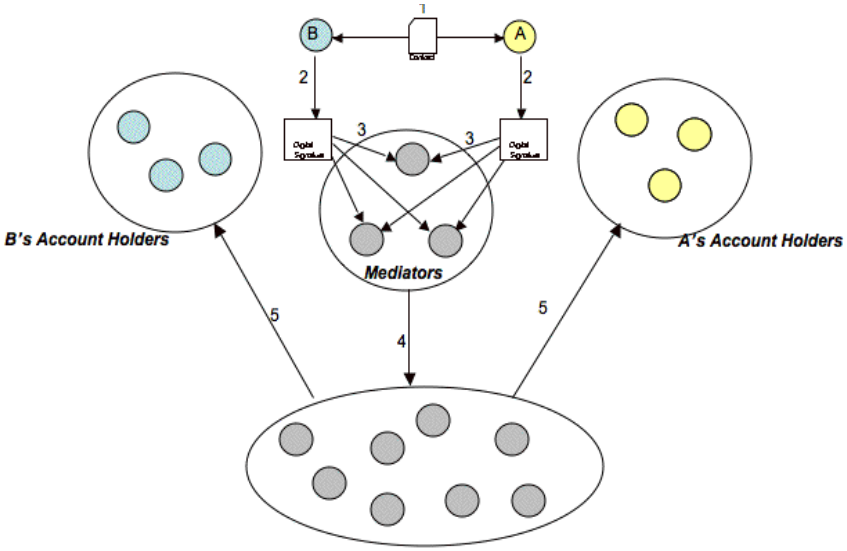

Figure 10: OPAALS Distributed Accountability Model

The model shows two peers (A and B) interacting in a peer-to-peer services trading environment.

Prior to interaction between the peers, each peer is assigned a set of trusted peers as account holders. This assignment is based on historical availability of peers and this set of peers is updated over time as the system evolves. Upon joining the network, each peer is assigned a unique 128-bit peer ID calculated fron that peer's public key using a secure hashing method. This is the approach taken by both The public key is provided by the peer's identity (modelled through Task 4.1, Distributed Identity Model in OPAALS). The original selection of account holders is based on a combination of trustworthiness and leaf sets closest to the peer to be accounted for. The trustworthiness of these peers is determined based on the distributed trust model being developed within workpackage 4 in OPAALS. In a similar way (as in *PeerMint*) the session peers are selected using a hash on the combination of the interacting peers IDs in combination with a timestamp. Both the account holders and the session peers are normal peers in the network. An interaction between A and B is shown in the diagram and is explained as follows:

Initially the peers exchange a contract which includes the charging schemes they agree for service consumption. At runtime when the peers agree to interact, they are assigned a set of session mediation peers. These mediation peers are provided with a copy of the contract. These peers are responsible for holding accounting data from both A and B for the current interaction session.

Each message passed between A and B each session is metered and the resulting usage data is digitally signed by the peer itself. In combination with the distributed identity model being

developed and the availability of peers' public keys, these signatures can be verified.

The digitally signed usage data is then passed to the mediation agents who compare A's usage with B's to check that both are agreeing on how they are interacting and that these interactions conform with the previously agreed contract.

As the session evolves, the accounted data is passed to the super-set of mediation peers operating as the Accounting Authority and checked for integrity with respect to the overall service composition.

At the end of the session, the usage data is consolidated into a charge record by the Accounting Authority. These charge records are delivered to the corresponding set of account holders for both A and B.

5 Distributed Collaborative Knowledge Sharing

This work has been previously reported in more detail in deliverable D4.4 [7]. The contribution of the work performed can be broken down into three different areas, as follows:

1. Framework for collaborative knowledge sharing
2. Novel tools for collaborative knowledge sharing
3. New algorithms of distributed knowledge sharing for digital ecosystems

This section will start by providing a brief introduction to the subject matter and then discuss each contribution of the work.

5.1 Introduction

Knowledge sharing is becoming the basic and important behaviour and culture of the individual, the organisation, the community and wider society. Knowledge sharing is also the key power of the development of individual, organisation, community and society. Knowledge sharing can help members of organisations, communities or individual to share common knowledge to achieve some common tasks. In a digital environment, a collaborative system is formed by the digital components such as digital data, information and knowledge. Collaborative knowledge sharing between participants of the ecosystem is one core of the digital ecosystem. The digital ecosystem should be an effective platform to enable the participants of the ecosystem to learn and share divers knowledge such as domain knowledge, skills, expertise, useful information, experiences and perspectives from others.

Data, Information and Knowledge

It is useful to distinguish between data, information and knowledge for the purpose of the work. Data is collection of digital symbols without context. Data comes in many types or formats such as numbers, words or pictures. Information is data that has context and meaning. Therefore we can understand information rather than data. The data becomes information once it is put into a framework or structure that provides context. Common ways in which data is taken and converted into information are in using spreadsheet or database programs to solve problems. Information can be converted to knowledge when information is useful to increasing the understanding of a subject,

and make decisions, form judgements or opinions or forecasting future outcomes. Figure 11 visualises the relationship between these concepts.

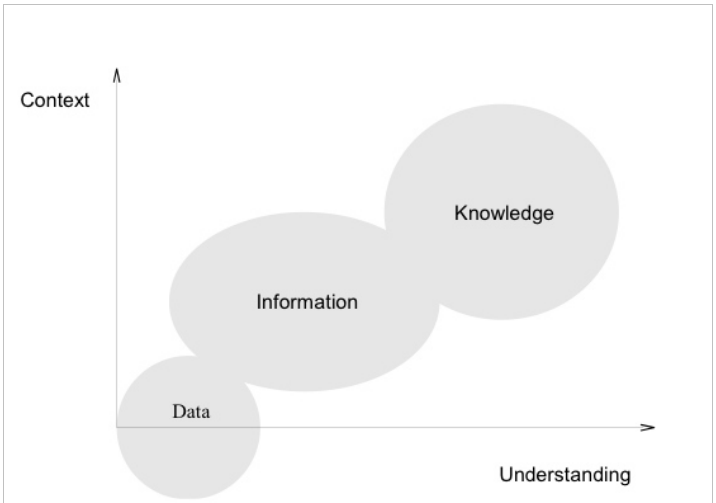

Figure 11: Data, Information and Knowledge

Knowledge Discovery

In the digital ecosystem, some knowledge is pre-existing, but some knowledge is previously unknown, implicit, i.e. hidden in large data. We therefore should extract the knowledge from large data with techniques such as data mining or KDD (Knowledge Discovery in Databases). The techniques of data mining are widely used in research and applications to discover relationships and knowledge that are implicit in large volumes of data and are interesting in the sense of impacting an organisation's practice. For example, data mining techniques can help organisations in providing better, more customised services and support decision making.

Knowledge Sharing Platforms

There are different models to share knowledge in different environments. We consider three sharing environments or platforms: Organisation, Community and Digital Ecosystem networks. An organisation, such as a company or an university, is a formal, social arrangement with collective goals and controls its own performance in a bounded and fixed environment. Such an organisation has its own internal knowledge system. The members of an organisation share knowledge via

intranet, local networks, internet or in person. Such organisation networks are a formal and structured platform that focuses on domain knowledge sharing that are critical to the organisation.

A community is a group of people who have common interest or work. The members of a community can be professionals within an organisation, or across several organisations, or they can simply form a non-work-related community. They may not all know each other, yet feel a sense of community because they have similar interests and face similar challenges. They can share tips, hints, ideas, and knowledge on-line, or in person. They realise the value of sharing knowledge with their peers, and learning from each other. Communities are often informal and loose in structure and governance.

5.2 Contributions

Our research works focus on distributed knowledge sharing for digital ecosystem. We have the following contributions:

- *Framework for collaborative knowledge sharing between participants of the ecosystem*

  We propose the framework for collaborative knowledge sharing (see Figure 12 below) that includes Infrastructure, Sharing platforms, Sharing requirement, Knowledge resource, Knowledge engine and Knowledge Sharing. Knowledge engine is for acquirement of knowledge that responds to where and how to find knowledge. Knowledge Sharing is for sharing knowledge in digital ecosystem. It focuses on the answer to how to share knowledge. A more detailed description is available in D4.4 [7].

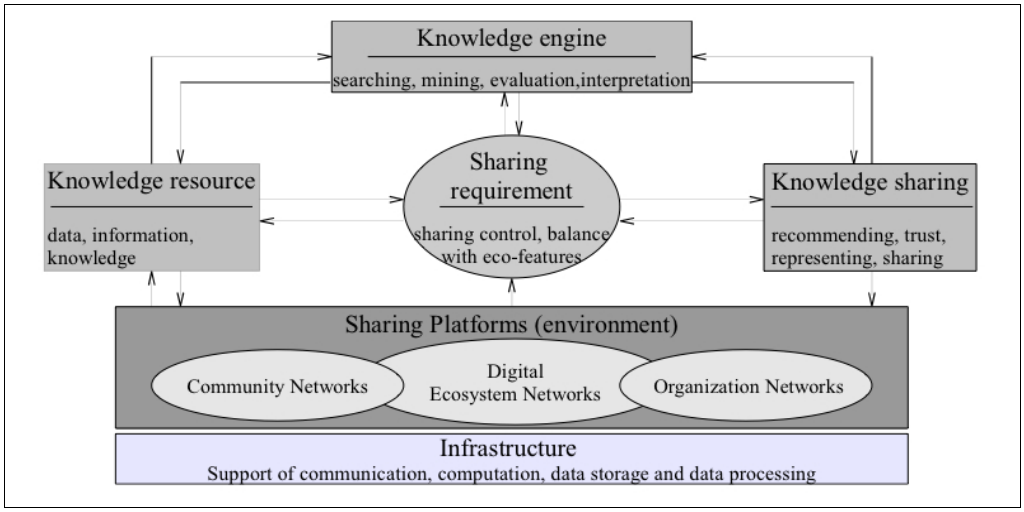

Figure 12: Framework for Knowledge Sharing

- *New strategy of sharing distributed knowledge for digital ecosystems.*

  We propose the integration of community networks, organisation networks and digital ecosystem networks three overlapped platforms for sharing distributed knowledge for digital ecosystems. The organisation networks provide a platform for local knowledge sharing, mapping and storage by one organisation, e.g. one enterprise. The community networks overlap with organisation networks to share distributed knowledge by common topics for a community, e.g. OKS research group. The digital ecosystem networks provide more wide, open and dynamic knowledge sharing space by the integration of community networks , organisation networks and other knowledge that are not included in community networks and organisation networks.

- *Data Mining and Knowledge Discovery in Databases*

  Knowledge Discovery in Databases (KDD) is a process of discovering non-trivial, previously unknown and potentially useful information from a huge collection of databases. Data mining is a step of KDD process, but also referred to KDD. The KDD process is an interactive process. Because it involves many steps with decisions made by users, different KDD process may have different steps. However, there are some basic steps in common (See Figure 13 below).

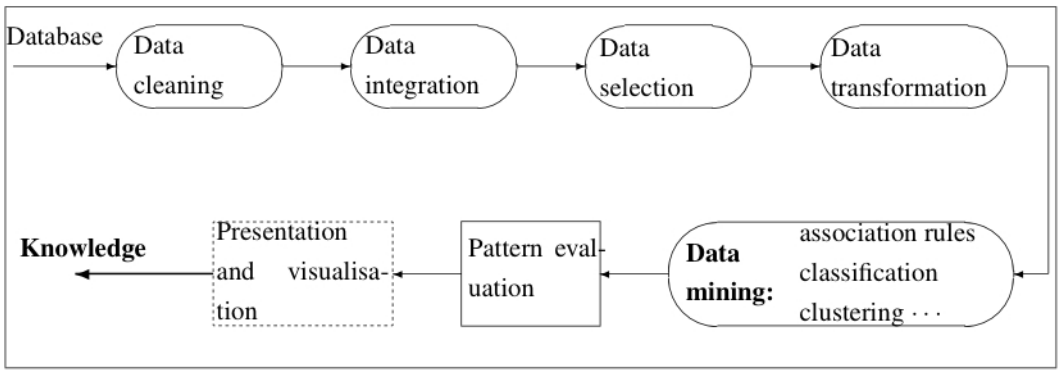

Figure 13: Data Mining and KDD process

- *Formal Concept Analysis (FCA) for digital ecosystems*.

  The formal concepts can be used to explore, present and interpret the knowledge from the large data in digital ecosystem so that facilitates knowledge sharing in digital ecosystem. FCA is an ontology analysis model. Using FCA, we can analyse, define, refine, merge, and share ontologies.

- *New algorithms of distributed knowledge sharing for digital ecosystem*

  We have developed new algorithms for extracting knowledge in digital ecosystems. The algorithm is based on formal concept lattice. Theoretical foundation of concept lattice founds on the mathematical lattice theory. Lattice is a popular mathematical structure for modeling conceptual hierarchies. Concept lattice is a method for deriving conceptual structures out of data. From concept lattice, we can study the relations between objects and attributes in a formal context, and how objects can be hierarchically grouped together according to their common attributes. Certain object subset and the set of their common attributes can represent each other, such duality sets form a formal concept, which the attribute subset is called intent and the object subset is called extent. Among the formal concepts, there exists an order relation, they form a complete lattice: concept lattice. Each node in the lattice is a concept and the corresponding graph (Hasse diagram) that is considered as the generalisation and specialisation relationships between concepts. Such graphical structure represents directly and visually the relations of conceptual hierarchies. It allows us to analyse and mine the complex data for such as classification, association rules mining, clustering, etc. Furthermore, concept lattice also provides an effective tool of

knowledge visualisation.

6 Future Directions

6.1 Distributed Identity Model

The work described in this document and in deliverable D4.1 [4] will be augmented with further work in Workpackage 3. This will take the form of further refinement of the Identity Operations and a close integration with the Trust Model described in [5]. Issues such as a possible requirement for global uniqueness versus the lack of single point of control need also to be addressed. Also implementation plans for deploying the Identity Model in the planned infrastructure. Currently the proposed infrastructure will be based on JXTA[7].

6.2 Distributed Trust Model

The Trust model described in [5] and in this document rely on experience reports generated internally within a digital ecosystem in the creation of trust ratings and also on those trust ratings being shared among Trust Managers to create a community driven reputation based trust system. In the next period of the work (in Workpackage 3) we will examine the idea of institutional trust, i.e. ratings made about entities by external trusted institutions. An example of this type of institutional trust would be a driving licence verified by a state Department of Transport. In addition to this we will address integration points for our trust overlay model in other aspects of the peer-to-peer network being developed in Workpackage 3. In particular, we will look at how data generated by the rating agencies and distributed transaction facility can be used in the trust update process. We will also examine algorithms for trust in more detail.

6.3 Distributed Accountability Model

The Accountability model described in [6] and in this document provides most of the functionality required to allow for distributed accountability in peer-to-peer networks where dynamic service composition is the enabler for multi-peer collaboration. However, some requirements are not satisfied fully by this model, namely security issues. Privacy is a security aspect which also needs to

[7] JXTA Community Project, https://jxta.dev.java.net/

be addressed in a future evolution of this model. In the model provided here, any peer which holds *A*'s public key can, in theory, access details of *A*'s account from its account holders. In the next evolution of this model an access control mechanism needs to be added to overcome this issue. Furthermore, decisions on hashing methods, routing tables, leaf-sets and selection of account and session holders need to further considered as the network design decisions emerge from Workpackage 3 and will be reported in Deliverable D3.8.

6.4 Distributed Collaborative Knowledge Sharing

One main aim of the OPAALS is to develop an integrated theoretical foundation for Digital Ecosystems research spanning three widely different disciplinary domains: social science, computer science, and natural science. Our proposed tools, data mining and FCA, have wide applications for social science, computer science, and natural science. We will develop a common models and platform of knowledge acquirement, knowledge sharing and knowledge management for social science, computer science, and natural science.

The techniques of data mining and FCA have many potential applications for collaborative knowledge sharing of the ecosystem. We will develop suitable models and algorithms to analyze, extract, acquire, deliver and share knowledge of the ecosystem. This development can be integrated with other WPs in Phase 2 of the project such as WP5 (Integration with the Digital Ecosystem platform) and WP10 (Sustainable Research Community Building in the Open Knowledge).

References

[1] OPAALS Consortium, *Distributed Identity Model for the Digital Ecosystem*, 2008, http://files.opaals.org/OPAALS/Year_2_Deliverables/WP04/D4.1.pdf

[2] OPAALS Consortium, *Distributed Accountability model for an Autopoietic P2P network*, 2007, http://files.opaals.org/OPAALS/Year_1_Deliverables/WP04/D4.2.pdf

[3] OPAALS Consortium, *Trust Model for the DE*, 2008, http://files.opaals.org/OPAALS/Year_2_Deliverables/WP04/D4.3.pdf

[4] OPAALS Consortium, *Distributed Collaborative Knowledge Sharing Framework*, 2008, http://files.opaals.org/OPAALS/Year_2_Deliverables/WP04/D4.4.pdf

[5] Koshutanski, H., Ion, M. and Telesca, L., 2007, *A distributed identity management model for digital ecosystems*, in Proceedings of International Conference on Emerging Security Information, Systems and Technologies (SECURWARE'07), IEEE Press

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

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

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

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

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

[11] Douceur, J., 2002, *The Sybil Attack*, in Proceedings of International Workshop on Peer-to-Peer Systems,

[12] Telesca L., Koshutanski H., *A Trusted Negotiation Environment for Digital Ecosystems* , Building the foundations of Digital Ecosystems: FP6 Results and Perspectives, pp, 2007, Eurpean Commission.

[13] Hausheer, D., Stiller B., *PeerMint: Decentralized and Secure Accounting for Peer-to-Peer Applications* , NETWORKING 2005, pp40-52, 2005, Springer Berlin / Heidelberg.

[14] Zhang, Y., Lin, K.,, 2007, *Hierarchical Management of Service Accountability in Service Oriented Architectures*, in Proceedings of IEEE International Conference on Service-Oriented Computing and Applications, IEEE Press

[15] Schedler, A., *Conceptualizing Accountability* , The Self-Restraining State: Power and Accountability in New Democracies, pp13-28, 1999, Lynne Reiner Pulishers.

[16] Vishnumurthy, V., Chandrakumar, S., Sirer, E. G., 2003, *KARMA: A Secure Economic*

Framework for Peer-to-Peer Resources, in Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA.,

[17] Zhang, Y., Lin, K., Yu, T., 2006, *Accountability in Service Oriented Architecture: Computing with Reasoning and Reputation*, in Proceedings of IEEE Conference on e-Business Engineering, IEEE Press