Title: Security Specification

Editor: John Ibbotson

Authors: John Ibbotson (IBM)

Victor Tan (UoS)

Reviewers: All project partners

Identifier: D4.2.1

Type: Deliverable

Version: 2.0

Date: 17th February 2006

Status: Public

Summary

The purpose of this document is to further develop the logical security architecture for a provenance aware system first introduced in [IHT05], with the primary intention of demonstrating an approach towards its application and implementation utilizing Web Services / Grid technologies in the context of the two project application domains of aerospace engineering and organ transplant management.

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

Members of the PROVENANCE consortium:

- IBM United Kingdom Limited United Kingdom
- University of Southampton United Kingdom
- University of Wales, Cardiff United Kingdom
- Deutsches Zentrum fur Luft- und Raumfahrt eV. Germany
- Universitat Politecnica de Catalunya Spain
- Magyar Tudományos Akadémia Számítástechnikai és Automatizálási Kutató Intézet Hungary

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

Foreword

This document has been edited by John Ibbotson (IBM) based on input from project partners.

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

Table of Contents

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

1 Overview

One of the key features for a provenance architecture within the context of this project is security. Many of the application domains in which a provenance architecture could potentially be deployed have stringent requirements on access to data. Correspondingly, process documentation that incorporates or are derived from these data are likely to share similar security restrictions as well. In particular, the security requirements for the application domains of organ transplant management and aerospace engineering have been enumerated [And05b], and a security architecture that justifies these requirements has being developed in [IHT05] and further refined in [GJMM06]. We can now proceed in this document to detail a possible mapping of the initial security architecture to an actual implementation based on Grid / Web Services technologies within the context of the two target application domains (aerospace engineering and organ transplant management).

A discussion of various general provenance related security concerns was provided in both [GJMM06] and [IHT05] as a precursor to the development of the security architecture in these documents. In Section 2 of this document, we discuss these concerns again, but limit them to those that are immediately relevant to the requirements of the application domains and which we intend to address. The security architecture for the provenance store described in [GJMM06] is presented again in Section 3, along with details of interaction between its various constituent components. In Section 4, we discuss the security issues associated with the other components of the provenance architecture as well as security aspects salient to scalability considerations (such as distributed provenance stores). The security functionality provided by the Globus Toolkit container environment, the chosen deployment environment for the provenance store [HIB06], is then described in Section 5. An approach for utilizing this functionality to realize the security architecture for the provenance store as well other related security concerns such as federation is described in Section 6. We conclude in Section 7 with a summary of this document, along with an indication of future work that will be presented in the final security deliverable.

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

2 Provenance Related Security Issues

An introductory discussion of the relevant security concepts and terminology is provided in Section 4.1 of [GJMM06]; the reader is referred to this document for further details. In this section, we recap the provenance related security issues discussed in that document, but limit our review to the issues that are immediately relevant to the security requirements of the two application target domains that we wish to address. We note that our intention to limit the issues we address was already present and stated explicitly in [GJMM06].

1. Access control to the provenance store. This is the primary security issue as the provenance store is considered to be central to the logical architecture. While the access control mechanisms utilised are situated in the context of the specific requirements of the project, this notion of security here is conceptually identical to the general case of securing a database with multiple users.

2. Integrity and non-repudiation of p-assertions. Recording actors store p-assertions created by asserting actors in the provenance store. In the event that the asserting actor is not the recording actor, there is a need to ensure that information within the p-assertion is not altered unintentionally or maliciously by either the recording actor or provenance store. This can be achieved by having the asserting actor sign the p-assertion it creates. The signature also serves the additional purpose of ensuring that the asserting actor cannot deny responsibility for the creation of the p-assertion in question. This can be necessary when legal or other requirements mandate establishment of liability for the consequences arising from utilising the information in a p-assertion.

3. Ascertaining asserter identity in a p-assertion. By implication of the previous point we discussed, the asserter identity should correlate with the identity associated with the signature on the p-assertion, since only the asserting actor should sign the p-assertion. A check can be done to ascertain whether this is true, and can be undertaken by either the provenance store to which the p-assertion is recorded to, or by the querying actor retrieving the p-assertion in question.

4. Delegation of identity or access control. The logical architecture involves a number of components that interact with each other, the provenance store as well as potential users of the provenance system. All of these interactions may be restricted by security considerations in the same manner as interactions with the provenance store will be. Situations may arise where a component may wish to delegate its identity or access rights to another component for a temporary period of time to allow the second component the authorisation to perform a specific task on its behalf.

5. Federated security for distributed provenance stores. As discussed in [GJMM06], p-assertions can be stored across distributed provenance stores, which themselves may be located in different security domains. A potential security issue arises if the querying actor is not recognised or does not have the necessary authorisation to retrieve the relevant p-assertions in the security domains of all these stores. Some form of federation must be accommodated to allow different domains to communicate identity and authorisation information between each other.

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

6. Verifying integrity of referenced data. In [GJMM06], the issue of scalability with respect to data contained with a p-assertion is discussed. In particular, when the amount of data to be stored in a p-assertion is inordinately large, a reference can be used instead in the p-assertion which points to the location of the actual data. If this reference is later resolved and used to retrieve the actual data, an assurance must be provided that the retrieved data had not been altered in any way.

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

3 Provenance Store Security Architecture

The logical design for a security architecture for the provenance store was first described in [IHT05] and further refined [GJMM06]. We present it here again to provide continuity to the new material that follows in Section 5 and 6. An overview of this architecture is illustrated in Figure 1; components enclosed in ovals indicate that they potentially (although not necessarily) exist in security domains separate from the domain of the provenance store. We first describe the functionality of each of these components and then proceed to outline the possible interactions between them. Finally, we discuss some of the broader security issues that are not considered in this architecture.

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

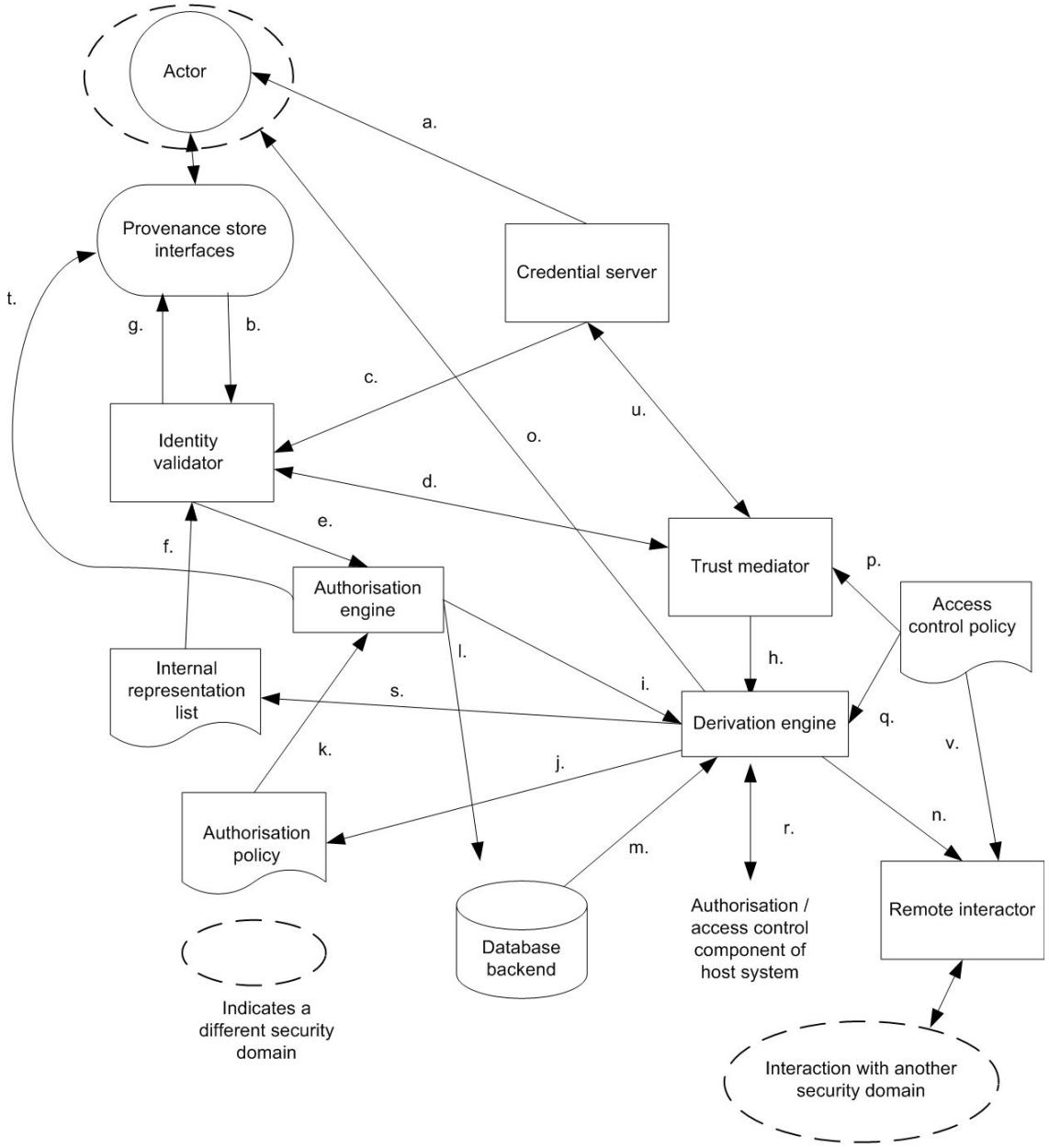

Figure 1: Provenance store security architecture

3.1 Components of the Security Architecture

The provenance store exposes three different interfaces (recording, management, query) for different purposes. All of these interfaces can be enhanced with additional security relevant operations or parameters. The identity validator accepts all incoming requests and accompanying credentials (such as certificates) over a secure link supporting either transport or message level encryption. It is also important that the validator and actor interacting with it mutually authenticate each other during this secure transmission. This is necessary from the viewpoint of the provenance store as the identity of the

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

actor is the first step towards enforcing appropriate access control. However, it is equally relevant as well for the actor who needs to circumvent potential impersonations of a valid provenance store by malicious parties that would then gain unauthorised access to the p-assertions.

The identity validator then performs four functions:

1. Verifies that the submitted credentials are valid within the context of the domain. This may require interaction with the trust mediator in the event where federated identity validation is required. It also needs to take into account that the submitted credentials may incorporate some form of delegation.
2. Maps these credentials to an internal representation (IR). This could assume a combination of various forms (an identity, a role, a list of attributes, a list of privileges, etc). This should include basic role information to support an RBAC implementation. A common way of doing this is to map the identity to a role which has a predefined set of authorisations or privileges.
3. Ensures that the asserter identity on the submitted p-assertion tallies with the identity associated with the signature, if any, on the p-assertion. This operation is optional, and correlates with the third security issue in Section 2. If the asserter identity is to be utilised in the access control decision, then it needs to be mapped to a corresponding IR as well.
4. Formats the request into an appropriate representation for access control purposes.

The first two functions are performed with help from an internal representation list that specifies the appropriate mapping relationships, including roles.
The credential server fulfils the role of being a trusted third party holding identity-related information for all potential users of the provenance system within a given security domain, as well as providing them with suitable credentials and other related security tokens for authentication purposes. The authorisation engine essentially performs the access control functionality in two main ways based on the authorisations specified in the authorisation policy and the IR produced from the identity validator.

- The request is granted or denied solely on the basis of the information from the authorisation policy and the IR related to the identity of the requesting actor. It is also possible that the IR of the asserting actor is taken into account in the access control decision as well; we assume that this possibility exists, but use the term IR to refer to the IR of the recording actor for the sake of brevity in the remaining discussion. If granted, the requested operation is performed and the appropriate acknowledgement or data item is returned directly to the requestor without further intervention from the authorisation engine.

- The granting of the request may additionally be dependent on information contained within the data item that the request is related to (such a condition would be specified accordingly in the authorisation policy). For example, a read operation associated with an IR on a given p-assertion might be permitted only if the p-assertion contained relevant information pertaining to that IR. In this case, the p-assertion in question would have to be retrieved first and assessed accordingly by the authorisation engine before a final decision can be made on granting or denying the request.

Depending on the nature of the authorisation engine, it may be necessary that the assignment of a role to an IR for the case of a RBAC should be achieved by the authorisation engine instead of the identity validator. In addition, it is possible to employ either one or both of these two approaches to specifying authorisation:

- an identity / role is assumed to have no authorisations in the initial case, and explicit authorisations have to be granted;

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

- an identity / role is assumed to have complete authorisation in the initial case, and explicit restrictions have to be placed.

The semantics and the granularity of the authorisation assertions within the authorisation policy will determine how fine grained and flexible access control can be in the infrastructure. Ideally, authorisations should be specifiable at the level of individual p-assertions, and could be refined to individual elements within a p-assertion if such need arises.

The access control policy is a higher level security policy that specifies the ways in which the authorisation policy and/or internal representation list can be modified by the components which access them. It also describes in a high level manner the configuration of the provenance store from a security viewpoint and the protocols that external entities (such as actors or other provenance stores) need to adhere to in order to communicate with it. The access control policy, authorisation policy and internal representation list would constitute the security policy of the provenance store as a whole. The database backend provides actual physical storage for the p-assertions.

The trust mediator is used to support federation of authentication and/or authorisation for the case where distributed provenance stores exist in different security domains. Its role is to obtain security assertions or credentials from other relevant entities in order to support the specific federation methodology employed. These entities could be a trusted third party (such as the credential server) in the local or remote security domain, or the trust mediator of another provenance store. The gathered assertions or credentials can then be used locally (for example, to verify other credentials received by the identity validator) or can be passed on to the remote interactor. Like the trust mediator, the remote interactor is intended to interact with external entities, but within a more general context rather than a security specific one. The remote interactor uses the credentials provided by the trust mediator for secure communication towards this end. The operational parameters for both the trust mediator and remote interactor can be configured via the access control policy.

If there is additional information contained within the p-assertions that are being retrieved which can subsequently effect the authorisation decision, then some form of checking or filtering must be performed on these p-assertions before (or if) they are returned to the user. This is the functionality of the derivation engine. The derivation engine may additionally need to interact with other external systems in order to perform specific tasks related to the storage and retrieval of p-assertions. It will perform these via the remote interactor.

3.2 Interaction Between Components

We illustrate the interaction between these components using some simple scenarios in a technology independent manner. The flows of information are denoted by labelled arrows in Figure 1 and our description makes reference to them accordingly in brackets.

Scenario 1: Submission of a p-assertion to be stored by a recording actor

1. The p-assertion along with other relevant information is submitted as a an invocation message (b.) in accordance to the schema of the recording interface. The submission link is secured using appropriate actor side library functionality, which may involve encryption or signing of the invocation message. The required credentials can be obtained (if necessary) from the credential server prior to the invocation process. (a.).

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

2. The identity validator intercepts the message and attempts to verify the submitted credentials; again this may involve another interaction with the credential server (c.). An attempt is made to resolve the supplied credential information with the internal representation list (f.) If the credentials cannot be verified, but there is additional information present to indicate the manner in which they might be verified, an appropriate request is sent off to the trust mediator (d.).

3. Depending on the methodology used for federation purposes (see Section 5.7), the trust mediator may opt to contact a trusted third party (the credential server or another entity) within the current or remote security domain (u.) to obtain the necessary credentials to communicate with the party that is capable of verifying the actor's credentials.

4. The external communication with the designated party is handled by the remote interactor using the credentials obtained by the trust mediator. If the result of this interaction is the verification of the actor's credentials, there may arise a need to add new updates to the local IR or authorisation policy. Such a decision (if required), is made by the derivation engine (h.), which will in turn make the appropriate additions to the internal representation list (s.) and the authorisation policy (j.). The entire operation is mediated using the guidelines specified in the access control policy (q.)

5. If the credentials fail to be verified successfully, an appropriate security exception is returned via a fault (g.). Otherwise, the validator converts the store request into an appropriate format and sends it off to the authorisation engine (e) along with the role and accompanying security attributes.

6. The authorisation engine first needs to ascertain whether the store request is valid for the specified role based on the authorisations specified in the access control policy (k.). If it is not, an appropriate security exception is again returned via a fault (t.). Otherwise, new authorisation information for the p-assertion to be stored needs to be determined. The authorisation engine formulates an appropriate statement which is then sent off to the derivation engine (i.). For the case of the third security issue mentioned in Section 2, the submitted p-assertion will also contain accompanying authorisation information; this will be also be taken into account in the formulated statement of the authorisation engine.

7. The new authorisation for the p-assertion to be stored is added to the authorisation policy (j); this can be either achieved dynamically or statically after the fact. The p-assertion is now sent onwards to the database backend, along with relevant authorisation information or metadata that it is meant to be stored with (l.).

8. An acknowledgement (as well as other pertinent information) is returned to the derivation engine (m.), which is processed accordingly and another acknowledgement returned to the recording actor (o).

Scenario 2 : Retrieval of a p-assertion by a querying actor

The sequence of interactions is nearly identical to that for the case of storage. The primary difference arises from the fact that there is no need to generate new authorisation information as there is no new p-assertion to be stored. However, when the requested p-assertion is returned (m.), further transformations may be performed on it, in accordance to the initial authorisation information associated with the request as well as any additional authorisation information stored and associated

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

with the p-assertion itself. The transformations which are undertaken by the derivation engine, may take the form of filtering out portions of the p-assertion or transforming the information in the p-assertion in some specific manner.

Depending on various factors such as the particular deployment of provenance stores, the p-assertions returned and the nature of the request made by the querying actor, there may arise a need for the provenance store to engage in remote interactions with other systems (potentially located in different security domains as well) in order to return a satisfactory result set to the querying actor. This activity is undertaken by the remote interactor (n.), which will procure the necessary security credentials for its interaction from the trust mediator. Again, this operation is coordinated under the guidelines specified in the access control policy (p., q., v.).

Scenario 3: Management of the provenance store by a managing actor

Management operations on the stored p-assertions are achieved in an identical manner to that for scenario 1 and 2. Submission of a management operation request is treated like the submission of a p-assertion to be stored, with the difference that the management request is not stored but rather processed by the derivation engine and the appropriate functionality then enacted. This may require retrieval of p-assertions, if so, these are then returned to the managing actor in a similar manner to that in Scenario 2. There may also be modifications of internal representation list/authorisation policy which may include deletion, modification and addition of entries. All of these operations are consequent on the identity validator first recognising that the authenticated managing actor has the role or capability to perform these management type activities.

Scenario 4 : Integrating authorisations of the provenance store and the host system

This can be accomplished by providing a link / interface between the access control /authorisation components of the host system and the derivation engine. If the provenance store architecture is tightly integrated with its host system, this link may not need to be secured as all communications between the architectures are internal within the operating system, rather than through an exposed network medium. Changes that need to be made to the authorisation policy / internal representation list can then be propagated through the derivation engine (r).

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

4 Security in Other Architecture Components

In the previous section, we presented and described the functioning of a security architecture to protect the provenance store, a key component of the logical architecture. Here, we study the security considerations underlying interactions involving other components of the logical architecture.

4.1 Between other components and the provenance store

The other components in the logical architecture that interact directly with the provenance store will now require corresponding security functionality as well in order to ensure their interactions are secured properly. We describe the nature of the required functionality below for application services, management UIs and processing services.

1. A facility is required for accessing credentials that are to be submitted to the identity validator in the provenance store. This can be provided as additional libraries in the corresponding actor side libraries [GJMM06] or as interfaces that permit interoperation with external third party applications that provide credential generating functionality. A straightforward example would be a keystore manager application that generates, archives keys and certificates and obtains approval for these certificates from a CA.

2. If a keystore or some other facility for storing cryptographically generated material is to be used by the actor side libraries, it has to be secured appropriately (e.g. located in a secure account, encrypted and contents accessible only by the provision of a username/password combination).

3. A facility is required for accessing specific security mechanisms such as signing or time stamping. This is necessary, for example, when the asserting actor needs to sign the p-assertion it created.

For the case where authorisation information is desired to be submitted alongside p-assertions, an interface must be provided as part of the domain specific services that allows the retrieval of this information from the appropriate locations (such as a local database). This interface should be congruent with the specific format in which the authorisation information can be expressed in.

4.2 Intermediate components

By intermediate components, we refer to components that are not directly accessible by potential users of the provenance system. Such components may themselves be invoked or accessed by other components rather than by the user, and may interact directly with the provenance store. For example, a user may use a presentation UI to access a presentation service which in turn accesses the provenance store. In the application domain, a user may access an application UI that in turn invokes a chain of other application services before a final invocation is made to the provenance store. In such cases, the intermediate component may require authentication of incoming requests to it. It is possible to reuse the security architecture developed for the provenance store for this particular component as well. The primary differences would be, with reference to Figure 1, are:

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

1. As the incoming request is to the intermediate component, it is unlikely to be a p-assertion, rather a generic data item (which may contain a p-assertion) submitted in accordance with the schema of the interface to this intermediate component.

2. Once the request is approved by the authorisation engine, it is sent off (l) to some internal function of the intermediate component for further processing, rather than to a database backend (as is the case for the provenance store). Once this processing is complete, a result is returned to the invoking actor (m) and / or a further invocation is made to another component.

4.3 Delegation of identity or access control

The need to delegate access control may arise if the intermediate component described previously exists in a separate security domain from both the user and the provenance store. Consider again the logical architecture in Figure 1 and assume that a user is performing a query on the provenance store through the presentation UI and a processing service. Assume now three separate security domains: one containing the user and the presentation UI, the second containing the processing service, and the third encapsulating the provenance store.

When the presentation UI under the users control sends a request to the presentation service, an appropriate credential is submitted by the user for purposes of authentication. If the request is authorised, the presentation service will then decide the type and number of provenance store queries that need to be made in order to satisfy the request. When making these queries, the presentation service needs to present suitable authentication credentials to the provenance store. There are essentially two ways to proceed here:

- Authenticate to the provenance store using the credentials of the presentation service, whereupon subsequent authorisation decisions will be based on the identity or associated role of the presentation service. This approach requires the presentation service to be trusted and known to the provenance store security administrators, and that it has the appropriate authorisation to access a wide enough pool of p-assertions to satisfy requests from all potential users (or at least users that are known within the security domain of the presentation service).

- Authenticate to the provenance store on behalf of the original user. This approach requires that a form of delegated identity or access control credential be created by the presentation service, possibly in negotiation with the presentation UI. The identity validator of the provenance store must then be able to recognise and process this delegated credential accordingly, and infer the identity or associated role of the original user. Subsequent authorisation decisions are then on the basis of the users identity, and may also need to take into account additional constraints specified in the delegated credential itself.

The first approach is suitable if all potential users making queries can ever only do so through the medium of a presentation service. Here, the responsibility of checking authorisations for the actual users is effectively offloaded from the provenance store to the various presentation services in the system. If the number of presentation services known within the provenance store security domain is significantly smaller than the potential number of users, then the overhead of authorisation is equivalently reduced as there is now only a need to check on these presentation services.

The second approach will incur an overhead associated with communication between the presentation UI and the presentation service in order to create an appropriate delegation credential. Depending on

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

the delegation act itself, there may be a need also for further communication between the security architecture of the provenance store and the user / presentation UI during the authentication or authorisation process in the security architecture of the provenance store. This might happen, for example, when delegating access control is expressed through the modification of the authorisation policy in the provenance store to reflect the delegation of authorisations between the security domains of the user and the provenance store.

Even when credential delegation is used, the presentation service may also have an installed security policy that dictates the nature of the results to be returned to the user / presentation UI. For example, assume that a request from the user to the presentation service results in several corresponding query requests being sent in turn to the provenance store along with a delegated credential. P-assertions pertaining to the authorisation associated with this credential are then returned to the presentation service. At this point, the security policy of the presentation service as pertaining to the user in question may dictate further processing of the results (such as transforming or filtering it in some way) before finally returning it to the user. In this case, filtering or transforming of the returned results based on authorisation considerations happens at two stages: once at the provenance store, and then subsequently at the presentation service. In both stages, it is performed by the derivation engine of the respective security architecture. There may also be need to communicate between the authorisation engines of both the presentation service and provenance store via their respective remote interactors and trust mediators, if complex authorisation decisions are to be affected.

The description in this subsection is equally applicable to intermediate components in other places in the logical architecture, for example with an application service that is located in a different security domain from the actual application service that makes the final submission of p-assertions to the provenance store. Similarly, delegation of identity or access control can also occur multiple times if there is an invocation of a chain of application services (such as that might occur in a workflow), with all these services located in different security domains. In cases like this, it is necessary to ensure that the delegation mechanism being used (for example, proxy certificates [WFK04]) can support multiple acts of delegation.

4.4 Security issues relating to scalability

The architecture document [GJMM06] discusses the need for distributed provenance stores in the context of scalability reasons. The incorporation of distributed provenance stores implies that related p-assertions can now be stored across several provenance stores, each potentially located in separate security domains. A potential security issue arises if the querying actor is not recognised or does not have the necessary authorisation to retrieve the relevant p-assertions in the security domains of all these stores. Consider a querying actor submitting a complex query to the provenance store, whereupon the querying functionality determines the necessary p-assertions required to satisfy the query and attempts to retrieve them from the provenance store. In the event that not all of the required p-assertions can be found locally, there are two possible ways to proceed:

- The querying functionality itself can use the links in the available p-assertions to locate other distributed provenance stores which it subsequently needs to query. Thus, the querying functionality queries the distributed provenance stores on behalf of the querying actor.

- The querying functionality can return the available p-assertions to the querying actor, which then itself has to navigate the links and make queries to the relevant distributed provenance stores.

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

In the first approach, the provenance store to which the original query is directed needs to interact with other provenance stores, possibly in other security domains. This interaction will be handled by the remote interactor using credentials (if necessary) acquired by the trust mediator in the manner previously detailed in Scenario 2 in Section 3.2. The remote interactor may also choose to use the delegated credentials of the querying actor in its interaction with the remote provenance store, if so desired.

In the second approach, the querying actor will need to obtain the necessary credentials in order to communicate with the remote provenance stores, possibly through a credential server. Alternatively, depending on the federation methodology employed, the querying actor could attempt to authenticate directly using its own credentials and the remote provenance store would attempt to verify its credentials in the manner outlined in Scenario 1 in Section 3.2.

P-assertions may also be copied or moved between stores that are located in different security domains (i.e. for the staging of data [GJMM06]), the access control restrictions on them in their new destinations needs to be defined. In the simplest case, the newly moved or copied p-assertions retain the same access control restrictions that were associated with them in their original domain. These restrictions can be provided as authorisation information along with the p-assertions as they are recorded to their new destinations.

If the authorisation information involves identities from the originating domain that are currently unknown in the destination domain, then this identity information needs to be communicated between the trust mediators of both domains. The communication can be performed when the p-assertions are initially recorded, or at a later time when a request is made to the provenance store from an entity that is not recognisable in the new provenance store domain. The process of moving p-assertions between different stores also needs to ensure that the transfer medium is secure (if such a requirement is present), and that both stores are properly authenticated to each other prior to the movement.

Security considerations can also arise in the use of references, where a p-assertion contains a unique identifier for data stored elsewhere rather than the actual data itself. In this case, both the asserting and querying actor may require some sort of assurance that any data eventually retrieved by resolving the identifier in the p-assertion is actually the same piece of data that was referred to by the identifier at the time of p-assertion creation by the asserting actor. This can be accomplished by having the querying actor include a digest of the data being referred to along with the identifier of that data in the p-assertion.

The signature on the p-assertion assures that the digest and the identifier will not be changed. Subsequently, if the referenced data is retrieved later, its digest can be computed and compared against the digest within the p-assertion as a check of its integrity. When references are being used, it may also be possible for the provenance store to resolve the references itself. Retrieval of the remotely stored data will be achieved by the remote interactor of the security architecture, with the required credentials being obtained by the trust mediator.

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

5 Security in the Globus Toolkit container

The reference implementation of the provenance Store uses the Globus Toolkit (GT4) as the container deployment of choice; the reader is referred to [HIB06] for relevant background information and associated implementation details. Here, we briefly review the security features of GT4 that are relevant to the security requirements that we intend to address, and which were discussed in the preceding sections.

5.1 Grid Security Infrastructure

The fundamental security component of GT4 is the Grid Security Infrastructure (GSI). It provides the building block for extended security architectures like the Community Authorisation Service (CAS) that will be discussed in the next section. GSI satisfies the following generic security requirements for in a Grid environment:

- Secure and tamper-proof communication between Grid components such as users, resources and programs
- Single sign-on for Grid users across multiple resources
- Privilege delegation from one entity to another for proxy operations
- Interoperability with security operations implemented at participating organisations

GSI includes both resource (server) and client based tools for Grid security. On the resource side, GSI includes tools for managing X509 credentials to identify resources in addition to tools to spawn processes on behalf of authenticated clients. On the client side, GT4 provides tools to create temporary proxy credentials to allow single sign-on and delegation.

A client connecting to a remote resource performs mutual authentication with the resource using the X509 certificates and establishes a secure, encrypted communication established after the initial handshaking between client and resource. Optionally the client can choose to delegate its credentials to the resource to allow further resource access without further intervention.

GSI supports both message and transport level security. Message level security is supported through an implementation of the WS-Security [WSS] and WS-SecureConversation [WSC] standards to provide message protection for SOAP messages. Transport level security is provided through support for the TLS protocol with X.509 certificates. The message level security allows GT4 to be compliant with the WS-Interoperability Basic Security Profile.

The components of the GT4 security stack can be viewed as four functions that can be composed into a SOAP message. These functions are message protection, authentication, delegation and authorization. GT4 implements different standards to provide each of the functions:

1. TLS (transport level) or WS-Security and WS-SecureConversation (message level) are used to protect SOAP messages
2. X.509 End Entity Certificates or Username/Password are used as authentication credentials
3. X.509 Proxy Certificates and WS-Trust are used for delegation
4. SAML assertions and grid-mapfile are used for authorization

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

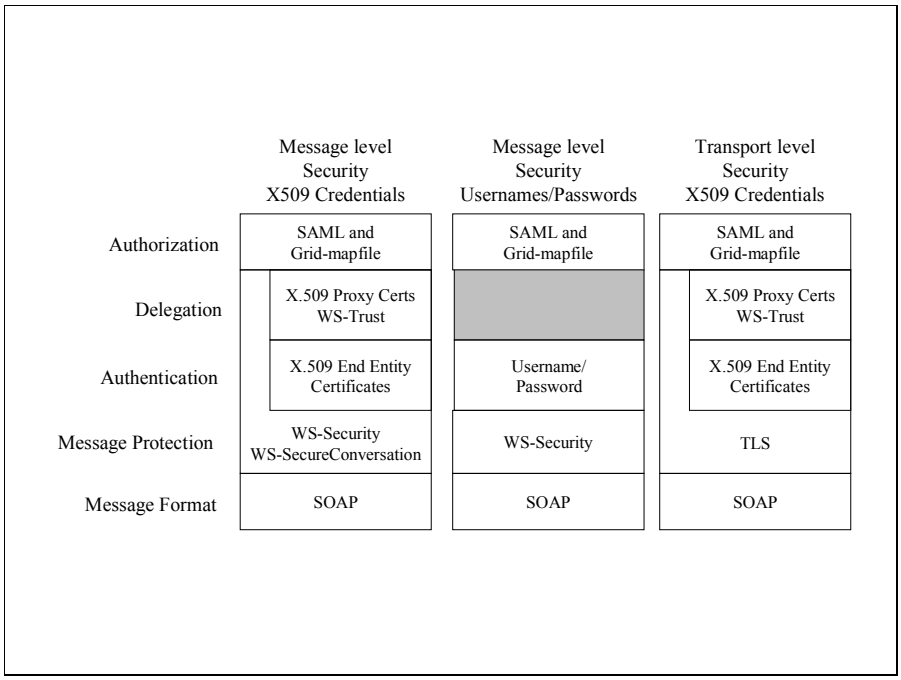

Figure 1: GT4 Security stack

5.1.1 Message level security

The SOAP specification allows for the abstraction of the application specific portion of the payload from any security (digital signature, integrity protection, encryption etc) applied to the payload. This allows the GSI implementation to be applied consistently across SOAP messages for any GT4 Web Service based application or component

As mentioned previously, GSI provides support for both WS-Security and WS-SecureConversation. WS-Security uses mechanisms to provide security on a per-message basis; that is, to an individual message without any pre-existing context between the message sender and receiver (apart from sharing some set of trust tools). WS-SecureConversation on the other hand, allows for an initial exchange of messages to establish a security context which can then be used to protect subsequent messages in a way that requires less computational overhead by trading off the initial overhead in establishing the context for lower overhead for individual messages. WS-SecureConversation is only provided with GSI when using X.509 credentials.

Both WS-Security and WS-SecureConversation are neutral to the types of credentials used to implement the security. GSI allows for both X.509 public key credentials and the combination of username and password for this purpose. GSI when used with either username/password or X.509 credentials uses the WS-Security standard for authentication; that is a message receiver can verify the identity of the message sender. When used with X.509 credentials, GSI uses WS-Security and WS-SecureConversation to provide additional protective mechanisms that can be combined. These are:

- Integrity protection: A message receiver can verify messages were not altered in transit from the message sender
- Encryption: Messages can be protected to provide confidentiality

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

- Replay prevention: A message receiver can check that it has not received multiple copies of a message

The implementations of these protections vary between WS-Security and WS-SecureConversation. In the case of WS-Security, the keys associated with the sender and receiver's X.509 credentials are used. In the case of WS-SecureConversation, the X.509 credentials are used to establish a session key that is used to provide the message protection.

The WS-SecureConversation is a proposed standard from IBM, Microsoft and others which has not yet been formally ratified as a standard but is sufficiently documented to allow implementation, while the WS-Security standard has been ratified by OASIS and defines a framework for applying security to individual SOAP messages.

5.1.2 Authentication

GSI has traditionally supported authentication and delegation through the use of X.509 Certificates and public keys. In GT4, GSI also supports authentication through username and passwords as a deployment option. GSI uses X.509 end entity certificates (EECs) to identify persistent entities such as users and services. X.509 EECs provide each entity with a unique identifier (i.e., a distinguished name) and a method to assert that identifier to another party through the use of an asymmetric key pair bound to the identifier by the certificate. The X.509 EECs used by GSI are conformant to the relevant standards and conventions. Grid deployments around the world have established their own Certification Authorities based on third party software to issue X.509 EECs for use with GSI and the Globus Toolkit.

GSI also supports delegation and single sign-on through the use of standard X.509 Proxy Certificates. Proxy certificates allow bearers of X.509 EECs to delegate their privileges temporarily to another entity. For the purposes of authentication and authorization, GSI treats EECs and Proxy Certificates equivalently. Authentication with X.509 Credentials can be accomplished either via TLS, in the case of transport-level security, or via signature as specified by WS-Security, in the case of message-level security.

5.1.3 Delegation

GT4 supports a delegation service that provides an interface to allow clients to delegate (and renew) X.509 proxy certificates to a service. The interface to this service is based on the WS-Trust specification. When delegation is performed from a proxy certificate, the type of the delegated proxy will always be the same type as the initial proxy.

5.1.4 Username and password authentication

GSI may use WS-Security with textual Usernames and Passwords as described in the WS-Security standard. This mechanism provides a means to support more rudimentary Web Services applications while conforming to WS-I profiles. When using usernames and passwords as opposed to X.509 credentials, GSI only provides authentication and not advanced security features such as delegation, confidentiality, integrity, and replay prevention. However we do note that one can use usernames and passwords with anonymous transport-level security, i.e. unauthenticated TLS as described in Section 4.1, to allow provide privacy of the password.

5.1.5 Authorisation

In addition the grid-mapfile found in earlier versions of the Globus Toolkit, which provides access control based on a list of acceptable user identifiers, GT4 GSI uses the SAML [SAML] standard from

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

OASIS. SAML defines formats for a number of types of security assertions and a protocol for retrieving those assertions. GSI uses SAML AuthorizationDecision assertions in two ways:

1. The Community Authorization Service (CAS) ([CCO03], [PWF02]) issues SAML AuthorizationDecision assertions as its means of communicating the rights of CAS clients to services or resource providers. We will discuss CAS in the next subsection.
2. GSI uses a callout based on the SAML AuthorizationDecision protocol to allow the use of a third party authorization decision service for access control requests to GT4-based services.

GT4 implements a framework for authorization where services can plug in customized decision points on whether an operation can be invoked or not. These decision points are called Policy Decision Points (PDP). The framework is illustrated in Figure 3. In this framework, the GT4 Authorization Engine permits or denies a request access to a service. The decision is based on the combined decisions of a chain of PDPs installed when the GT4 container is initialized. Each PDP (from 1..n in the diagram) returns a PERMIT or DENY decision to the Authorization Engine based on their own analysis of the request message's credentials.

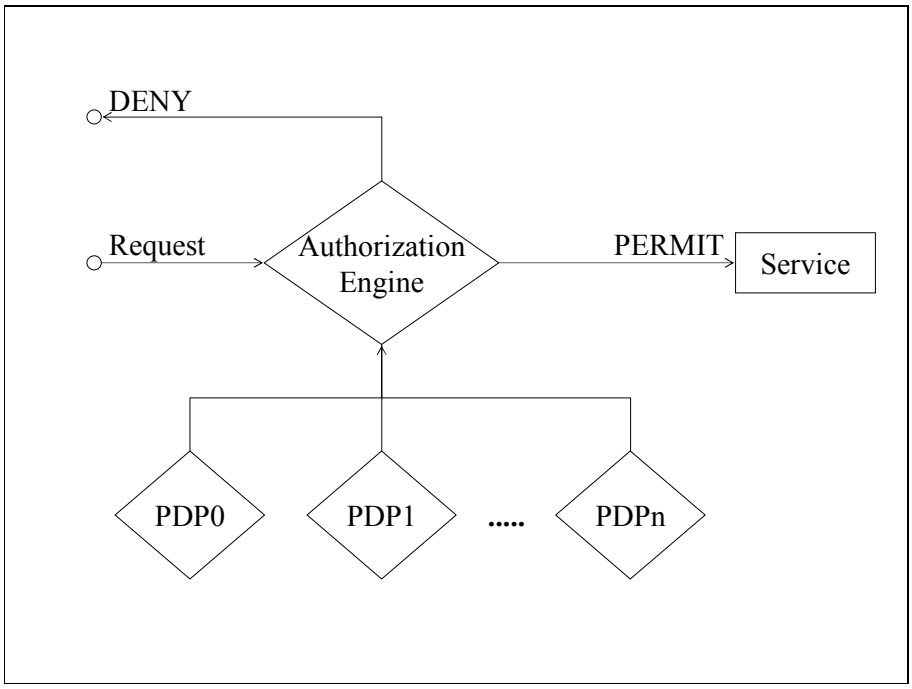

Figure 2: GT4 Authorization Processing

To provide a customised authorization policy, services can implement their own PDP and insert it into the authorization chain. The GT4 framework provides a PDP Java interface that the customised PDP implements.

5.2 Community Authorisation Service (CAS)

CAS ([CCO03], [PWF02]) was originally conceived in the context of providing flexible authorisation policies and enforcement in Virtual Organization (VO) environments which typify many Grid-type

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

application domains. The primary idea is to grant access to blocks of resources to a community (or VO) as a whole, and let the VO itself manage fine-grained access control within that framework.

A VO runs a CAS server to keep track of its membership and fine-grained access control policies. The CAS server contains entries for CAs, users, servers and resources that comprise the community and groups that organize these entities. It also contains policy statements that specify who (which user or group) has the permission, which resource or resource group the permission is granted on, and what permission is granted. CAS also supports the notion of groups, both of users and resources, thus allowing the community to have different roles within the community and to grant members the rights of those roles by assigning them to the appropriate groups.

A user wishing to access community resources contacts the CAS server, which delegates rights to the user based on the request and the user's role within the community. These rights are known as capabilities, which users can present to a resource provider known to the community to gain access to these resources on behalf of the community. When the user presents the capability to a resource, the resource provider grants the user access to the local community resources based on local policy for the community (determined using the resource server's normal local access control mechanisms) and the community policy for the user (determined by examining the policy statements carried in the capability). In other words, the resource server will permit a request authenticated with a capability if the resource server's local policy authorizes the request for the grantor of the capability, and the capability itself authorizes the request for the bearer. The user thus effectively gets the intersection of the set of rights granted to the community by the resource provider and the set of rights defined by the capability granted to the user by the community.

Once a resource provider has installed the appropriate resource server software, that provider can grant access to specific resources to specific communities by using local access-control mechanisms to grant access to those resources to the subject names associated with those communities' CAS servers. Prior to granting that access, the resource providers may use offline methods to verify that the CAS server is run by someone who actually represents the community, and that the community's policies are compatible with those of the resource provider.

In order to support CAS, the basic GSI infrastructure supporting delegation via proxy certificates has been extended to support proxy certificates that carry additional policy information restricting its use (a restricted proxy). CAS servers use these restricted proxies to delegate to each user only those rights granted to that user under the community's policy. The policy information within the proxy certificate is signed by the CAS server, and may optionally include the user's identity as well. This separation of CAS policy assertion from user proxy credentials allows resources accepting these credentials to identify the user involved in a request as if they were using normal GSI credentials. This strategy in turn allows a resource provider to use normal mechanisms for auditing and to apply an additional level of policy enforcement based on the user's identity.

A sample sequence of a typical CAS use case scenario is illustrated in Figure 4 and detailed below.

1. The user authenticates, using personal proxy credentials, to the CAS server serving the user's VO. The CAS server established the user's identity and rights in the VO using a local policy database.

2. The CAS server issues the user a signed policy assertion containing the user's identity and rights in the VO. The CAS client generates a new proxy certificate for the user and embeds the CAS policy assertion in the proxy certificate as a non-critical X.509 extension. This embedding of the assertion in the proxy is not necessary from a security perspective but instead allows any application that can use a normal GSI proxy to be able to use the proxy

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

with CAS assertion embedded in it, as it looks like a standard proxy certificate to existing APIs and protocols.

3. The user uses the proxy certificate with the embedded CAS assertion to authenticate to a resource. The resource authenticates the user using normal GSI authentication, allowing it to determine the user's identity and apply local policy as desired. It then parses the policy assertion in the proxy certificate and, based on the assertion, allows the user to access the resource using the VO account but constrains the user's activities using the rights in the assertion.

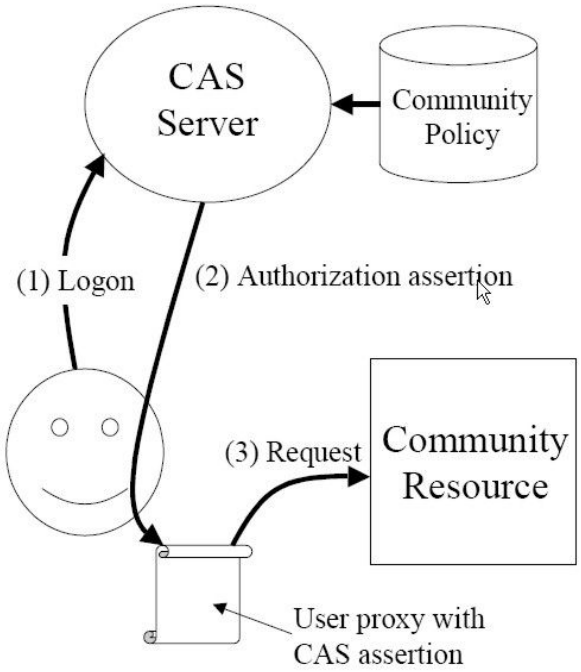

Figure 4: CAS Use Case Scenario

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

6 Implementing provenance security architecture with GT4

In this section, we discuss how the security functionality of GT4 can be used in the realization of an implementation of the security architecture that we had detailed earlier in Sections 3 and 4. The following mapping between GT4 security features and our architectural components or functionality is identified:

1. Secure communication and mutual authentication between actors and provenance stores are achieved using either the WS-SecureConversation or WS-Security facilities of GSI. X509 EEC certificates and GSI proxy certificates are used for authentication purposes. It is not intended to support username/passwords as authentication credentials, but they may be used as an initial authentication mechanism in order to obtain the relevant certificates from a remote keystore. This can be incorporated as part of the client side library.

2. Delegation of identity / access control as a result of multiple interactions between various components in the logical architecture (as discussed in Section 4.3) is accommodated directly using the proxy certificates of GSI.

3. The identity validator is incorporated as part of GSI, and this extracts a relevant X500 identity (possibly delegated, or as part of an EEC X509 certificate) that can then be mapped to a desired, customised Internal Representation (IR) of the security architecture. A direct approach would be to simply allow the IR to be the extracted identity information without any further mapping, although we reserve the possibility for additional mapping through a customised specification of the Internal Representation List.

4. The authorization policy is equivalent to the GT4 grid map file for the default GT4 authorization scheme. As discussed earlier in Section 5.1.5, it is possible to implement a customised authorisation engine as a PDP module in the GT4 authorisation chain (Figure 3). This can be configured to read from a local policy file specified with a specific schema that fits the semantics and granularity of authorisation expressions required in the two target application domains. In the first instance, we intend to commence with an XML file which lists X500 DNs to permitted operations as specified in the provenance store WSDL, and refine the expressions from there. Alternatively, we can resort, if required, to standards such as XACML [XACML] to express more complicated authorisation assertions in a standard format.

5. The trust mediator maps to the GT4 component that performs SAML authorisation call outs to a third party authorisation service. This third party service is functionally equivalent to the credential server in our security architecture. This component is also useful in creating SAML assertions as part of the CAS infrastructure that we use for federated security that we will discuss shortly.

6. The derivation engine will incorporate some of the filtering ability associated with the querying functionality of the provenance store [GJMM05] in order to ensure that p-assertions returned conform to the authorisation assertions specified in the authorisation policies. This filtering functionality is independent of GSI, but would typically be implemented as another PDP module in the GT4 authorisation chain.

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

7. Authorisations based on RBAC model (which are particularly relevant in the organ transplant application domain) are accommodated using the role and group structure policy assertions already provided as part of the CAS infrastructure.

8. Mapping between the native authorisation assertions of the authorisation policy of host environment (for example, the WebDAV server in the aerospace engineering application domain) to the authorisation assertions of the authorisation policy of the provenance store. This effectively requires a translation between the schemas, semantics and granularity of both policies and will be accomplished by a statically defined approach for such translations. This approach in the first instance will be a document outlining guidelines for such a translation, which may be automated to some extent at a later stage. We note that there are no specific requirements to support the automation of such a translation process, and the semantics and granularity of the authorization policy for the provenance store will be designed to only support provenance requirements.

In Section 4.4, we discuss several security issues pertaining to distributed provenance stores. The need to federate identity in this instance can be accommodated through an appropriate structuring of the CAS infrastructure. Each provenance store within a separate security domain is considered as a unique resource provider, with potential provenance users being organized into membership of various VOs. These VOs could, for example, correspond to hospital or regional authorities (for the case of the organ transplant management application domain) or engineering departments (for the case of aerospace engineering application domain). As we have already discussed, access control policies to provenance stores are now specified in context of VOs, while the identities and roles of the various users are specified in the policies of the VO at a central CAS server.

Registration of new users from different VOs or security domains is handled by the CAS server independently of the different provenance stores, while registration of new VOs can be handled by communication between the CAS server and the various provenance stores that are registered with that CAS server. Such a communication can be accomplished using the SAML assertions exchanged between the CAS server and the trust mediator of the provenance store.

In Section 4.4, the first approach for resolving distributed provenance queries involve provenance stores interacting with other provenance stores in different security domains. In order for this to happen, the remote interactor of a given provenance store would need to obtain the appropriate CAS proxy credential to present to the identity validator of the remote provenance store. Here, the given provenance store could elect to obtain its own CAS proxy certificate from the CAS server that the remote provenance store is registered with. Alternatively, the provenance store could generate a new proxy certificate from the CAS proxy certificate presented by the actor submitting the original distributed query.

Last but not least, signatures on p-assertions, verifying asserter identity and the generation of digests to validate references in p-assertions (all mentioned in Section 2) can be accomplished through some pre and post processing of p-assertions using the XML Signature [XMLSign] and Encryption [XMLEncrypt] standards (for which several open source implementations exist), since information in the p-assertion is structured in an XML format. All of this processing can be achieved independently of the underlying GSI security framework, and may alternatively reuse some of the functionality already present in the GSI security libraries towards this end.

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

7 Conclusion

This document essentially presents an approach towards implementing the security architecture first conceived and described in [GJMM06] and [IHT05] in the context of the GT4 security framework. The first 3 sections (Section 2, 3 and 4) of this document iterates on material presented in [GJMM06] and [IHT05] to provide background context for the more implementation-specific discussion that follows in Section 5 and 6. The preliminary discussion in these first 3 sections is also refined to focus on the specific security issues that the project intends to address; we thus omit other optional, albeit interesting, issues discussed in [GJMM06] and [IHT05]. In Section 5, we provide a brief description of the aspects of the GT4 security framework which are relevant to our implementation approach. Finally, the actual details of how the features of this framework are mapped to the original security architecture for implementation purposes is elaborated in Section 6.

We note that the first release of the reference implementation (D9.3.2) contains an implementation of the authentication and authorisation components described earlier. The next deliverable in this workpackage (D4.3.1) will provide implementation-specific details of the approach outlined in Section 6 of this document, along with associated schemas for authorisation and access control policies, where relevant.

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

Appendix A References

[CG05] Chen L et al. An Architecture for provenance Systems. Internal document for Provenance project.

[MGBM05] Simon Miles, Paul Groth, Miguel Branco, and Luc Moreau. The requirements of recording and using provenance in e-science experiments. Technical report, University of Southampton, 2005.

[IHT05] John Ibbotson, Neil Hardman, and Victor Tan. D4.1.1: Security requirements. Technical report, IBM Hursley, September 2005.

[GJMM05] Paul Groth, Sheng Jiang, Simon Miles, Steve Munroe, Victor Tan, Sofia Tsasakou, Luc Moreau. D.3.2.1: An Architecture for Provenance Systems. Technical report. University of Southampton, February 2006

[And05b] Árpa´d Andics (ed.). D2.2.1: Software requirements document. Technical report, MTA SZTAKI, February 2005.

[CCO03] Shane Canon, Steve Chan, Doug Olson, Craig Tull, and Von Welch. Using CAS to manage role-based VO sub-groups. In Proceedings of Computing in High Energy Physics 03 (CHEP '03), 2003.

[PWF02] L. Pearlman, V. Welch, I. Foster, C. Kesselman, S. Tuecke. A Community Authorization Service for Group Collaboration. Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002.

[WFK04] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, F. Siebenlist. X.509 Proxy Certificates for Dynamic Delegation. 3rd Annual PKI R&D Workshop, 2004.

[HIB06] Neil Hardman, John Ibbotson, Alexis Biller. Provenance Implementation Design. Technical Report. IBM 2006

[WSS] WS-Security specification.
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

[WSC] WS-SecureConversation specification.
http://specs.xmlsoap.org/ws/2005/02/sc/WS-SecureConversation.pdf

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme

[SAML] SAML specification.
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[XACML] XACML specification.
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

[XMLEncrypt] XML Encryption specification.
http://www.w3.org/Encryption/2001/

[XMLSign] XML Signature specification.
http://www.w3.org/TR/xmldsig-core/

The *PROVENANCE* Project receives research funding from the European Community's Sixth Framework Programme