

Distributed Support for Public and Private Accountability in Digital Ecosystems*

Paul Malone
TSSG, Waterford Institute of Technology
Cork Road
Waterford, Ireland
pmalone@tssg.org

Brendan Jennings
TSSG, Waterford Institute of Technology
Cork Road
Waterford, Ireland
bjennings@tssg.org

ABSTRACT

Digital ecosystems are distributed software environments through which organisations can seamlessly access customised, potentially disposable, services to aid them carry out a myriad of tasks. Peer to peer networks are often cited as a suitable platform for digital ecosystem deployment. A typical feature of such systems is the lack of a point of control. In this regard these are untrusted environments. This lack of trust acts as a barrier to commercial applications emerging on these platforms. Suitable mechanisms for identity, authentication and trust evolution are required to overcome this. This paper provides a model for distributed accountability in digital ecosystems which can strengthen the trust in the system both from an external viewpoint (i.e. the system as a whole) and between individuals within the system.

Categories and Subject Descriptors

H.4 [Information Systems Applications]: Miscellaneous;
C.2.4 [Computer-Communications Networks]: Distributed Systems; C.2.0 [Computer-Communications Networks]: General—*Security and protection*

Keywords

Digital Ecosystems Infrastructure, Security and Privacy

1. INTRODUCTION

Many in the business and research communities are pursuing the vision of digital ecosystems – distributed software environments through which organisations can seamlessly access customised, potentially disposable, services to aid them

*Permission to make digital or hard copies of all or part of this work or personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MEDES 2009, October 27-30, 2009, Lyon, France.
Copyright 2009 ACM 978-1-60558-829-2/09/0010.\$10.00.

carry out a myriad of tasks. These services can be either pure software applications, or "real world" services represented by a software wrapper supporting automated transaction processing. Full realisation of this vision requires deployment of facilities for the dynamic discovery, composition, interoperation and execution monitoring of a potentially huge number of available services. If we are to provide a distributed digital ecosystem platform for commercial or sensitive applications, then accountability is required to ensure trustworthiness in the platform and thus encourage take-up and usage. The work reported in this paper provides such an accountability facility and is performed under the auspice of the EU FP6 Network of Excellence OPAALS. The model provides for accountability mechanisms between untrusted parties and also takes into account the composed services among those parties.

The rest of this paper is organised as follows – Section 2 provides some background to accountability in computing and information systems. Section 3 provides a description of the distributed accountability model. Section 4 shows how this can operate with a trust overlay network to strengthen trust between players. Section 5 provides a conclusion.

2. BACKGROUND

In recent times the terms "accountability" and "governance" and their inter-relationship have become increasingly popular across a wide range of disciplines. As an indicator of the importance of this relationship, accountability is one of the six World Governance Indicators used by the World Bank in measuring the governance of nations¹. [4] provides a model linking governance, accountability and legitimacy (trust in the system), which is a useful reference point in showing the relationship between these subjects and how changes in governance can ultimately reduce trust in the system.

A useful and regularly cited definition of accountability is given by [14] 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.

¹The Worldwide Governance Indicators (WGI) project, <http://info.worldbank.org/governance/wgi2007/>

2.1 Accountability in Computing Systems

Since the proliferation of computer systems in the 1970s in business and governmental support systems, an acknowledgement for the need of accountability in these systems has steadily grown. This requirement has been further accelerated by the availability and general ubiquity of the Personal Computer through the 1980s and 1990s through to today's widespread availability of broadband Internet access for small businesses and individual domestic users. An early example of this recognised need for accountability in computing systems is [11]. Nissenbaum recognised that a community which insists on accountability where participants are answerable for their actions provides a signal for high quality work and encourages responsible actions among the participants. The absence of such accountability provides a situation where no participant is answerable for risks or harm. She believed that accountability was being undermined in the emerging computerised society and that this undermining was due to three underlying factors rather than emerging computer system proliferation:

1. A narrow understanding of the concept of accountability
2. A set of assumptions concerning the capabilities and shortcomings of computer systems
3. An acceptance that producers of computer systems were not fully answerable for the consequences of the usage of those systems

2.2 Accountability Protocols

The purpose of traditional network security protocols is to provide secure communications in insecure networks. These protocols prevent unauthorised agents or persons from obtaining private data or impersonating another entity. These protocols provide protection where there is trust established between the network resource and the authorised user. They do not provide any protection where there is no such trust. In a scenario where two participants in an electronic exchange of data or services are unknown to each other, more stringent protocols are required to ensure the identity of the participants and to account for their actions during the interaction. One method of strengthening trust and achieving this goal is a preliminary registration process such as SET² or Visa 3-D³. Accountability protocols go further than this and provide the ability to ensure two further elements are present in the interaction [2]:

1. Non-repudiation of actions: neither party can deny their actions in the interaction.
2. Fairness: Both participants receive the expected outcome of the transaction, or neither do.

An accountability protocol provides lasting evidence about actions performed by the peers. In general such protocols

²Secure Electronic Transaction (SET) is a standard protocol for securing credit card transactions over insecure networks, specifically, the Internet.

³Visa 3-D Secure is an authentication technology that uses Secure Sockets Layer (SSL/TLS) encryption and a Merchant Server Plug-in

involve three parties, the two participant peers and a trusted third party (TTP). In a message interchange between A and B, non-repudiation of origin (NRO) provides a validation to B that the message originated from A, while non-repudiation of receipt (NRR) provides validation to A that B received the message. Such non-repudiation is normally brought about through the use of signatures, encryption, notarisation and data integrity mechanisms. [7] provides a detailed description of several fair non-repudiation protocols with no TTP, with inline, online and offline TTPs. Two such protocols using online TTPs are the fair non-repudiation protocol [20] and the certified email protocol [1].

The *fair non-repudiation protocol* ([20]) provides the ability for neither the sender or receiver of a message to obtain an advantage due to the transmission. The protocol makes use of a TTP and the main idea behind the protocol is to split the message definition into two parts, a commitment and a key. The commitment is exchanged between the two participants and the key is lodged with the TTP. An assumption is made that all parties have a private signature key and relevant public keys. A label L links all messages in a session together. Flags (e.g.

f_{SUB}) indicate the purpose of the message. At any of the steps in the protocol, either party can halt the process without giving an advantage to the other, thus providing fairness.

The *certified email protocol* [1] provides non-repudiation for email delivery. Unlike the previous protocol, no public key infrastructure is used in this protocol. While the TTP has encryption and signature keys, A and B only have a password to TTP. In order for the protocol to work correctly A and B must agree in advance on a challenge response mechanism. In the case of the certified email protocol a sender A can send an email to a receiver B , so that B reads the message only if A receives the corresponding return receipt.

In both of the above protocols, the TTP does not see the encrypted message or email, thus reducing the need for strong trust in the TTP. Two further protocols without the need for a TTP are the Markowitch and Roggeman protocol[8] and the Mitsianis protocol [10].

The Markowitch and Roggeman protocol [8] provides a non-repudiation protocol designed to provide probabilistic fairness without the need for a TTP. The protocol is iterative and is such that except at the final iteration, neither entity is at an advantage, thus providing fairness. During the setup, A who wants to send a message m to B , picks a random number n which will determine the number of iterations of the protocol. A keeps the value of n as a secret and selects a set of random $n - 1$ values r_i and a key k , the random values and the key having the same size. Before the final $2n + 1$ step, B did not receive anything anything usable. The only way B can detect whether he received the key is by deciphering c using the value he has received. However, this computation is supposed to be too long compared to the time before A received the EOR and will not continue.

The *Mitsianis* protocol [10] is similar to the Markowitch and Roggeman protocol in that it uses an iterative approach to passing elements from A to B . Initially A sends the cipher c of the message m under a secret key K_δ to B . Once B

acknowledges the receipt of the cipher, A adds a padding of size ω to K_δ to obtain K'_δ . The size of the padding chosen secretly by A . A ciphers this new padded key with a shared session key K_λ and obtains the ciphered key K_ζ . A now chooses secretly a random value n and splits K_ζ into n parts k_i of different random lengths. Now n 2-moves iterations begin. Each iteration involves A sending a k_i (in the order of the split) and B acknowledges the receipt. When B acknowledges receipt of the last parts, k_n of K_ζ , A possesses the NOR of the message m and B can retrieve the session key to recover the message. This is done by composing K_ζ by concatenating all the parts k_i received, deciphered using K_λ giving K'_δ . As B knows the size of the session key K_δ , the padding can be extracted from K'_δ to obtain K_δ and B can decipher the message. Further details of all these protocols are available in the corresponding papers and of other protocols in [7].

2.3 Accountability and Service Oriented Architectures

The rise of Web Services and an increased interest in Service Oriented Architectures (SOA) in general has provided business-to-business (B2B) type transactions with a set of open inter-operable standards through which Internet business transactions can be performed. As an approach to building IT systems, SOA connects applications across a network via a common communications protocol, allowing organisations to reuse old software. Web Services have in fact become the de-facto standard for such B2B interactions. This proliferation of Web Services in providing inter-operable, scalable and composed B2B services has a drawback in terms of providing accountability; there is no standardised approach to providing accountability within the current set of protocols and standards. While there exist specifications on Security [WSS] and on Trust [WST], these are only applicable within trusted domains. This lack of clearly specified accountability models has prompted some work in the area of accountability in Web Services/SOA (e.g. [12], [19], [18]). It has become common to standardise B2B interaction with message-exchange patterns. RosettaNet define a set of externally observable elements of B2B exchange in terms through a set of Partner Interface Processes (PIPs). Typically each message is acknowledged with a confirmation of receipt. Figure 1, below shows the delivery of a business message and its associated acknowledgements. The initial message receipt is acknowledged by B, who in turn sends a message signifying whether the original message was valid or invalid. Upon receipt of this message, A again acknowledges receipt.

Fair Non-repudiable Web Services [12] describes a protocol (which the authors call WS-NRExchange) to enable non-repudiable and fair interactions with Web Services. Their protocol makes use of the non-repudiation protocol of [20] and is based on applying such a protocol to the message exchange pattern shown in Figure 1. This is achieved by the introduction of an inline TTP, the Delivery Agent (DA). This approach provides four types of evidence for non-repudiation.

1. NRO: non-repudiation of origin; msg originated at A
2. NRS: non-repudiation of submission: msg was submitted by A

3. NRR: non-repudiation of receipt: msg was received by B
4. NRV: non-repudiation of validation: msg was validated (positively or negatively) by B

In terms of implementation Robinson et al suggest the use of interceptors to abstract the non-repudiable message exchange from the service endpoints. Most implementations of web services have such interceptors available (e.g. Axis Handlers).

Zhang et al [19] present a model framework for accountability in SOAs which leverage Bayesian Networks in diagnosis of faults and a learning process linked with reputation. This model is more concerned with the self-healing aspect of autonomies through the monitoring of faults and reasoning, than providing a system of non-repudiation service provision.

They recognise that in the real world, business processes are often naturally organised hierarchically. This hierarchical management is efficient in that it aids the large-scale service failure as it facilitates a divide and conquer approach. They present a 3-D approach (Detect, Diagnose and Defuse), which is implemented as follows:

1. **Detect:** Agents monitor behaviour of atomic services according to a predefined and report exceptions to an Accountability Authority (AA).
2. **Diagnose:** The AA makes use of Bayesian Network reasoning to discover the root of the problem when Service Level Agreement (SLA) violation occurs. This reasoning makes use of reported monitoring information gathered at the Detect phase.
3. **Defuse:** The AA updates the reputation of each atomic service based on the outcome of the Diagnosis phase. The AA then re-composes the service through selection of services least likely to violate SLAs, based on reputation.

They designed a Hierarchical Diagnosis Algorithm (HDA) and a Accountability Authority (AA) to which faults are reported in order to address these issues in SOAs.

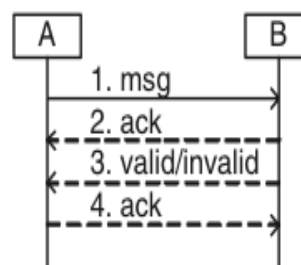


Figure 1: Business Message Delivery Pattern

2.4 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 suitable 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 often 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. The remainder of this section will present research published in the area of accounting and accountability in peer-to-peer networks. While much of the work on accountability in peer-to-peer systems is motivated by a desire to reduce the 'freeloading' which can occur in resource sharing systems, there are also stronger real world economic drivers where these systems are used in sustaining digital communities with service provision.

Karma[16] 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 in resource sharing networks. The system is economic in that it keeps track of the resource purchasing power of peers in the network.

Each peer is represented by a scalar value called a *karma* (much like a community currency) which represents the purchasing power of that peer. If a peer requests a resource, but does not have sufficient karma to purchase the resource, the transaction may not proceed. Participants are thereby incentivised to contribute as many resources (or more) as they consume. 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.

It is assumed that the minimum number of nodes in the network is k . Karma maintains its internal state via a peer-to-peer distributed hash table (DHT). The bank-set $Bank_A$ is a set of peers that independently maintain the karma balance of peer A . Each participant is assigned a unique identifier in a circular identifier space e.g. $NodeID(A)$. The bank set $Bank_A$ is the k closest nodes in the identifier space to $HASH(NodeID(A))$. This mapping is chosen as it allows the implementation to be layered on top of an existing DHT such as Pastry [13]. Using this method the routing to each of the bank-set nodes can be performed efficiently. Each node in the bank-set $Bank_A$ independently stores the karma balance of A , signed by A 's private key (making the value tamper resistant). Each bank-set node also contains a transaction log of recent payments. Karma is not concerned with how the resources are shared or how a price is agreed upon for resource.

When a new node joins the system, a randomly selected public/private key pair is provided and using a value x , such that $md5(Kpublic) = md5(x)$, in the lower n digits, where n is a system wide parameter, the new node's *nodeID* is set to $md5(Kpublic, x)$. The node certifies that this calculation has been performed by signing challenges from its allocated bank-set with its private key. Initially A sends B a signed authorisation for $Bank_A$ to transfer the agreed karma amount to $Bank_B$. B forwards this message to $Bank_B$, which in turn contacts $Bank_A$. If A has sufficient karma to satisfied the transfer amount, the amount is deducted from $Bank_A$'s balance and credited to $Bank_B$. $Bank_B$ then notifies B that the transfer took place and B releases the resource to A . The initial transfer from A includes the balance A will have at the end of the successful transaction (signed by A 's private key). Similarly when B passes this message to its bank-set, it includes a signed version of its own expected balance at the end of the transaction. This mechanism ensures that each bank node contains the latest balance corresponding to their client node.

In 2005 Hausheer and Stiller published a decentralised scheme for accountability in peer-to-peer networks which they called *PeerMint* [6]. The work was performed as part of the MMAPPs 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.

While the accounting data is secure in that it ensures availability and integrity of data, it does not provide mechanisms for confidentiality or privacy. The interface to the accounting mechanisms is generic and can be easily applied to various environments.

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 PeerMint scheme was implemented in MMAPPS using a local instance of FreePastry⁴, an open source implementation of Pastry. The experimental results showed that the scheme's message overhead increased slowly in small networks, but that this overhead levels off as the network grows, thus showing that the scheme provides scalability. No real world implementation is available.

2.5 Accountability versus Privacy

Accountability and privacy bring a tension of interests together. Public accountability requires a transparency that is not always available when private accounting is in place. There is much discussion across many disciplines on this dilemma. See [3],[15],[17] for some discussion related to information technology. While the introduction of a privacy mechanism to ensure no unauthorised access to the accounted data brings clear benefits in terms of protection of business interests it also brings about the case where entities' reports of experience actions are not verifiable by the community as a whole. This in turn leads to a case where experience reports generated to evolve trust in transacting entities cannot be verified as accurate by the mediators or the account holders. With this in mind, it is the part of the purpose of protocol development in this task to cater for a model capable of performing in a publicly accountable manner in addition to a private accountability one.

3. ACCOUNTABILITY MODEL DESCRIPTION

3.1 Accountability Scenarios

There are three separate Accountability scenarios which need to be supported by our model.

1. *No Accountability*: This is the simplest case where a service or data consumption is open and there is no need for accountability (e.g. accessing web pages on a public website). This is the absence of accountability and is a trivial case but the need for this prescribes the case that accountability is optional and is in operation on a per-service case.
2. *Public Accountability*: This is the case where service or data consumption is to be accounted for and the usage data describing the course of the transaction is open and available to anyone. This is useful in the usage of accountability data in verifying trust updates when incorporated with the OPAALS Trust Model.
3. *Private Accountability*: This is the case where the service or data consumption is to be accounted for but the

usage data describing the course of the transaction is only available to the participating parties. This data is less useful in the verification of trust updates as the semantics of what the data contains cannot be substantiated by third parties. However the provision of this type of mechanism is necessary for the sensitivity of the usage data.

3.2 Requirements

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 all 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 (shown in Figure 2) shows two peers (*A* and *B*) interacting in a peer-to-peer services trading environment. The model is influenced by the *PeerMint* model published by [6] combined with an *Accounting Authority* to cater for composed services similar to that of [19]. The idea here is to take the concept of the *Accounting Authority* introduced by Zhang and to deploy this as a distributed service and combine this with the distributed PeerMint model. Although the diagram shows a simple transaction between 2 services, the introduction of the Accounting Authority provides the

⁴FreePastry, open-source implementation of Pastry, <http://www.freepastry.org/>

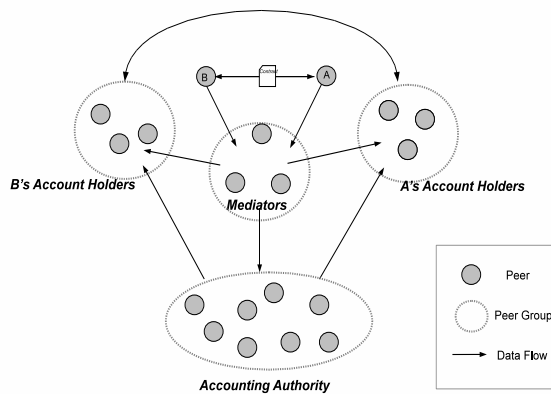


Figure 2: Distributed Accountability Model showing Actors and data flow.

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.

3.3 Reducing the ability for group infiltration

In order to reduce the possibility of cheating, it is important that sets of *Account Holders* or *Mediators* cannot be infiltrated by either participating party or by a third party. In the case that *A's* Account holders identities can be pre-determined it might be possible for an entity to infiltrate this set of nodes and inject fraudulent data into the nodes. A non-deterministic periodic issue of a randomly created nonce by each node to perform a recalculation of its *Account Holders* would make this very difficult to achieve and can be considered a suitable mechanism against infiltration. Mediator selection is done on a per-session basis. When a session begins Mediators are assigned to the task of checking accounted data for consistency. At the end of the session, the set of Mediators is released. The selection of Mediators is performed by performing a hash function on the combination of both peers' identities combined with a processID, serviceID and a timestamp to ensure that Mediator selection cannot easily be pre-determined by potentially rogue peers.

3.4 Securing the Accounted Data

Securing the data relies on a PKI facility being in place. Each node has a public/private key pair which can be used to create shared keys (e.g. Diffie-Hellman ([5])). When *B* wishes to communicate with *A* it sends its public key together with a time-stamped process identifier. If *A* wishes to continue the communication, it returns its public key together with the time-stamped process identifier. Now each party can use their private keys in combination with their respective public keys to create a temporary shared key with is then further hashed with the agreed process id to create a shared key for the session. This shared key is kept private to each party and is not transmitted. This shared key is then used for three purposes, securing a channel of communication, selecting a set of mediators and encrypting accounted data.

3.5 Actors

The model comprises Actors and Protocols for transmission of messages. There are four Actors in the model. Their roles are explained here.

Peers

The Peers are responsible for the creation of the evidence and the delivery to the Mediators. Peers can also act as Mediators, Account Holders and members of Accounting Authorities.

Mediators

The Role of the Mediators is to collect the evidence from each Peer and compare both records for consistency. Alarms are raised when disagreement is discovered. Mediators also release periodic digests to Account Holders and Accounting Authorities. Selected Peers take on the Mediator role for the duration of the session or transaction. The Mediators can be seen as a distributed type of trusted third party (TTP) except no pre-established trust is needed.

Account Holders

The Account Holders are responsible for the delivery, persistence and retrieval of accounted data. Account Holders are more long-lived than Mediators and Peers are periodically reselected through the issue of a nonce by the Peer for which they persist data.

Accounting Authorities

The Accounting Authority deliver service composition relative data to Account Holders for persistence as well as raising alarms in the case of service composition inconsistencies. Peers operate as Accounting Authorities when they are Mediators and Peers of a service composition scenario.

3.6 Protocols

Two protocols are presented, one for accountability of public data and one for private data exchange. The protocols are similar except that in the case of the private accountability the messages are encrypted with a shared secret.

Public Accountability Protocol

1. *A* requests service access from *B*
2. *B* grants access and provides *A* with its public key and a signed contract.
3. *A* acknowledges receipt and acceptance of the service contract, signs it and sends *A* its public key. The set of Mediators is now calculated using a pre-agreed hash function on a combination of *A* and *B's* Ids, the time stamp and the process ID. The contract and other relevant information is lodged with the mediators.
4. *A* consumes the service and each message sent and received is signed with *A's* private key, then lodged with the Mediators. *B* also signs each message and sends to the Mediators. The Mediators examine each pair of messages and compare for agreement. In the case of disagreement, both *A* and *B* are notified of the disagreement by the Mediators.
5. When service consumption ceases both *A* and *B* send a final statement to the Accounting Authority. Also,

Step	Flow	Message
1	$A \rightarrow B$	B, T, S
2	$B \rightarrow A$	$B_{k_{pub}}, A, Con_S, T, S, P$
3	$A \rightarrow B$	$Con_{ACK}, B, A_{k_{pub}}, S$
4a	$A \leftrightarrow B$	M_{signed}, A, B, P
4b	$A, B \rightarrow M$	M_{signed}, A, B, P
5a	$A, B \rightarrow AccA$	R_{final}, A, B, P
5b	$M \rightarrow A_{ACC}, B_{ACC}$	M_{digest}, A, B, P

Table 1: Public Accountability Protocol

the Mediators send a digest of the accounted data to both A's and B's Account Holders where the data is persisted.

The algorithm as it is written above considers full public accountability. The messages are not encrypted and are open to everyone. It is possible to insert enciphering and deciphering of the messages to ensure more confidentiality. If this is done the Mediators would require the public keys to ensure that the data can be verified semantically

The protocol is described more formally in Table 1.

Private Accountability Protocol

1. A requests service access from B
2. B grants access and provides A with its public key and a signed contract.
3. A acknowledges receipt and acceptance of the service contract, signs it and sends A its public key. The set of Mediators is now calculated using a pre-agreed hash function on a combination of A and B's Ids, the time stamp and the process ID. Both parties generate a secret using the other's public key and their own private key. The contract and other relevant information is lodged with the Mediators.
4. A consumes the service and each message sent and received is encrypted with the secret and signed using A's private key, then lodged with the Mediators. B also encrypts and signs each message and sends to the Mediators. The Mediators examine each pair of messages and compare for agreement without being able to examine the contents of the encrypted message. In the case of disagreement, both A and B are notified of the disagreement from the Mediators.
5. When service consumption ceases both A and B send a final statement to the Accounting Authority. Also,

Step	Flow	Message
1	$A \rightarrow B$	B, T, S
2	$B \rightarrow A$	$B_{k_{pub}}, A, Con_S, T, S, P$
3	$A \rightarrow B$	$Con_{ACK}, B, A_{k_{pub}}, S$
4a	$A \leftrightarrow B$	M_C, A, B, P
4b	$A, B \rightarrow M$	M_C, A, B, P
5a	$A, B \rightarrow AccA$	R_{final}, A, B, P
5b	$M \rightarrow A_{ACC}, B_{ACC}$	M_{digest}, A, B, P

Table 2: Private Accountability Protocol

the Mediators send a digest of the accounted data to both A's and B's Account Holders where the data is persisted.

The protocol is described more formally in Table 2

4. TRUST AND ACCOUNTABILITY

There exists a strong relationship between trust and accountability. The existence of system accountability mechanisms can provide a means of measuring trust in the system and trust between entities operating within the system. [4] provides a model linking governance, accountability and legitimacy (trust in the system). If we are to provide a distributed platform for commercial or sensitive applications, then accountability is required to ensure trustworthiness in the platform and thus encourage take-up and usage.

4.1 Trust Manager Overlay Model

A Trust Model suitable for digital ecosystems has been developed for OPAALS and is described in [9]. The approach is to use a trust overlay network for providing a community based approach to trustworthiness based on the reputation of entities. The entity can represent a node, service, resource, a service provider or a service consumer. Each entity has a Trust Manager associated with it. The entities gain experience from interacting with other entities and publish reports of these experiences to the Trust Manager. Using a pre-defined context dependent algorithm the Trust Manager updates the entities' local trust and based on a policy of sharing trust information provides trust updates to other Trust Managers in the overlay network providing a reputation based trust overlay network.

4.2 The Use of Accountability Data in Evolving Trust

Accountability data is a source of experience for trust manager overlay network. There are several ways in which the data can be used for gathering Experience Reports for trust evaluation.

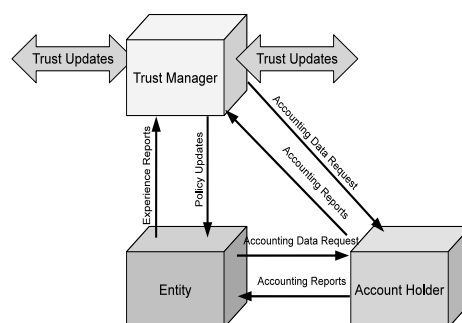


Figure 3: OPAALS Trust Overlay and Accountability Model

- Accountability data can be used directly as Experience Reports. Algorithms can be developed to operate directly on the accounted data and published to the Trust Manager. See Figure 3 for how these two models integrate.
- Retrieved data from an Account Holder can be analysed by the Entity and used as an input to pre-published trust algorithms.
- Access to accountable data can be inserted in algorithms and the Trust Manager can run the algorithm against the resultant query.
- Alarms raised by Mediators and Accounting Authorities can also be used for the generation of trust. These alarms are reported in the accounted data.

Other sources of experience reports for trust can also be checked against the accountability data. For example, if a peer's trust is degraded due to false reports, the accountability framework provides a means of the peer disputing the false report with accountable evidence. In this regard, private accountability presents a situation where false reports cannot be defended against without revealing private keys to view the evidence.

5. CONCLUSIONS AND FURTHER WORK

This paper has described a distributed accountability model for peer to peer digital ecosystems. The model takes a protocol approach and is influenced by traditional accountability protocols in this regard. Protocols for public and private accountability are provided and the model allows for composed service accountability which is considered one of the requirements of digital ecosystems. Finally, how this model integrates with the OPAALS distributed trust model was described.

The model is being implemented in the FP6 OPAALS Network of excellence. Work is ongoing in integrating the accountability model implementation with the distributed entity and distributed trust model described in this paper.

6. ACKNOWLEDGMENTS

This work is funded by the EU FP6 Network of Excellence OPAALS, <http://www.opaals.org>

7. REFERENCES

- [1] M. Abadi and N. Glew. Certified email with a light on-line trusted third party: design and implementation. In *Proceedings of the 11th International Conference on World Wide Web (WWW-02)*, pages 387–395. ACM Press, 2002.
- [2] G. Bella and L. C. Paulson. Accountability protocols: Formalized and verified. *ACM Transactions on Information and System Security*, 9:138–161, 2006.
- [3] D. Brin. *The Transparent Society: Will Technology Force Us to Choose Between Privacy and Freedom?* Perseus Books, 1999.
- [4] G. Bullock. *Governance, accountability, and legitimacy*, 2006.
- [5] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, (22):644–654, 1976.
- [6] Hausheer and Stiller. PeerMint: Decentralized and secure accounting for peer-to-peer applications. In *Networking: Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications, International IFIP-TC6 Networking Conference, LNCS*, volume 4, 2005.
- [7] S. Kremer, O. Markowitch, and J. Zhou. An intensive survey of fair non-repudiation protocols. *Computer Communications*, 25(17):1606–1621, 2002.
- [8] O. Markowitch and Y. Roggeman. Probabilistic non-repudiation without trusted third party. Technical Report 399, ULB, 1999.
- [9] J. McGibney and D. Botvich. A trust overlay architecture and protocol for enhanced protection against spam. In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, pages 749–756, 2007.
- [10] J. Mitsianis. A new approach to enforcing non-repudiation of receipt, 2001.
- [11] H. Nissenbaum. Computing and accountability. *CACM: Communications of the ACM*, 37, 1994.
- [12] P. Robinson, N. Cook, and S. K. Shrivastava. Implementing fair non-repudiable interactions with web services. In *EDOC*, pages 195–206. IEEE Computer Society, 2005.
- [13] A. Rowstron and P. Druschel. Pastry, scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of IFIP/ACM Middleware*. Springer, 2001.
- [14] A. Schedler. *Conceptualizing Accountability*, pages 13–28. Lynne Rienner Publishers, London, 1999.
- [15] F. B. Viegas. Bloggers' expectations of privacy and accountability: An initial survey. *Journal of Computer-Mediated Communication*, 2005.
- [16] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer. Karma: A secure economic framework for peer-to-peer resources. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.
- [17] P. Winn. Online court records: Balancing judicial accountability and privacy in an age of electronic information. *Washington Law Review*, 2004.
- [18] Y. Zhang and K.-J. Lin. Hierarchical management of service accountability in service oriented architectures. In *SOCA*, pages 55–64. IEEE Computer Society, 2007.
- [19] Y. Zhang, K.-J. Lin, and T. Yu. Accountability in service-oriented architecture: Computing with reasoning and reputation. In *ICEBE*, pages 123–131. IEEE Computer Society, 2006.
- [20] Zhou and Gollmann. A fair non-repudiation protocol. In *RSP: 17th IEEE Computer Society Symposium on Research in Security and Privacy*, 1996.