Title: Security Requirements

Editor: John Ibbotson

Authors: John Ibbotson (IBM)

Neil Hardman (IBM)

Victor Tan (UoS)

Reviewers: All project partners

Identifier: D4.1.1

Type: Deliverable

Version: 1.0

Date: 20th September 2005

Status: Public

Summary

The purpose of this document is to define an architecture for security in a Provenance aware system. Security functionality is closely coupled to the actual Provenance architecture, which in the first instance is represented by the project's logical architecture definition. This document introduces the logical security architecture and describes its major components and their interactions.

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 s.V. 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 Introduction

1.1 Purpose of the Document

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 in have stringent requirements on access to data manipulated within the system. Correspondingly, p-assertions that incorporate or are derived from these data are likely to have similar security restrictions on them as well. Although security is a non-functional requirement, software engineering methodology strongly recommends that security considerations be integrated into the development life-cycle as early as possible. With this as a motivating factor, we proceed in this document to outline a security architecture for the logical architecture [CG05] of the provenance project.

1.2 Overview of the Document

In Section 2, we briefly define some of the common security concepts that we use in this document. In Section 3, we survey the security issues relevant to the conception of provenance. Following that, we present the security architecture for the provenance store and describe the functionality and interaction between its constituent components in Section 4. In Section 5, we discuss the security issues pertaining to other components in the logical architecture. We then outline the security issues that remain unaddressed in Section 6, and conclude in Section 7.

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

2 Background

This section provides a brief narrative that encompasses some of the more common terminologies encountered in the field of electronic security. It is not intended to be a comprehensive treatise of the area, and merely seeks to provide a conceptual background for the security discussion in the remaining sections of this document.

We consider a system that offers some functionality through a set of resources that can be accessed and manipulated. It is usually the case that these resources can only be accessible or manipulated in specific ways in order to ensure that the functionality offered by the entire system is unaffected. The *integrity* of a resource is a property of that resource that is preserved as long as the resource is accessed or manipulated in the prescribed manner. It is assumed that these restrictions on resource manipulation necessary to preserve its integrity are known to the entity responsible for the system resources, which we shall term as the *system administrator*. Hence for the trivial case where a system administrator accesses or manipulates a system resource, there is no risk of intentional resource integrity violation. The role of a system administrator would be roughly analogous to that of a managing actor within the context of the provenance architecture.

However, systems are generally useful only where their functionality (as provided by their internal resources) is accessible to external entities. In situations such as this, the system administrator may not have direct control over these external entities and cannot ensure that their behaviour is compliant with preservation of resource integrity. There is therefore the need to perform *access control* to these resources, and this is typically achieved by restricting access (out of the overall group of entities that are capable of accessing the system resources) to a specific group of entities that are trusted by the system administrator. We do not consider in our discussion the context of trust and how it is established in the first instance between the system administrator and a group of external entities.

A preliminary and necessary requirement for access control is *authentication*, which is the process of producing an *identity*} based on some *credentials* submitted by the entity to the security infrastructure. An identity produced from a successful authentication process can subsequently be used in access control to ascertain whether an entity's accompanying request to access some resource in a specific manner is permitted or not. An entity that is allowed to access a given resource in a specific manner is said to be *authorised* to perform that access on the specific resource, and such an authorisation can be expressed in different ways. For example, in a mandatory *access control* system, entities are authorised to access resources on the basis of the relationship between different security labels or clearance levels assigned to the various resources and entities. In a discretionary *access control system*, authorisations are generally expressed in the form of a direct relationship between a given identity and the resources accessible to it.

The set of authorisations in a system is typically predetermined by the system administrator according to some existing *security policies*, and the scope of enforcement of this policy is generally known as a *security domain*. It should be noted that the identity produced from an authentication process is only meaningful to the system performing the authentication; it is entirely possible that a single entity may be represented in different systems with different internal system identities. Authentication and access control are often tightly interlinked components in a security architecture.

Situations may arise when a data resource (or a copy of it) has to be transported across an open medium, such as a network connection, where it is no longer protected by the security infrastructure of the system. *Privacy* is a property of this data that is achieved in this context by transforming the data

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

into a form (typically via the use of *symmetric cryptographic mechanisms*) that is unintelligible to entities that were not originally authorised to access it. Integrity of this data is achieved in this context by ensuring that any processing or modification of the data while in transit becomes detectable. This generally involves the use of *asymmetric cryptographic* mechanisms such as *digital signatures*.

Signatures are generated by using the private portion of a *public/private key pair* to generate a message digest on a piece of data. Only the owner of the private key is capable of generating a digest on that data that can subsequently be verified successfully by any entity possessing the public portion of the key pair. This ensures that any modification of the data by any entity other than the owner would be detectable via an unsuccessful verification attempt. In addition, the uniqueness of the private key enables the establishment of a direct link between the key owner and a piece of data signed with that key. This is sometimes useful in attempting to guarantee the property of *non-repudiation*, which seeks to ensure that an individual is held accountable for an action in the system and cannot deny having undertaken this action post hoc. If such an action is expressible in the form of a data item, then a signature on this item undisputedly establishes corresponding responsibility for the action on the key owner.

Certificates are electronic documents used to link a public key with an identity of an entity possessing the corresponding private key. The reliability of this link is established by a signature on the certificate by a party trusted by all entities that use the certificates. This trusted party is usually a *certificate authority (CA)*, who is the primary entity responsible for the life cycle management of these certificates within a Public *Key Infrastructure (PKI)*.

Interactions across different security domains can sometimes occur, particularly in large scale, distributed systems exemplified by the Grid or the Web Services environment. For example, a workflow initiated by an individual may interact with resources from several systems, each with separately administered access control schemes. Here, the individual would need to authenticate to the relevant security components of each of these systems, since the individual would very likely have distinct internal identities in the different security domains. *Federation of identity* is a method which seeks to simplify the security procedure, and hence the overall workflow process, by requiring the individual to authenticate only once (usually known as *single sign-on*) in order to access resources across several security domains. In order to accomplish this while still retaining the original level of security, the infrastructure of each of these security domains needs to be structured to communicate relevant information, particularly pertaining to actor identities, between themselves.

Another requirement that arises within a distributed environment is the need for delegation of *access control rights*. For example, during the process of workflow execution by an enactment engine, a service invoked by the workflow engine might need to invoke another service in order to fulfil the requested functionality. If these services exist in different security domains, then the individual responsible for initiating the workflow would need to authenticate twice: once to each of them. Once again, a single sign-on capability can be provided if a mechanism is implemented in the security infrastructure that empowers the first service to invoke the second service based on the access control rights transferred to it from the individual concerned. Note here that while the conception of single sign-on is the same as is the case in identity federation, the motivating situations are slightly different. Delegation of access control generally also carries the implication that the delegated access rights are only qualified within a certain context: for example, during the duration of a workflow or to access specific resources only. There must be a way to ensure that a service that has been delegated some rights from an individual does not maintain the ability to use these rights indefinitely outside of the given context, nor to delegate it further onwards to other entities unless permitted to do so.

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

It needs to be borne in mind that delegation of access control and federation of identity are not novel security methodologies nor do they enhance the security capabilities of a system. They merely provide a way to maintain the existing level of security in individual security domains while attempting to simplify the security requirements that arise when complex interactions between these different domains occur.

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

3 Provenance Related Security Issues

In this section, we outline the security issues that we believe are relevant pertaining to our notion of provenance. We note however that not all of these issues are relevant in the context of the provenance software requirements (see [CG05] chapter 9), and the eventual security architecture will only address those that are.

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. *Authenticating recording actors that submit p-assertions*. Recording actors submit interaction and actor p-assertions for storage; it is essential that these actors are correctly authenticated and their identities associated with these p-assertions in the manner mandated by the application domain requirements. The association process can be subsumed under the protocol that provides secure remote delivery of the p-assertion to the store, or can be part of some pre-processing prior to final storage in a back-end database. The establishment of responsibility or liability may be critical in certain legally motivated provenance uses; guaranteeing non-repudiation would then become a necessary addendum to identity association.
3. *Derivation of authorisation information relating to p-assertions*. It is likely that p-assertions will contain or be derived in some fashion from an existing piece of data in the system. For example, an application actor with access to a database may send a message containing an item from that database to another actor. This item is likely to have certain access control restrictions enforced upon it within the security domain of the database in question. When a p-assertion is created for the transmitted message and recorded to the provenance store, appropriate access control restrictions (or authorisations) must now be established for this new entry to ensure that any future access to it is in accordance with the security policies of the provenance store.

   In some situations, it may be useful to relate the authorisation for the newly recorded p-assertion in some way to the access control restrictions on the original database item that the p-assertion is based upon. This effectively allows for a more flexible specification of authorisations on p-assertions by taking into account information other than that found in statically predefined security policies on the provenance store. A possible approach towards this end is for the recording actor to submit additional information along with the p-assertion to be stored. This additional information can then be utilised in an automated manner by the provenance store to generate appropriate authorisations for the new p-assertion.

   Once p-assertions have been recorded in a provenance store, provenance representation can be used by processing services and presentation user interfaces. The former provide added-value to the query interfaces by further searching, analysing and reasoning over recorded p-assertions, whereas the latter essentially visualise query results and processing services outputs. Figure 3.1 of [CG05] provides examples of such processing services and presentation UIs offering functionality discussed in [MGBM05]. For instance, processing services can offer auditing facilities, can analyse quality of service based on previous execution, can compare the processes used to produce several data items, can verify that a given execution was semantically valid, can identify points in the execution where results are no longer up-to-date in order to resume execution from these points, can re-construct a workflow from an execution trace, or can generate a textual description of an execution. Presentation user interfaces can for

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

instance offer browsing facilities over provenance stores, visualise differences in different execution, illustrate execution from a more semantic viewpoint, can visualise the performance of execution, and can be used to construct provenance-based workflows. We note that such a list of processing services and presentation UIs is illustrative and not exhaustive; furthermore, it does not represent a commitment by

4. *Context-based authorisation specifications.* Processing services provide added-value to the provenance query interfaces by further searching, analysing and reasoning over recorded p-assertions. Some of the operations that can be performed by a processing service have a well defined functionality; for example, comparing processes used to produce several data items. In order to perform this operation, a certain set of p-assertions identified by certain criteria will need to be retrieved from the provenance store. Another operation, for example, verifying that a given execution was semantically valid, will require the retrieval of another set of different p-assertions. Situations may arise where it is useful to ensure that certain actors are authorised to access only the relevant p-assertion subset necessary for a specific operation (or more generally, any type of context in which provenance representations can be used in). This would require an ability to express authorisations at this level, as well as some way to translate these context-based authorisations into finer grained authorisations at the p-assertion level.

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

4 Provenance Store Security Architecture

In this section, we present the logical design for a security architecture for the provenance store. 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.

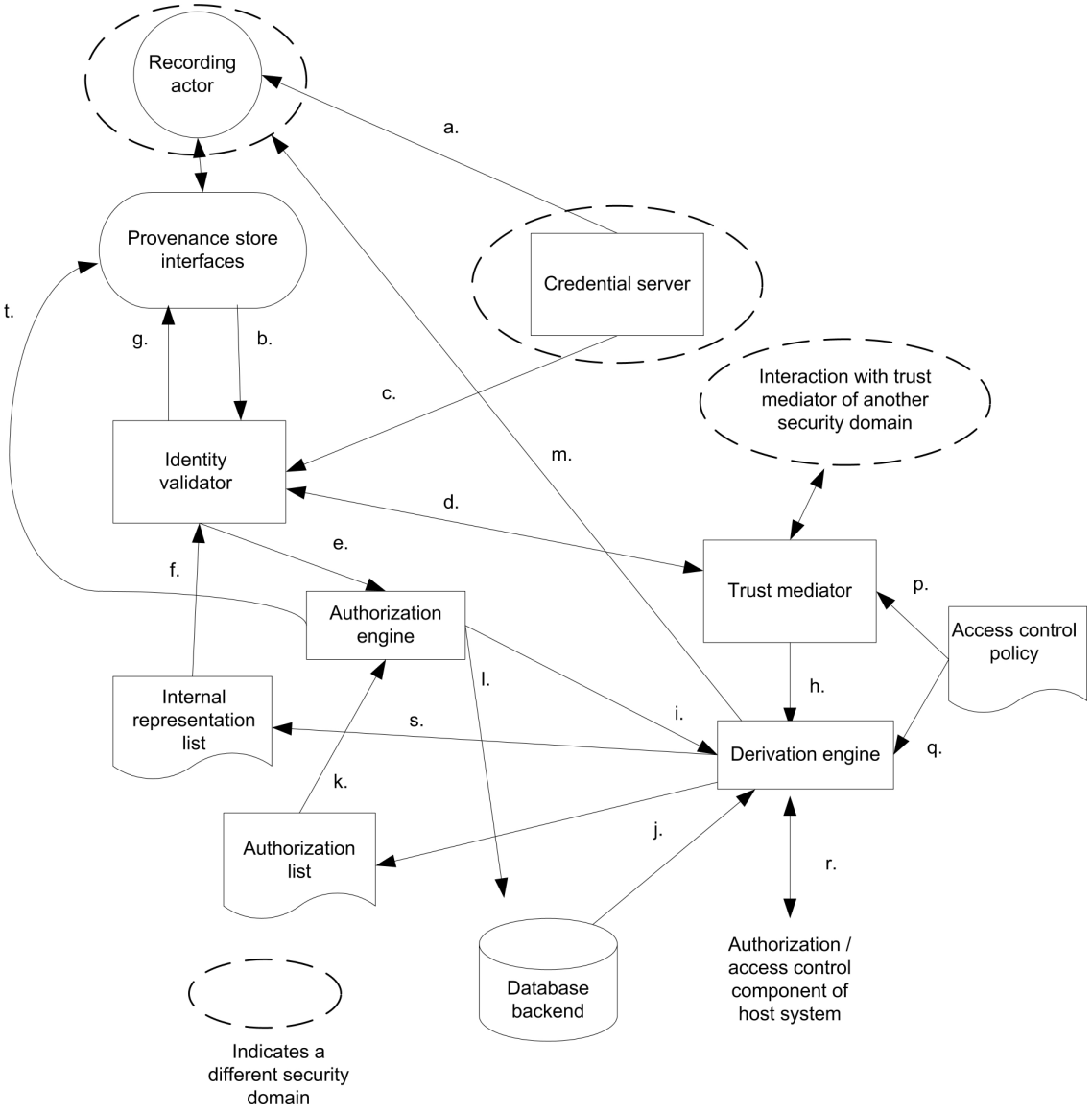

Figure 1 Provenance Store Security Architecture

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

4.1 Components of the Security Architecture

The provenance store in the logical architecture exposes three different interfaces (recording, management, and query) for different purposes. All of these interfaces can have optional operations that access the required security functionality in the actor side libraries. 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 recording actor mutually authenticate each other during this secure transmission. This is important from the viewpoint of the sender in order to circumvent potential impersonations of a valid provenance store by malicious parties that would then gain access to the provenance data.

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 imply 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 a Role Based Access Control (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 appropriate identity is registered on the submitted passertions in the event of a store request operation. This correlates with the second security issue in Section 4.2. The fact that the recording actor submitting the p-assertion may not necessarily be the asserting actor that created that p-assertion (i.e. the submitting entity is a delegatee for the actual originator) needs to be taken into account.
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. If granted, the requested operation is performed and the appropriate acknowledgment 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

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

validator. In addition, it is possible to employ either one or both of these two approaches to specifying authorisation:

1. An identity / role is assumed to have no authorisations in the initial case, and explicit authorisations have to be granted;
2. An identity / role is assumed to have complete authorisation in the initial case, and explicit restrictions have to be placed.

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. Both the access control policy and authorisation policy could be subsumed under the broad umbrella of provenance store policy in the logical architecture. The *database backend* provides actual physical storage for the p-assertions. The *trust mediator* is an optional component of the architecture that is required only if federated identity management is to be supported. It provides the interface to other security domains, and is the component through which security assertions are exchanged about local internal representations and authorisations. The *derivation engine* provides the following functionality:

1. Derives new authorisations from existing authorisation information. For the case of the third security issue for provenance as identified in Section 4.2, the authorisation information would originate with the p-assertion as part of submitted request from the actor. Alternatively, there may be a need to correlate the authorisations as specified in the access control policy with the authorisations in the host systems security architecture, in the event that a tighter integration between both architectures is required.
2. Creates a set of appropriate authorisations corresponding to a higher level context-based authorisation specification. This corresponds to the fourth security issue for provenance, and can be considered to be optional functionality.

4.2 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 labeled 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 client-side library functionality. The entire message could be signed, or some of its contents could be signed (for example, signing the p-assertion for non-repudiation. The required credentials can be obtained from the credential server prior to the invocation process. (a.).
2. The identity validator intercepts the message and verifies the signature if there is one; 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 immediately resolved but there is additional information to indicate the domain in which it might be recognised, an appropriate request is sent off to the trust mediator (d.).
3. The mediator interacts with its counterpart in the relevant domain using appropriate assertions as part of a security protocol, and then formulates a new security assertion based on the access control policy (p.). This assertion is then sent off to the derivation engine (h.). The recognition of the new IR is achieved by appropriate additions to the internal representation list (s.) and the authorisation list (j.) by the derivation engine, based on the access control policy (q.)

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

4. If the credentials fail to resolve successfully either in the current domain or the domain that the mediator attempts to interact with, 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.
5. 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 4.2, 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.
6. The derivation engine subsequently creates new authorisation information from the formulated statement based on the rules prescribed in the access control policy (q). For purposes of maximising performance, this new authorisation information could have been created by the recording actor doing the submission so that the derivation engine uses it directly without any further processing. The new authorisation for the p-assertion to be stored is added to the authorisation list (j). The p-assertion is now sent onwards to the storage pre-processor, along with relevant authorisation information or metadata that it is meant to be stored with (l.). Here, the p-assertion may be encrypted or signed using the private key of the provenance store domain (in the case that the database backend is in a different domain) and then formatted to correspond to the recording interface schema of the backend.
7. The p-assertion is then sent over a secure, mutually authenticated link to the database backend in an analogous manner to the way that the original submission by the recording actor to the provenance store was accomplished (o). The database backend, if hosted by a third party provider, could be exposed for remote access in a variety of ways with corresponding authentication and access control mechanisms; the storage pre-processor will need to be aware of these possibilities and cater to them accordingly by acquiring and generating the relevant security credentials. The acknowledgement sent back from the database backend (n.) is processed accordingly and sent back to the recording actor (m.)
8. A session connection can be established on a secure link so that future submissions no longer require the submission of authentication credentials that need to be verified. This requires that the implementation support such a secure persistent session connection.

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 for the derivation engine 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 authorization information associated with the request as well as any additional authorization information stored and associated 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.

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 list which

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

may include deletion, modification and addition of entries. All of these operations are consequent on the identity validator first recognizing 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 list / internal representation list can then be propagated through the derivation engine (r).

Scenario 5: Multiple provenance stores
An implementation of the provenance architecture may require distributed provenance stores for reasons such as scalability. In such an instance, p-assertions related to a specific workflow or sequence of execution may be stored in multiple provenance stores by the responsible recording actors. Consequently, a query to retrieve a group of related p-assertions may potentially require a series of queries to the various provenance stores holding the desired p-assertions. In addition, it is unlikely that the querying actor making the query will have prior knowledge of the additional provenance stores that it might need to query in order to satisfy its original query.

We assume that in such a scenario, the p-assertions themselves will be recorded with links [CG05] that indicate the provenance stores in which other associated p-assertions are contained within. On retrieval of the relevant p-assertions from an initial provenance store, a querying actor is then able to navigate the trail of links to query the relevant provenance stores. Retrieval of the required p-assertions from each one of these distributed provenance stores will follow the sequence outlined in Scenario 1. If the querying actor is not recognized (i.e. does not exist as a valid identity) in all the provenance stores it makes a query to, the trust mediator of the provenance stores that do not recognize it will need to communicate with trust mediators of other provenance stores that do. Thus, the credentials submitted by the querying actor must provide enough information for such interaction between trust mediators of different provenance stores.

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

5 Security in other Architectural Components

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

5.1 Between other Components and the Provenance Store

The other components in the logical architecture that interact 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 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 client 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. Again this can be provided as additional libraries or as an interface to external applications.
4. 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.

5.2 Intermediate Components

By intermediate components, we refer to components that are not directly accessible by the user. 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:

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. The derivation engine will not be used to create new authorization information as the submitted data item is not intended to be stored. However it may be used in performing some security-related functionality on the data item, for example encrypting or filtering out a certain

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

portion of it. This will be accomplished in conjunction with security policy dictating the operation of this intermediate component.

3. Once the request is approved by the authorization 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.

5.3 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 contains the user and the presentation UI, another containing the processing service, and the third encapsulating the provenance store.

When the presentation UI under the user's control sends a request to the presentation service, an appropriate credential is submitted by the user for purposes of authentication. If the request is authorized, 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:

1. Authenticate to the provenance store using the credentials of the presentation service, whereupon subsequent authorization 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 authorization 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).
2. 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 recognize and process this delegated credential accordingly, and infer the identity or associated role of the original user. Subsequent authorization decisions are then on the basis of the user's 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 authorizations 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 authorization is equivalently reduced as there is now only a need to check on these presentation services.

There are some drawbacks however with this approach however. Firstly, authorization lists are likely to be duplicated between many presentation services, as it is unlikely that authorization for a specific user will differ between different services. Accordingly, changes or additions to these authorization lists must then also be propagated between the different copies on all services. Lastly, application services storing p-assertions through the recording interface must now provide authorization information pertaining to presentation services rather than specific users. This may necessitate additional overhead in communication between application services and presentation services.

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

The second approach therefore appears to be a more feasible one. There will however be an overhead associated with communication between the presentation UI and the presentation service in order to create an appropriate delegation credential. Depending on 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 authorization 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 authorization list in the provenance store to reflect the delegation of authorizations 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 authorization 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 authorization 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 authorization engines of both the presentation service and provenance store via their respective trust mediators, if complex authorization decisions are to be affected.

The description in this section 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) can support multiple acts of delegation.

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

6 Additional Security Issues

While this document discusses security considerations for all components of the logical architecture, the primary focus is on the security architecture for the provenance store as we have established it as the core component in the logical architecture. The construction and implementation of this architecture will therefore take precedence over security considerations for other components. In particular, if the provenance system is to be integrated into an independent application domain of which the developers of the provenance system have no control over, then it is assumed that some, if not all, of the security issues relating to the application services have already been addressed adequately. Such issues include the need for delegation of access control, which was already discussed at the end of the previous section. In this section, we describe a few more of these types of security issues, which also do not completely come under the purview of the security work to be achieved for the provenance architecture.

1. Digital signatures on p-assertions within interaction provenance between two services. The requirement of non-repudiation may mandate that signatures be applied to specific p-assertions within a given interaction provenance trace as proof that a service undertook an action within a specific context. The application of these signatures will have to be handled by the two participating services in that interaction and remains outside the purview of the provenance store security architecture.
2. Mutual authentication and secure transport of p-assertions between two services. Both activities have to be handled or negotiated between the two services involved in the production of interaction provenance.
3. Anonymisation of data. The organ transplant management application requires the exchange of patient-related information during the interaction of services. Legal restrictions mandate that data of this nature is anonymized (patient identity is removed) and depersonalised (i.e. the identity of the patient cannot be traced based on other information in the record). Again, this requirement remains outside the context of the security architecture.
4. Establishing the access control polices for a RBAC system. Authorisation in this system is very much policy-driven; specifying the nature of these policies within the particular context of RBAC is vital to the correct and efficient functioning of the architecture. Policy related issues will likely to be addressed in a separate document.
5. Delegation of access control. As noted before, invocation of a chain of services that may occur in a workflow could mandate for delegation of access rights. The mechanisms to achieve this remains part of the workflow implementation and is not addressed directly by the security architecture. The architecture however can provide support in two ways:
   - The identity validator can be enhanced with functionality to process specific credentials that are representative of the delegation act.
   - Modification of the authorisation list via the trust mediator to reflect some delegation of access control rights within a given security domain. This could be particularly useful when a more centralised approach to delegating access is adopted.
6. Long term storage of provenance information. If a third party database provider is used, then provenance information may need to be encrypted or signed by the storage pre-processor prior to sending it off for storage. In the event that this provenance is intended to be stored for a relatively long period (e.g. 100 years), a situation likely to arise is one where the original cryptographic keys and / or algorithms become outdated or expire. Such issues must be catered for in some way, for example, by having a key archival facility and re-signing / re-encrypting provenance information periodically over the intended storage duration.
7. Expiry of certificates. For workflows that run over a relatively long period, it is possible that certificates could expire in the middle of a workflow run. If an actor uses a certificate as part

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

of the authentication process to the provenance store, then expiry of this certificate would mean that submission invocations that were once accepted within the context of this workflow have now become invalid. To avoid situations like this, proper management of certificates and keys at the actor end is called for (i.e. workflow duration is estimated against certificate life time prior to commencing a workflow). Alternatively, the provenance store security policy could be articulated appropriately to avoid this situation. For example, the authorisation component could keep track of all invocations from a given actor within the context of a specific activity and allow remaining invocations to proceed in that activity as long as the initial invocations were signed with a valid certificate.

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

7 Conclusion and Future Work

In this document, we discussed security issues that were relevant in the context of provenance. The security architecture for the provenance store is then presented along with an explanation of the functionality of its constituent components. This is followed by an illustration of the interaction of these various components for some standard interactions with the provenance store. We then discuss security issues pertaining to other components of the architecture. Finally, we outline some of the security issues that we do not address or are out of scope of the proposed security architecture.

Deliverable D4.2.1 of the provenance project will refine further the material presented in this document. In this later draft, we intend to describe a concrete instantiation of the security architecture at a level of detail suitable for an implementation of the architecture. This will include standards, interfaces and technologies that are relevant in the development of a physical implementation of the logical architecture.

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.

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