SEVENTH FRAMEWORK PROGRAMME
Challenge 1
Information and Communication Technologies

Trusted Architecture for Securely Shared Services

Document type: Deliverable

Title:	Specification of secure data repositories and authoritative sources

Work Package:	4
Deliverable Number:	4.2
Editor:	David Chadwick, University of Kent
Dissemination Level:	Public
Preparation Date:	31 December 2010
Version:	1.3

Information Society Technologies

The TAS³ Consortium

Beneficiary Nr	Beneficiary name	Beneficiary short name	Date enter project	Date exit project
1	Katholieke Universiteit Leuven (Proj. Coord)	KUL	M1	M48
2	Synergetics nv/sa	SYN	M1	M48
3	University of Kent	KENT	M1	M48
4	University of Karlsruhe	KARL	M1	M48
5	Technical University Eindhoven	TU/E	M1	M48
6	Consiglio Nazionale delle Ricerche	CNR	M1	M48
7	University of Koblenz-Landau	UNIKOLD	M1	M48
8	Vrije Universiteit Brussel	VUB	M1	M48
9	University of Zaragoza	UNIZAR	M1	M30
10	University of Nottingham	NOT	M1	M48
11	SAP research (S&T Coord.)	SAP	M1	M48
12	Eifel asbl	EIF	M1	M30
13	Intalio Ltd	INT	M1	M18
14	Risaris Ltd	RIS	M1	M48
15	Kenteq	KETQ	M1	M48
16	Oracle	ORACLE	M1	M48
17	Custodix nv/sa	CUS	M1	M48
18	Medisoft bv	MEDI	M1	M12
19	KIT	KARL	M1	M48
20	Symlabs SA	SYM	M18	M30

Contributors

	Name	**Organisation**
1	David Chadwick	Kent
2	Michele Bezzi	SAP
3	Andreas Pashalidis	K.U.Leuven
	Reviewers	
4	Danny De Cock	K.U.Leuven
5	Sampo Kellomäki	Risaris
6	Marc Santos	UNIKOLD
7	Antonia Bertolino	CNR

Table of Contents

Table of Figures

LIST OF ACRONYMS

AIPEP – Application Independent Policy Enforcement Point
EPAL - Enterprise Privacy Authorization Language
PRIME DHP - Prime Data Handling Policies
ID – Identity or Identifier depending on context
IdP – Identity Provider
OAI-ORE - Open Archives Initiative - Object Reuse and Exchange
P3P – Platform for Privacy Preferences
PDP – Policy Decision Point
PEP – Policy Enforcement Point
PII – Personal Identifying Information
PoO – Proof of Ownership
PRIME - Privacy and Identity Management for Europe, an EC Integrated Project
RBAC – Role Based Access Controls
RDF - Resource Description Framework
SAML – Security Assertions Mark-up Language
SSO – Single Sign On
SP – Service Provider
SSL/TLS – Secure Sockets Layer/Transport Layer Security
TPM - Trusted Platform Module
URL – Universal Resource Locator

1 Executive Summary

This document describes the security mechanisms that are needed to control access to data repositories. Data repositories (or repositories for short) are web based resources that hold data in a secure and privacy preserving manner. They provide web services interfaces that allow the data to be input, updated, retrieved and deleted.

Repositories will hold data of varying sensitivity levels. They may store publicly readable data, or they may store very sensitive data such as medical records. The data may be accessed by the data owner, the data subject, or by third parties. In particular this document describes how data subjects can delegate their access rights to third parties, regardless of whether those third parties are already known to the data repository or not.

The TAS^3 project is primarily concerned with data that is personal, so called personal identifying information (PII). PII can generically be thought of as the attributes of a person, such as their age, name, address, qualifications, medical details etc. This deliverable introduces the concept of an authoritative source (or authentic source – the terms are synonymous) which is the trusted source of a particular data item (or personal attribute). It is preferable that relying parties who wish to access the PII assets of subjects, obtain them from their authoritative sources rather than from "any old" data repository. In this context, it is preferable that personal information from a set of different authoritative sources can be merged together to form a composite data object, without actually copying the data from its authoritative sources to a third party repository. This document describes how such composite objects can be built and secured by using "proof of ownership" references to the authoritative sources. The use of references honours the principle of data collection minimization. It also alleviates the integrity problems associated with data duplication and the existence of multiple distributed copies (replicas and/or caches).

Appendix 1 of this document presents a set of use cases for using secure repositories and indicates the steps involved when a user: first registers with a repository, creates a PII asset in a repository (including the sticky policy to go with it), delegates access rights to a third party, and deletes her account and PII asset from the repository. It also shows the step involved when a third party accesses the repository to either read or append information to the PII asset, and follows the proof of ownership references to retrieved embedded PII assets.

Finally this deliverable shows how secure repositories can be mapped onto the TAS^3 architecture and the recommended service oriented protocols.

This is a minor update of the first version of the description of TAS^3 secure repositories. The reader should be aware that this document represents the output from the first three years of the project. Implementation experience gained in the third year of the project will further ensure that this version will be improved in its final version.

2 Introduction

Data repositories are web enabled containers for storing digital information. In TAS^3 we are particularly concerned that these repositories can be trusted, and that they should hold data in a secure and privacy preserving manner. These repositories should provide web services interfaces that allow the data to be input, updated, retrieved and deleted by authorised users only. They should allow new users to subscribe to them to create accounts, and existing users to unsubscribe and delete their accounts and all their stored data. These repositories will hold data of varying sensitivity levels. They may store publicly readable data, or they may store very sensitive data such as medical records or military information. The data may be accessed by the data owner, the data subject, or by authorised third parties. In order to ensure that the data remains secure, even after it has been retrieved from the repository by a third party, we introduce the concept of sticky policies. A sticky policy is a set of rules which dictate how the data should be handled in order to maintain its security and privacy properties. The sticky policy is attached to the data when it is created in the repository, and when the data is retrieved, the sticky policy accompanies the data. The policy is cryptographically bound (or stuck) to the data when the data leaves the repository so as to ensure both the policy's and the data's integrity. The sticky policy might in fact be comprised of a set of access control and privacy policies, but it can be regarded as a single sticky policy at a high level of abstraction. Likewise the data might be compromised of multiple parts, but at a high level of abstraction it can be regarded as a single data object.

In TAS^3 we are particularly concerned about repository information which is **personal identifying information** (PII). Whilst PII is not our sole concern, we believe that if we provide secure and trusted repositories for PII these repositories can easily be used for other types of information such as federation meta-data or security policies, whilst the converse is not necessarily true. From a TAS^3 repository's perspective, a TAS^3 application is a distributed application that needs to access PII (or other data) that is distributed around the globe and stored in multiple repositories. A TAS^3 application is trusted to respect the security and privacy of this distributed PII. It will only disseminate this PII according to the policies that are stuck to the PII. A TAS^3 application may need to merge parts of the distributed PII into a seamless whole for a (trusted) third party to view and possibly update or append, before passing this onto other third parties for them to do likewise. During this process the sticky policies of the parts will be honoured and the merged document might contain a merged sticky policy. The third parties who retrieve the repository data may be web services or the front end clients of human beings. Web services must be TAS^3 enabled and trusted before they will be given PII by TAS^3 applications, whereas the front end clients of humans (at least in the short term) may not be (see below for more discussions of this).

The TAS^3 infrastructure is application independent. It makes no difference to the TAS^3 network whether the distributed application is an electronic health care application, which needs to access a patient's electronic health care records which are distributed around the globe and held in different hospital, GP surgery and dental practice repositories at home and abroad, and need to be viewed and updated by medical practitioners at different locations; or is an e-employability application that will assemble a user's e-portfolio from a variety of distributed digital assets such as references, qualifications, certificates etc. that are stored in various repositories and need to be collected and viewed by potential employers as directed by an employability consultant. Consequently TAS^3 repositories are also application and data independent.

The generic technical aspects of such distributed repository information, which may be needed by many TAS^3 distributed applications, are as follows:

- a user has multiple sets of electronic data or PII. Each set is of varying size and complexity, and is held in different digital formats (e.g. pictures, text, videos etc);

- one data set may refer to another data set. In general the sets may be linked in an arbitrary mesh;

- a user's complete set of data is distributed around the globe in multiple repositories, and no-one person or organisation fully knows where all the information is. The user is the only person who is most likely to know where most (if not all) of her data is stored, but even with the best will in the world, a user may forget where some of it is stored, or she may wish to "forget" that some of it exists (such as a criminal record or a bad reference from a past employer);

- subsets of the data are likely to be identified in different repositories with different unique but unrelated identifiers. Even in countries which have national ID card systems, there is no guarantee that all repositories holding a user's data will use the same national ID to identify it.

- all PII is governed by data protection legislation and therefore has to be handled appropriately by the recipients and holders of the PII.

- different subsets of the data have different sensitivity levels. Some of it may be publicly accessible information (e.g. postal address, telephone number), some of it the user may be willing to share with a wide range of people (e.g. degree classification or other awards), other of it will be highly confidential with the user only wishing to share it with close associates (e.g. a medical record), whilst yet other of it may have strict legal restrictions on who may view the PII and under what conditions (e.g. criminal record);

- most of the data will currently be held in legacy (non-web services enabled) computer databases (or legacy repositories) and therefore will need gateway machines to make them accessible to TAS^3 applications;

- most of the initial likely recipient's of a user's personal data wont be members of a TAS^3 network but they ought to be catered for in order to organically grow the system.

- multiple, possibly conflicting, copies of at least some of a user's PII will exist in different repositories so there needs to be a way of determining which are the authoritative copies.

Given the above complexity of today's world, we would like to engineer a TAS^3 technical security infrastructure that exhibits the following properties at least:

- Each element of PII has an authoritative (or authentic) source which is responsible for its veracity. An authoritative source is "a data repository, which is managed by one or more entities that are functionally responsible for the collection, validation and updating of the data originating from the actual source of the information" [12] e.g. a governmental entity, educational institution, or the data subject himself, and which is recognized and trusted as being the most qualitative source of such information. For example, a university may be trusted as being the authoritative source of degree certificates for its alumni, whilst the user herself may be authoritative for her music preferences.

- Each element of PII has a set of security related policies that control who should access it, for what purposes, what credentials they need to access it etc.

- These security related policies should be "sticky", in the sense that they stay with the PII and control access to it as the PII travels around the TAS^3 network.

  Note that the sticky policy can only be guaranteed and enforced whilst the data remains in the TAS^3 network. If data covered by a sticky policy is exported from a TAS^3 network, the TAS^3 connector that bridges between the TAS^3 world and the non-TAS^3 world is responsible for complying with these policies as far as is possible, i.e. if a sticky policy specifies that the data should not be exported outside the TAS^3 network, the TAS^3 connector should not do so. But once the data and its policy has left the TAS^3 network, the policy's "stickiness" can no longer be guaranteed.

- The set of security related policies of an element of PII may be made up of different sub-polices, such as an access control policy, a privacy policy, a user consent policy, a trust policy, a delegation policy and an authentication policy etc.

- Different actors may be responsible for setting different policies from the PII's set of security policies e.g. the user (data subject) may set some components of the policy, government (i.e. the law) may set some other components, and the repository owner may set yet other components of it. Consequently the set of security policies might be in conflict, so there will need to be a conflict resolution policy to control such situations.

- When repositories collect PII, the PII must be collected for specified, explicit and legitimate purposes. The purpose of the processing should be defined at the latest at the moment of the collection of the PII. This is called the Finality Principle in [12]. The purpose of processing should be held in the sticky policy of the PII to ensure compliance.

- There should be a mechanism that allows a user to link together different elements of their PII, held in different repositories, into a composite object, so that copies do not need to be taken (which leads to loss of integrity). This honours the principle of data collection minimization. Whilst each PII element should be collected just once, it is potentially used many times by many different recipients, but each access must be controlled by the PII's sticky policy.

- When composite objects are created they too should have their own sticky policies that control access to them.

- Data collection minimisation raises the question of how the finality principle can be ensured long term, i.e., if PII is intended to be re-used in a great deal of instances, how can we ensure compatibility with the purposes for which the PII was originally collected? We can enforce this by recording the purpose in the PII's sticky policy, but what if a given purpose was not foreseen (or did not exist) at the moment of collection? In this case the sticky policy must be updated to reflect the new purpose(s). Thus repositories must allow sticky policies to be dynamically updated. But if the user is not connected to the repository at the time the data request is made and the policy update is needed, this introduces the need for a "user call-back" mechanism in order to obtain the additional consent.

- The PII elements may be held in different repositories under different unique identifiers. Consequently, any recipient of a composite PII object needs to be assured that all the linked elements do indeed belong to the user who claims ownership of them; otherwise a dishonest user might link together the PII of different people and claim that it all belongs to him. We might call this the 'cheating data subject' scenario, in which the cheating user includes references to other people's data when creating a compound object. TAS^3 repositories need a "proof of ownership" mechanism to protect against this.

- Each element of PII needs corresponding meta-data that describes it i.e. its content, format, security policies, authoritative source etc.

- A TAS^3 repository needs to have a new user registration facility in order to integrate new users who are not currently part of the TAS^{3}network.

- A TAS^3 repository needs to cater for PII recipients who are unknown the TAS^3 network and who do not wish to register as members, e.g., by having a data export capability.

- A TAS^3 repository must be generic and be usable by many different applications.

- A TAS^3 repository should utilise suitable pre-existing components wherever suitable e.g. web browsers, SSL/TLS to secure the communications between the web browser and the repository, SSO using SAML assertions etc. We should not re-invent the wheel unless absolutely necessary.

- The technical infrastructure must be easy to use for human users. Therefore it should use hotlinks to connect the various elements of PII together when the user wishes to do this.

- In most scenarios human users should not be expected to have to install new software in order to use TAS^3 repositories. This implies that TAS^3 repositories should support browser based clients. However, there may be some high security applications where this is not secure enough, and which require user workstations to be trusted i.e. only use trusted components that will honour the sticky policies attached to the data. We would like all user workstations to migrate to this trusted configuration in due course, but recognise that for practical reasons this will not be possible in the short term.

Note that the above list does not cover general features that most (insecure) repositories might be expected to already have, such as version control, data integrity, good performance, backup and restore mechanisms etc.

3 Composite Repository Objects

Digital resources are currently distributed around the web in various repositories. When a composite object such as an e-portfolio or patient health record is created from existing distributed repository objects, it is common practice today to "import" or copy the individual data sets into the new composite object. It would be preferable if the distributed resources could simply be referenced from the composite object, rather than copied into them, and then a fresh copy downloaded at the time of use, as is the case with documents on the web today. This will minimise the data collection, facilitate data integrity, and ensure that authoritative sources are used for each data set (or PII asset). Composite objects should therefore hold references to embedded objects rather than actually embedding them. Similarly the referred to objects may be composite objects themselves. This provides the essential feature of recursion.

However, this model poses a significant number of challenges from a security and usability perspective:

- Firstly we need to ensure that a user can easily create a composite PII object from his existing PII assets (which may be atomic or composite).
- Secondly we need to ensure that granting access to the composite object continues to honour the security policies of the existing PII assets and does not bypass them. Conversely, the recipient of a composite PII object who has appropriate access rights should be able to access all the distributed component PII assets as if it were a single object.
- Thirdly the recipient may need to be assured (i.e. have confidence and trust) that all the distributed PII assets do really belong to the same user who claims ownership of the root composite object, even if the user has different login identifiers or identities at the different repositories. We term this *proof of ownership*, and it is designed to counter the *cheating data subject* scenario.

Concerning usability, references or URLs to the distributed PII assets should be capable of being embedded within new composite objects, and the composite objects should then be able to be stored in repositories in the same way as any other PII asset. As with any digital document, the composite object may be edited to grow or shrink in size, and may include other digital documents within it, either directly or by reference. It should be the task of the user interface to support the user in providing different secure, managed views of the composite object for different purposes and audiences. Appendix 1 provides several use case scenarios which indicate what the usage experiences of using TAS^3 secured repositories and composite objects could be like.

Concerning the security aspects, recipients of references to composite PII objects should be given access rights to each of the components parts as appropriate. Some component parts may be inaccessible to some of the recipients, and other component parts may have different access rights to different recipients. Furthermore a recipient may wish to have confidence that each of the component parts does really belong to the user who is claiming ownership of

the root composite document referenced by the root URL. This is important, in order to stop users wrongly claiming ownership of attributes that belong to others. When users are known by different identifiers at different repositories, it is a challenging task to ensure that the recipient is able to successfully link all these together, whilst simultaneously protecting the privacy of the PII subject. The security aspects of this are discussed in more detail in the next section.

Repository documents should be viewable within web browsers. In principal, the PII assets can be any kind of resource. In practice, they will be of a type which can be rendered by a web browser[1].

3.1 Structure of Composite Objects

Should composite repository objects be simply a set of references/URLs and associated text, or should there be some higher level meta-information that describes the structure of a composite document? Current practice tends to suggest that this is needed. However there is currently no de-facto or international standard for this meta-information, and several proprietary domain specific solutions exist. Whether or which meta-information to use has not been resolved yet within the TAS[3] project, but in this context is it worth noting the Open Archives Initiative Object Reuse and Exchange (OAI-ORE) [6] project, which is developing standards for the description and exchange of compound digital objects[2]. These compound objects will typically be composed of (possibly distributed) resources of different digital media types, as in the TAS[3] project. The specifications developed under ORE will allow repositories to exchange meta-information about their constituent digital objects, and provide services that facilitate access and ingest of these representations. In other words, ORE supports the actual transfer of composite digital objects between repositories, and allows composite objects to be printed as one complete document. The ORE model describes the structure and semantics of aggregations of web resources through a Resource Map, which enumerates the constituents of the aggregation and the relationships among those constituents by using Resource Description Framework (RDF) triples of subject, predicate, object. For example, a composite object Book comprising two Chapters might be described by the RDF triples: Book aggregates Chapter 1, Book aggregates Chapter 2, Chapter 2 follows Chapter 1.

Whilst there are obvious parallels between TAS[3] repository documents and OAI-ORE compound objects, OAI-ORE currently says nothing about the security or privacy of its compound documents or constituent parts. Thus if we choose to use a model such OAI-ORE it will need to be supplemented with appropriate security controls before it can be applied to TAS[3] documents.

4 Level of security required for repositories

Recipients of PII have a legal duty of care, as proscribed in the UK and European Data Protection Acts [2, 3], to ensure that PII is only accessed for its intended purposes by those that need to access it. Often today this is implemented by the recipient publishing its data protection policy, and users trusting them to enforce this by internal means. However such a system often breaks down, see for example the highly publicised case of the UK Government losing the bank details of 25 million citizens [4]. Consequently there is an increasing need for computers to keep control of PII according to security policies formulated by humans. Such a

[1] In cases where this is not so, modern web browsers will display a suitable error message, and often prompt the user to download a suitable plug-in in order to view the resource.

[2] A so called compound digital object is simply an aggregation of discrete digital objects. A typical example is a thesis or dissertation which may be comprised of a number of text documents, images, etc.

policy controlled security mechanism should be generic and applicable to a large number of distributed applications. It can then be tailored to the specific requirements of each TAS^3 enabled application. Such a policy controlled system is not available today, but it is the intention of the TAS^3 project to provide such sticky policy controlled repositories.

Most personal information in digital repositories will not be considered as sensitive or as life critical as the PII contained in say electronic health records or military systems. Nevertheless some elements may be, such as a criminal record or compromising photographs. For example, an e-portfolio may contain elements of an electronic health record when it is required for certain job applications. Thus the TAS^3 repository security model should be capable of handling highly sensitive information even if the majority of the PII and use cases won't use it. The security model should therefore support a broad range of security mechanisms starting from very little security e.g. for publicly accessible information, up to very strong security for access to highly sensitive and restricted PII. When a recipient wishes to validate some received PII, the model should support a full range of possibilities, ranging from no technical validation possible (i.e. relying on personal trust in the sender) up to strong technical proof of the authenticity of each individual PII component and that they all belong to the same user.

Collecting all this distributed PII together into a composite object, and then delegating third parties the right to access it, can be a technically challenging task. In the simplest case it does not have to be. For example, the user may collect all her PII data together by copying it into a new compound "thick" document, with no technical proofs that she has the right to claim any of the information embedded in it. She may then send the entire composite document to a recipient with no security proofs that it actually originated from her (e.g. via a plain email message) with an implied policy that it should not be given to anyone else. In this case the recipient has to trust that the sender is who she claims to be, and that all the claims in the composite document are true. Similarly the user has to trust that the recipient is who she thinks he is, and that he will not distribute her PII further. The TAS^3 infrastructure should not forbid users from adopting this simple scenario if it is deemed to be "good enough" for their use cases and does not breach any legal obligations. The legal component of the PII's sticky policy will ensure that the latter does not occur.

On the other hand the TAS^3 infrastructure should be able to support a much more technically complex and secure scenario as well, thereby allowing the user community to choose the model which best suits their needs. In the most secure and privacy preserving scenario, the user creates a skeleton or "thin" composite document and stores it in a secure repository. The skeleton contains embedded "proof of ownership" references to the various authoritative sources of pre-existing PII assets. Each proof of ownership reference contains (directly or indirectly) information allowing the authoritative source to prove to the recipient that the subject of both the referred to and skeleton objects is the same physical person. In addition, each PII asset, as well as the skeleton itself, has its own "sticky" security and privacy policy which controls access to it. When the user creates a proof of ownership reference to the skeleton, this binds her identity to the skeleton object. She may then securely transfer this to the recipient, and the recipient's system will acknowledges safe receipt of it. The user may have revealed her true identity to the recipient or may be a pseudonym, at her discretion. Before the recipient's system is allowed to access the skeleton it must confirm to the repository that it is willing and able to abide by the sticky security and privacy policy of the composite object, likewise for each embedded PII asset. In the most secure case and in the longer term, this will be provided by a Trusted Platform Module (TPM) module on the recipient's PC confirming that it has the necessary machinery to enforce the sticky policy. In the short term, before the TPM facility and supporting software are ubiquitously available, this will be provided by the recipient confirming conformance to a legal statement displayed in his browser, possibly with a machine-readable version of the privacy statement being downloaded and automatically compared with the user's privacy preferences, thereby facilitating the recipient in making an informed choice about his commitments.

Once the recipient has gained access to the skeleton document, the embedded proof of ownership references (PoO) in the skeleton will direct the recipient to the embedded PII assets, and again these PoO references will contain information allowing the source to provide proof to the recipient that the referring and referred to objects belong to the same user. The recipient will have to be granted access by the policies of the embedded PII assets before gaining access to them and being provided with proof that the assets all belong to the same user as the skeleton. In this way the model is recursive. The recipient contacts the various authoritative sources of the embedded PII assets, and if he is authorised, the sources will return digitally signed copies of the PII assets and their sticky policies. The sticky policies will control the recipient's further use of the PII assets. The recipient will be provided with proof that all the component parts belong to the same user. In this way the recipient can technically verify each authoritative source who asserts each PII asset on behalf of the user, and that all parts truly belong to the same user. The recipient's trust is now transferred from the user to the external trusted third parties (the authoritative sources) who assert the various PII assets. Likewise the user's trust is transferred from the recipient to the trusted repositories and authoritative sources who will only return the user's PII to recipients which they know can be trusted to enforce the user's sticky policies.

In between these two extremes, we expect there to be various configurable levels of security, which have varying levels of privacy and trust associated with them, via varying technical mechanisms. It will be for users of the system to determine which levels of security, privacy and trust are most appropriate for their particular use cases.

5 Repository Security in Detail

Security is discussed under the following headings: authentication, authorisation (which includes privacy protection) and proof of ownership.

- Peer Entity Authentication is *the corroboration that a peer entity in an association is the one claimed* [13]. It is the process whereby one party proves their identity to the other party with which they are communicating. Data origin authentication *is the corroboration that the source of data received is as claimed* [13]. It can be used for example to prove that a PII asset has been received from its authentic source.
- Authorisation is the granting of rights, which includes the granting of access based on access rights [13]. It is the process whereby one party grants another party permission to access a resource. Privacy protection adds extra dimensions to authorisation such as: purpose of access, retention periods and agreement to conform to the subject's authorisation and privacy policies. Section 6 describes privacy policies in more detail.
- Proof of ownership allows a recipient to prove that an aggregated set of distributed PII assets all belong to the same user, even if the true identity of the user is hidden from the recipient.

Usually a resource owner is the party that grants another party access to the resource. In the case of PII, it is often unclear who the owner of a particular PII resource is. Is it the subject of the PII or the authoritative source of the PII? Since both have some legitimate claims about being the owner, both may sometimes wish to set the access permissions and privacy policies on a particular PII asset. The TAS^3 model allows multiple users to set different policies on the same resource (see Deliverable 7.1 [7] for more details).

The act of delegation is considered under both authentication and authorisation. Delegation (of authentication) is the process whereby one party grants to another party the right to perform authentication of third parties on its behalf. Delegation (of authority) is the process whereby one party (the delegator) grants to another party (the delegate) a permission enjoyed by the delegator. Delegation can be fixed (static) or recursive (dynamic). In static

delegation we have the delegator and their delegates. In dynamic delegation a delegate may subsequently act as a delegator, creating a delegation chain.

A requestor needs to be both authenticated and authorised before he can be granted access to the PII asset held by an authoritative source. The requestor also needs to explicitly or implicitly (e.g. by using computer machinery that automatically enforces policies) agree to abiding by the privacy policy that is "stuck" to the PII asset before being given the PII asset. However, the authoritative source does not need to perform all of these steps itself. It can delegate authentication to a trusted third party, or delegate authorisation to a trusted third party, or delegate both authentication and authorisation to the same or different trusted third parties.

5.1 Authentication

Authentication can be one way or mutual. With one-way authentication, only one of the communicating parties is authenticated by the other, but not vice versa. Typically this occurs when a user enters his username and password to enter a web site (without authenticating that the web site is genuine). With mutual authentication, both parties authenticate the other. Whether one-way or mutual authentication, or indeed no authentication, is used by repositories will be a configurable choice. However the TAS[3] repositories should be able to support all three modes.

Authentication can be direct between the two communicating parties, when they have a direct trust relationship, or can be delegated to a mutually trusted third party, in which case they have an indirect trust relationship. An example of a direct trust relationship for one-way authentication is when a repository is trusted by the user to store the latter's username and password, and these are used by the repository to authenticate the user. An example of delegated mutual authentication is when the two communicating parties both have X.509 public key certificates issued by the same Certification Authority which is trusted by both parties to authenticate the other party.

In a TAS[3] application some authentication may be one way and direct, e.g. between a user who authenticates to an authoritative source of the user's PII asset, such as a university who holds the user's degree classification. Other authentication may be mutual and delegated to one or more trusted third parties. In a TAS[3] federated environment user authentication is typically delegated to trusted third parties known as Identity Providers (IdPs). When the authoritative source of a user's PII asset, the user and the recipient of the PII asset, are all part of the same trust network that has adopted TAS[3] technology (hereafter called a TAS[3] trust network) it is relatively easy for them to authenticate to each other. In this scenario the repository service provider is indirectly authenticated to the client (i.e. the PII subject or PII recipient) via its SSL/TLS certificate provided by a Certification Authority trusted by the TAS[3] trust network. The client is indirectly authenticated to the repository service provider via a mutually trusted Identity Provider (IdP) who vouches for the authenticity of the client (without necessarily revealing the identity of the client). The IdP and client can utilise whatever authentication mechanism they chose, mutual or one way, direct or indirect, for example, username and password, Kerberos token, one time password etc. subject of course to any legal obligations to use a certain minimum level of security[3]. The IdP and repository service provider perform mutual indirect authentication using X.509 public key certificates issued by TAS[3] trusted CAs. The repository service provider is not involved in the authentication of the client, but rather trusts the IdP to correctly perform authentication on its behalf. The repository service provider simply has to be assured that the service consumer corresponds to the client that has been authenticated by the IdP.

[3] It should be possible to include a TAS[3] trust network's minimum Level of Assurance (LOA) in the trust network's meta-data in order to enforce any legal obligations that the network might have regarding authentication.

However, it is unrealistic to assume that every organisation and every user will ever be members of the same TAS^3 trust network or trust network. Some users and organisations may not be a member of any TAS^3 trust network. Whilst we should cater for this possibility by allowing and even actively encouraging new registrations, even in the middle of a PII exchange dialogue, we should also be able to cater for non-federated entities sharing information that is stored in TAS^3 enabled repositories. For example, when a user who is a member of a TAS^3 trust network wishes to share his CV with a potential employer who is not currently part of the same TAS^3 trust network, this should be allowed, as should facilitating the employer in registering as a new user. In this scenario, when the repository and PII recipient do not have an existing direct or indirect trust relationship, it is the well known problem of how do two strangers authenticate each other. In short, they cannot. This can only be achieved via a dynamically activated mutually trusted third party (which establishes an indirect trust relationship between the strangers). In the case of a TAS^3 infrastructure, when a PII recipient wishes to retrieve a user's PII asset from an authoritative source, but has no existing direct or indirect trust relationship with the source, and therefore the two parties are unable to mutually authenticate each other, they could both delegate authentication of the other to the user himself, who can then dynamically act as the mutually trusted third party. This means that the user will need to authenticate the PII recipient before telling the authoritative source who it is, which of course is in the user's best interests to do anyway, otherwise he risks giving his PII to an unknown entity. Likewise the user will need to authenticate the authoritative source before telling the PII recipient who it is, but the user does this anyway when he accesses it. However, the PII recipient is less likely to trust the user to authenticate the authoritative source on its behalf, as this would allow the user to pick a fraudulent source of information to vouch for his/her attributes e.g. the user tells the PII recipient that he has a first class honours degree from Oxbridge (when he has no degree at all) and points the PII recipient to www.Oxbridgedegrees.com (a (fictitious) site that sells degree certificates on demand to all comers). Unfortunately the PII recipient has no choice but to trust the user to tell it who the authoritative sources are, unless it has an alternative method of validating the authoritative sources, such as a trusted directory service operated by the TAS^3 trust network. In many cases this should be possible. Even when it is not, it can often be done by using out of band means such as a telephone call, letter, word of mouth or simply common knowledge. This type of user initiated mutual authentication is most easily achieved by the user passing:

- the identifier of the PII recipient to the repository, which the repository can use to identify the PII recipient,
- a secret between the PII recipient and the repository which the PII recipient can use to authenticate to the repository at connection time (it does not matter which of the three parties generates the secret), and
- the identity of the repository to the PII recipient which the PII recipient can validate via its SSL certificate passed to his browser at connection time.

5.2 Authorisation

Authorisation can be enabled in a variety of ways. If the repository/authoritative source and PII recipient are able to mutually authenticate, then the repository administrator could include the PII recipient in the repository's access control list or local policy for controlling access to the user's PII asset. But managing the access rights in this way is not a feasible approach since:

a) the repository administrator is unlikely to know all the potential recipients of the subject's PII asset
b) even if he did, the management overheads would be too great for the repository administrator to deal with.

Consequently the repository administrator will need to delegate authorisation rights to the user (PII subject) herself, thereby giving the user full control over who can access her PII.

Delegation of access rights to the PII subject (user) may be granted by the repository in at least the following five ways:

i) allowing the user to directly (or indirectly) manage the repository's local policy or access control list, by inserting (and removing) the names of the PII recipients into the list or policy;
ii) allowing the user to issue an authorisation token to the PII recipient, which the repository can subsequently validate when the recipient requests access to the PII asset;
iii) issuing its own authorisation token to the PII recipient when requested to do so by the user;
iv) allowing the user to create her own "sticky" policy that is stored in the repository along with the user's PII asset;
v) utilising a trusted delegation service to issue delegation tokens when requested (directly or indirectly) by the user

Method i) is not likely to be popular with repository administrators, since they are unlikely to want users to meddle with their access control lists or local policies.

Method iii) may prove to be popular with repository administrators, since it is very easy for software developers to implement. Repositories only need to validate authorisation tokens that they have previously issued themselves, therefore their content does not need to be standardised. It gives implementers complete flexibility over the contents of the tokens. It is similar to the current practice of placing cookies in web browsers and then subsequently retrieving them. Method iii) is in fact the OAuth model for delegation [8]. However this method *on its own* does not provide the user with any control over her PII once the PII recipient has received a copy of it, and since the tokens are not standardised, if the user moves her data from one repository to another, her access control rules are lost. However, method iii) is a useful method and it will be discussed further under Secret URLs.

Method ii), allowing users to issue their own authorisation tokens, will require some standardisation effort since the created tokens will need to be understood by multiple systems. Furthermore, the tokens will have to be digitally signed to prove their authenticity, so this will require users to have their own asymmetric key pairs. To date, it has proven difficult to encourage users to manage their own key pairs and public key certificates, and unless a new usability paradigm is found for asymmetric keys, this is likely to continue to be the case. However, if we provide users with a trusted token issuing service that can issue authorisation tokens on behalf of users, then this mechanism does become viable. It is method v). The Delegation Issuing Service described in Deliverable 7.1 is one such system [7].

Method iv), allowing users to create their own sticky policies, is the primary method chosen for TAS^3 repositories when participants can authenticate each other. The advantage of this method is that it provides users with control over their PII, even after it has left the repository. It also allows users to move their data and policies between repositories, providing of course that the policy is either standardised or understood by both systems. Note that some repositories may not give the user complete control over the contents of their sticky policies, and may provide the user with a template containing a restricted range of options to choose from, but conceptually this is still the same mechanism, even if its scope is somewhat limited.

With method v), the trusted delegation service can issue tokens on behalf of either the user or the repository. The delegation token will be issued when the user requests it, either directly or indirectly, and will ultimately be consumed by the repository. The repository may utilise a delegation service without the user being aware of this (indirect request), or the user may utilise the delegation service directly (direct request). The advantages of a delegation service are: the delegation tokens are standardised as they are used by multiple parties, the user does not need a key pair in order to issue a delegation token, the repository service can be simplified by offloading aspects of user and resource management to the delegation

service. The Community Authorization Service (CAS) [15] from the Grid world is one example of the latter.

If the repository and PII recipient are not able to mutually authenticate each other, then we cannot use methods i) and iv) above. Method ii) could be used, but method iii) is easier to implement for reasons outlined above. In this scenario the repository and PII recipient will need to rely on the user to act as the trusted third party during the process of issuing the authorisation token. The user will need to authenticate to the repository and request an authorisation token be issued for the (anonymous) PII recipient. At least three solutions are possible. One method is for the repository to embed a secret in the authorisation token that only it knows; another method is for the repository to securely embed the public key of the (unknown) PII recipient in the authorisation token; a third method is for the PII recipient to give the user a secret/unique identifier which the user can pass to the repository for it to validate when the PII recipient arrives.

Embedding a secret in the authorisation token and giving this token to the user for him to pass to the PII recipient of choice, allows the user to act as the trusted intermediary, and is simple in concept, use and implementation. Possession of the secret token by any third party indicates to the repository that this third party is trusted by the user to be the PII recipient. The PII recipient is authorized (by the user) but not authenticated, i.e. the PII recipient is never identified to the repository and remains essentially anonymous as far as the repository is concerned. Note that this is a relatively weak trust model and it is open to abuse by the PII recipient, who may copy and share the secret token with others. There is no complete audit trail for the token. Authorisation tokens containing secrets are a type of *bearer* token, meaning that whoever possesses or bears the token is authorised to use it. Unfortunately one may have to revert to bearer tokens when the repository has no independent mechanism of authenticating the PII recipient. The latter simply presents the authorisation bearer token at access time.

One way of allowing the repository to authenticate an unknown PII recipient is by trusting the user to authenticate the PII recipient's public key. If the PII recipient has a public key (or public key certificate) which the user has authenticated, then the user can give this to the repository, which the latter can store or embed in the authorisation token it issues. In this case the repository will subsequently be able to both authenticate and authorise the PII recipient when he calls to access the PII, by getting him to sign a challenge, but the PII recipient essentially remains unidentified (or pseudonymous).

If the user does not have the PII recipient's public key (certificate), then the user could ask the PII recipient to give her a secret or unique identifier which she will give to the repository when she is delegating access to the PII recipient. The repository stores the identifier and/or encrypts it into the authorisation token, which the user then gives to the PII recipient. The PII recipient contacts the repository, is asked to provide the secret/identifier, and from this the repository identifies/authenticates the PII recipient. This method is only as strong as the secret/identifier since it relies on this not being easy to guess by an attacker. It also requires more protocol and message exchanges than the secret token method. This method can be combined with the secret token method to improve the security of both schemes, but the PII recipient still remains unidentified to the repository (unless the user tells the repository who the PII recipient is to be).

5.3 Secret URLs

Clearly bearer tokens are less secure than delegated authorisation tokens or sticky policies, but they have the advantages that they are easier for the repository to generate and are easier for the PII recipient to use. The also require a minimum of protocol message exchanges. One form of bearer token is the secret URL. The secret URL is based on the REST principles [1], in which the Internet is a finite state machine, and URLs are pointers to documents in various states. A PII asset is created by the repository (or copied from an

existing PII asset) along with its access control and privacy policy that describes its state. (Note that this is not a "sticky" policy in the sense that there is no mechanism to force an anonymous PII recipient to enforce it, although its contents could be the same as a conventional sticky policy.) A unique secret URL is then created which points to the PII asset in this state. The secret URL is transferred confidentially to the PII recipient either directly by the repository or indirectly via the user (e.g. embedded in a compound object) and by using this URL the recipient is able to access the PII asset according to the terms of the policy at this secret URL. Various policy conditions can be attached to the state of the PII asset at this URL. For example, it could be one-time access only, in which case the PII asset disappears from this state immediately after it has been accessed. Alternatively it could be time-limited access, and after the specified time expires the PII asset leaves this state. The secret URL model is not as secure as methods which require the PII recipient to authenticate to the repository, because there is nothing to stop a PII recipient copying the secret URL to someone else and abusing his permission. However, it may well be sufficient for many elements of e-portfolios. The Share URLs generated by Digitary [5] are of this type.

5.4 Identifying the delegate

If the user (i.e. delegator) knows the identity of the delegate, and the delegate is able to authenticate to the repository, then the user can inform the repository (or a trusted delegation service acting on its behalf) who the delegate is. Once the delegate has been identified by the user, she can subsequently authenticate to the repository in order to gain access to the user's resource. Identifying the delegate could be based on her roles or attributes (for role based and attribute based access controls), or on her unique identifier at a trusted IdP (for discretionary access controls). However, suppose the user does not know the login identifier of his chosen delegate, which is very likely. Or suppose that even though the user does know the role of his intended delegate, he does not wish anyone (or everyone) with this role to be given access. He only wishes his one chosen role occupant to be granted access (lets call this targeted RBAC). How is identification in these cases to be accomplished?

In the case of targeted RBAC, we can use parameterised roles, in which a parameter of the role identifies the specific role occupant e.g. role=doctor=David's GP. Providing that only one role occupant is assigned each parameter value, then role based access can be targeted at one specific role holder. This is a requirement of the Kenteq use case (see D1.2 [16] for further details.) In order for this to work effectively, both the delegator and the role assignment authority will need to agree on the parameter values that are to be used.

In the case where the delegator does not know the unique identifier of the delegate, one solution is for the delegator to obtain an invitation token from the repository (or delegation service). The invitation token is similar to a secret URL, in that it uniquely identifies the resource to which access is to be granted, and it should be complex (secret) enough so that it cannot be guessed by an attacker. However, its purpose is somewhat different. Unlike a secret URL, the invitation token does not directly grant access to the resource. Instead it invites the recipient to authenticate herself to the repository (or delegation service), after which access will be granted to her. Invitation tokens are used when the delegate is able to authenticate to the repository, secret URLs are used when the delegate is not.

5.5 Allowing third party annotations (append access)

So far we have primarily discussed providing PII recipients with (implied read) access to a user's PII asset. Providing update access needs further discussion. Whilst the actual process of authorising update access to third parties is identical to the process of authorising read access to PII recipients, the only difference being the mode of access, there are additional factors that need to be considered. Firstly what sort of update access should be granted to the third party? We can categorise update access into two different types:

- append access. This allows the accessor the ability to add information to a document but not to delete any existing information from the document,

- write or update access. This allows the accessor to add additional information to a document and also to delete existing information from a document (including all of the existing information).

In the context of TAS^3 documents, only the first type of delegated update access is desirable. For example, a referee may be asked to append a reference to an e-portfolio document, or a doctor may append a new episode to a medical record. However, under no circumstances should such a delegated third party be allowed to update or alter existing information in a TAS^3 document. Only the authoritative source (document originator) should be allowed to do this. If for example a referee sees an error in a user's e-portfolio document, or a doctor sees an error in a medical record, they can obviously comment on this in the input that they append to the document, but they should not be allowed to correct the error. Similarly the PII subject must not be allowed to update/alter any information that has been appended to her PII by a delegated third party.

The above can be summarised as: ***only authoritative sources should be allowed to update the information for which they are the source, but third parties may be allowed to append information to the data provided by an authoritative source***.

The enforcement of append access must be carried out by the repository that is holding the PII asset. When the PII asset is copied from the repository, then the sticky policy must be copied with it, so as to ensure continued enforcement of the user's policy. Note that continued enforcement cannot be guaranteed once the PII asset leaves the TAS^3 trust network, just like the enforcement of digital rights management can never be fully guaranteed once the digital content is in the user' PC. The use of sticky policies and TAS^3 network agreements/contracts offer maximal protection for minimum cost whilst the PII asset remains in the TAS^3 trust network.

Appending information to a PII asset leads to a second issue. If a third party appends information to a PII asset, should the updated PII asset be covered by the same sticky policy as the original PII asset, or should separate rules apply to the original PII asset, the updated PII asset and the additional information that has been appended? We propose the following solution. If the additional information is appended to the original PII asset *in situ* and thereby merged with it, we suggest that the same sticky policy should apply to the extended PII asset as to the original PII asset. If however the additional information is included in the PII asset by reference only i.e. by adding a URL to the original PII asset, then the additional information can be provided with its own access control and privacy policy by the authoritative source of the additional information. This policy is independent of that of the original PII asset and will be enforced independently by the TAS^{3}repository that is pointed to by the URL. For example, if the embedded URL is a secret URL, then whoever has been granted access to the original PII asset would be granted access to the additional information, subject to the policy held at the secret URL. If the URL is a public URL, then the repository holding the additional information would normally request the accessor to authenticate to it, before checking the sticky policy of the appended information, to see if the accessor is granted access or not.

5.6 Correcting Information in an Authoritative Source

It may be that an authoritative source actually holds incorrect information. This could be due to human error, mal-administration or some criminal act. How can a data subject correct her PII when she is not the authoritative source of the information? The answer is that authoritative sources should provide a user friendly interface for PII subjects to contact them and request that the information be corrected. This is usually a requirement of data protection legislation anyway, and so for those countries that already have data protection legislation, repositories will be already required to do this. We will make this functionality a mandatory requirement of TAS^3 repositories. TAS^3 repositories must allow PII subjects to authenticate to them and to either request that erroneous PII be corrected (when the PII

subject is not authorised to update the information directly) or correct it themselves. Use Case 6 in Appendix 1 addresses this issue.

5.7 Recursive Authorisation

Whilst the previous sections have addressed the issue of a user authorising a PII recipient to obtain direct access to her PII asset stored in a repository located at some URL, we also need to address the issue of recursive authorisation. By this we mean how is a PII recipient who is granted access to some PII asset located at some URL, to be granted access to the embedded PII assets that are referenced within the original PII asset, because the original PII asset is a compound object containing embedded URLs. The embedding of URLs in compound PII objects may be infinitely recursive, as in today's world wide web.

Since the embedding of URLs in electronic documents, such as e-portfolios, is recursive, then the authorisation mechanisms described above similarly must be applied recursively each time a PII recipient clicks on an embedded URL. A PII subject must be able to go to each of the repositories that hold her PII assets, and give them instructions (i.e. sticky policies) of who to release her PII asset to and for what purposes.

If the repositories are able to authenticate the PII recipients, then the PII subject can set a sticky policy in each repository for each of her PII assets. If the PII subject does not wish to provide proof of ownership (see next section), then the PII subject can simply copy and paste the URLs of her PII assets into a compound PII object. The URLs embedded in the compound document are therefore "public" URLs (public in the sense that anyone can see and use them), and "unfixed" URLs (unfixed in the sense that they can be used in any referencing location and copied and pasted and moved between repositories). If the PII subject wishes to provide proof of ownership, then the PII subject may need to ask the repository to create a proof of ownership "fixed proof" URL (fixed in the sense that this URL is fixed to the referencing location and can only be used from there. It cannot be moved between repositories). It is still public since any PII recipient can use it. Remember that proof of ownership does not mean that the PII subject has to reveal her true identity to the PII recipient, but is simply a way of proving that all the linked PII assets belong to the same person. Thus the user's identity can still be privacy protected even with proof of ownership. Each time a PII recipient clicks on a public URL (unfixed or fixed proof), he will be asked to authenticate to the repository (unless the repository provides public access to its PII assets). After authentication, the repository will pass the PII subject's sticky policy to the policy decision point (PDP) and ask the PDP if the PII recipient is authorised to access this PII asset. If the answer is granted, the repository will return the PII asset and its policy to the PII recipient, optionally along with proof that the PII asset does belong to the PII subject. In scenarios like this, the benefit of federations and single sign on (SSO) become immediately obvious, since the click through on URLs and the authentication of the PII recipient will take place automatically after the first authentication interaction has taken place.

If a repository is unable to authenticate a PII recipient, then the repository will need to issue a "secret" URL to the PII subject. This URL is secret in the sense that it is a bearer token and must only be seen by the rightful PII recipient. Once a PII recipient clicks on a secret URL he will gain immediate access to the PII asset. If the PII subject needs to embed a secret URL in her compound PII object, this means that the compound PII object then becomes "secret" and will need protecting by its own sticky policy. The embedding of secret URLs in compound documents which are themselves accessed by secret URLs will give unlimited recursive access to a PII recipient once he has the secret URL of the root document. Note that secret URLs may eventually fail to grant access to a PII recipient if the policies that have been attached to the secret URLs don't provide unlimited access, for example there is a time dependency or fixed number of allowed accesses associated with the PII asset in this state.

Secret URLs may be unfixed or provide fixed proof of ownership. An unfixed secret URL can be copied and pasted into multiple compound PII documents, and moved between

repositories. A fixed proof secret URL can only be embedded in the compound object for which it was issued, and can only be dereferenced from there. It cannot be embedded in multiple compound objects and cannot be moved between repositories.

The following table summarises the different types of URL

Type of URL	Unfixed	Fixed Proof
Public	Can be copied between repositories and PII assets. Will require the PII recipient to authenticate and be authorised before access is granted to the referenced PII asset	Are fixed for the repository and PII asset they are created for. Will require the PII recipient to authenticate and be authorised before access is granted to the referenced PII asset
Secret	Can be copied between repositories and PII assets. Will grant access to the referenced PII asset to anyone who gains access to this URL	Are fixed for the repository and PII asset they are created for. Will grant access to the referenced PII asset to anyone who gains access to this URL

Table 1. Properties of different types of URL

5.8 Providing Proof of Ownership

Users typically have different identifiers at different repositories, authoritative sources, IdPs and other service providers. Thus no-one can necessarily tell that two service accounts belong to the same physical person, other than the person herself. When a user creates a compound PII object by aggregating PII assets together, which are held at various authoritative sources around the Internet, the recipient of the compound PII object has the problem of determining that all these PII assets, which are bound to different user identifiers at the different repositories, all belong to the same person. This problem is similar to the problem that was faced by attribute aggregation for authorisation as described in Deliverable 7.1 [7]. However, there is one difference in the repository scenario, that of synchronicity. In attribute aggregation for authorisation the user was authenticated and the user's session was active when the relying party (the service provider) received all the aggregated attributes of the user from the various authoritative sources. The process was synchronous, and took place in a single user session. The relying party was assured that the user had been authenticated and the aggregated attributes were being collected and used to authorise the user to access its resource, all within the same session. However, in the case of aggregated PII assets from different repositories, retrieving the compound PII object by the relying party is asynchronous to the creation of the compound PII object by the user. In most cases the user will not be logged in when the relying party (the PII recipient) wishes to access the compound PII object. Therefore neither the PII recipient nor the repositories are able to authenticate the user (PII subject) during PII asset retrieval. Consequently PII subject authentication and the binding of the subject to the compound PII object and all its constituent PII assets, has to take place during the creation of the compound PII object. Information about this binding has to be remembered at the time the compound PII object is created, so that it can be validated at PII asset retrieval time by the PII recipient. The PII recipient will need to trust the various authoritative sources to perform this binding at construction time and provide it with proof of ownership at retrieval time. Proof of ownership is designed to provide the PII recipient with the assurance he needs, and thereby counter the cheating data subject scenario.

Conceptually the following solutions (at least) are possible:

i) Assign the user a globally unique permanent ID and bind all the PII assets to this one ID e.g. a national ID card identifier.

ii) Assign the user a globally unique temporary ID (or let the user choose her own alias) when she creates a compound PII document. Bind all the linked PII items to this random ID, so that when a PII recipient accesses the PII items the repositories will say that all the items belong to the same random ID (or alias).
iii) Assign the user a locally unique permanent ID for each element of PII and when the user links one item of PII to another store the PID mapping in the link. When the linked item is read return the mapping to the PII recipient. Furthermore, embed the ID of the user into the URL that references the top level compound PII object. The PII recipient can then validate that this URL belongs to the user who gave it to him. The IDs of the user should be encrypted in the URL, using a secret key of the referenced repository, in order to protect the privacy of the user. When the repository returns the PII asset to the recipient who uses the URL, it can decrypt the PID(s) and provide the identifier(s) of the PII subject to the recipient.
iv) When two PII items are linked together, assign the user a locally unique temporary ID for each PII item and store this in the link. Return this mapping when the PII recipient retrieves the linked PII item. Embed a temporary ID of the user into the URL of the top level compound PII object. The PII recipient must validate that this temporary ID belongs to the user when she gave him the top level URL.
v) When two PII items are linked together, then if they both belong to the same user the referenced repository securely embeds the URL of the referencing compound document into the URL of the referenced document (i.e. a back pointer). When the link is de-referenced the repository returns the referencing URL to the PII recipient. Similarly the ID of the user is embedded into the URL of the top level compound object. When this top level URL is used, the top level repository can return the user ID to the PII recipient who can then validate that this belongs to the user who gave it to him.
vi) The user has one or more asymmetric key pairs and (possibly self signed) certificates. Each authoritative source binds the subject's PII asset to one or more of the user's public keys as requested by the user and returns this when the PII asset is retrieved. The user sends her certificate and signed top level URL to the PII recipient, who can then validate that all the PII items belong to her. (The PII items are still signed by their authoritative sources when they are retrieved.)
vii) Create a trusted directory which knows about the user's assets at the various locations. The directory knows the user's various identities and it vouches to the PII recipient that they all belong to the same user.

Currently there is some debate within the TAS^3 project as to which of the above seven methods should be supported. **This is currently an unresolved issue.** Some members believe that all mechanisms should be supported if at all possible and the authoritative sources should choose their preferred method(s). Other members believe that the first solution should be rejected since it does not sufficiently protect the user's privacy, and that the fourth solution is probably unworkable since there is nothing to link together one compound PII document which is embedded in another compound PII document. This is because each pair of linked IDs is temporary and refers to that link only, and further, how does the PII recipient validate that the temporary ID in the top level link really belongs to the user? Likewise the second method is also probably unworkable, since although it can successfully allow a single compound document to be created, it cannot allow the same compound document to be embedded in two different compound documents as the alias names would not tally. Therefore in practice the user would either need to use the same alias for all compound documents, or have disjoint sets of compound documents with each set using the same alias to reference other documents in the set. Concerning method vii) we have not yet fully worked out the mechanics of how this might work. **This is for further study**. Currently methods iii), v) and vi) are known to be technically viable and privacy preserving whilst providing proof of ownership, whilst i) is viable but is not privacy preserving. The following table compares and contrasts methods iii), v) and vi) from privacy preserving and usability perspectives.

Functionality	Local Permanent ID	URL back pointers	Key Pairs
Ease of use to PII Recipient	PII Recipient has to check that all the mapped IDs in the linked PII items tally and that the top level ID belongs to the user	PII Recipient has to validate all the URL back pointers and that the ID of the user in the top URL belongs to the user	PII Recipient has to authenticate the user's signature and check that the user's public key is bound to every PII item
Ease of Use to PII Subject	Subject can link PII assets in any arbitrary mesh. System takes care of ID mapping. Documents cannot be copied between repositories as links are fixed. User must give his top level ID to PII recipient	Subject can link PII assets in any arbitrary mesh. System takes care of back pointers. Documents cannot be copied between repositories as links are fixed. User must give his top level ID to PII recipient	User has to manage key pairs and provide appropriate keys to repositories and to PII recipients.
Privacy protection of user's identity	User's IDs at all repositories are visible to PII recipient.	User's top level ID is visible to PII recipient	User's public key certificate is given to PII recipient

Table 2. Comparison of Various Privacy Preserving Proofs of Ownership Methods

It would appear that none of these three solutions are ideal from either a usability or privacy protection perspective. Those users who are unable to properly manage asymmetric key pairs are almost bound to use one of the other two schemes. Local permanent IDs and URL back pointers are very similar methods from technical and usability perspectives, the only difference being that the links contain either the IDs of the user at each end of the link, or the URLs of the documents at either end of the link. From a privacy protection perspective URL back pointers would appear to be the better method. We propose to investigate method vii) next to see if it is any better. Appendix 2 shows how proof of ownership might work using permanent IDs. It should be relatively easy for the reader to see how this could be changed to use URL back pointers instead of linked IDs.

6 Privacy Policies

Privacy policies add further constraints on the usage of PII assets. They typically express conditions on: the purpose of data usage (e.g., research, marketing, auditing), the retention period, any secondary uses (to who the data may be disclosed), arbitration in the case of disputes, etc.

In particular, they enable the user to determine the basic information about the handling of her PII data:

- who is collecting the data
- what items of personal data are collected
- the purpose of the collection
- the data retention period
- possible additional data recipients beyond the data collector, including the purposes that they will use the data for.

Additional information may also include obligations the data collector commits to, a pointer to a human readable version of the privacy policy, the possibility to opt-in or opt-out for some data usages, etc.

To illustrate how a privacy policy may be used, let us consider the case of a PII subject who wishes to create or transfer some PII into a repository (for more details see Use case 2, in Appendix 1). At first, the user needs to gain access to the repository, through one of the authentication and authorization mechanisms described in Section 4.1 and Section 4.2. Then she has firstly, to evaluate the repository's privacy policy. In most of cases this means that the repository's privacy policy is shown to the user as natural language text, and the user can accept it or not depending on her privacy preferences. Typically, if the user does not accept it, she is not allowed to proceed.

Although the TAS^3 infrastructure should support this simple mechanism, this process should be automated to enable more complex interactions to take place, such as supporting the user in tracking the usage of her PII, and facilitating enforcement of her privacy policy.

In the above scenario, the repository's privacy policy may be expressed using a machine readable policy language, then automatically compared with the user's privacy policy (often called *privacy* preferences [9]), which can be generic (i.e. not dependent on any specific PII asset) or specific to each PII asset she wants to transfer. The outcome of such comparison could be that the two policies (repository's and user's) are an exact match or, at least not conflicting, and the user can proceed.

In general there are three options: in the simplest case, the user cannot modify the repository's privacy policy (as in P3P [9]), so she may either simply accept it, and might have to relax her own privacy policy to do so, or else the transaction is aborted. In the more complex case, the repository may allow the user to add some constraints to its privacy policy (e.g., PRIME DHP [10]), or might even allow the user to present a complete sticky policy with the PII asset. This is the approach we are adopting in TAS^3. In the most complex scenario policy negotiation might take place. This would occur when the user wants to trade-off her privacy for some additional features of the repository service, such as the speed of the service, and the service is capable of performing such negotiations [11]. In this case an additional negotiation step must take place, where each side says what its requirements and capabilities are, and if they match the negotiation successfully concludes. This approach may also be adopted in TAS^3.

The development of a specific language for expressing privacy policies is beyond the scope[4] of TAS^3. In TAS^3 the authorization infrastructure, as described in Deliverable D7.1 [7], is being designed to handle privacy policies in any language as an additional type of authorization policy. In other words, a specific privacy PDP, which is able to evaluate a privacy policy as written in an existing privacy policy language (e.g. P3P, EPAL, PRIME DHP), should be callable by the Master PDP and return its decision to the Master PDP. Potential conflicts between decisions coming form the privacy PDP, the access control policy PDP and the trust PDP have to resolved at a higher level i.e. the Master PDP, which sends back the final decision to the Application Independent Policy Enforcement Point (AIPEP). Consequently, the enforcement of privacy policies does not differ appreciably from that of other sticky policies, as described in D 7.1, Chapter 8.

[4] A parallel FP7 European Project, PrimeLife, (www.primelife.eu), is currently developing a privacy policy language, addressing the mentioned issues. TAS3 may integrate a PrimeLife PDP, when available.

7 The Delegation Service

The delegation service is used to allow a user (the delegator) to delegate a permission to another user (the delegate). A full description of a general purpose delegation service is given in section 6 of Deliverable D7.1 [7]. In the case of repositories, we assume that users do not have asymmetric key pairs, and therefore any token based delegation will need to be indirect, in that the Delegation Service will issue the delegate's tokens on behalf of the delegator. The easiest implementation of a delegation service is for the delegation service to know all the users and to provide the delegator with a picking list or search function to allow her to choose the delegate. The delegator can then delegate to the delegate without involving the delegate in the process. If the delegate is unknown to the delegation service and is also unable to authenticate to the delegation service via a known and trusted IdP, then the delegate will need to register as a user with either the delegation service or a trusted IdP before delegation can take place. If the delegate is unknown to the delegation service but is known to a trusted IdP then either an invitation token can be given to the intended delegate by the delegator, or the intended delegate can give the delegator her name/identifier at her IdP, and the delegator can then enact the delegation directly without involving the delegate further in the process. In this case the IdP will need to refer to the delegate by this identifier when communicating with the delegation service. An example of a delegator delegating a role to a delegate is given in Use Case 5 of Appendix 1.

However, not all delegation of authority needs to be via a token issued directly or indirectly by the delegator. Instead, the delegator can login to the repository holding her PII asset and directly give access rights to a delegate by updating the sticky policy. The intended delegate can then be referred to by name/identifier, by role, or anonymously. This is described in Use Cases 3 and 4 in Appendix 1.

8 Audit Trails, Logging and Transparency

8.1 Audit log files

Repositories need to keep audit trails of who created, edited and deleted which data (documents or PII assets) and who accessed which data, along with their intended purposes. The same audit information is needed for changes to the sticky policies that are attached to the data. A repository (or its authorisation system) also needs to keep a record of all authorisation requests that were made and all authorisation responses that were returned, including denies and any returned obligations. The audit service that manages the audit trail has the following set of requirements:

- Append mode of access: Only append mode of access should be allowed, so that users or applications cannot rewind an audit log file and delete or modify information that has already been stored there
- Authorised writing: Only authorised parties should be able to append log records to the audit trail. Though unauthorised applications or attackers may gain access to the audit trail and try to append fake log records to the audit trail, or modify or remove the audit trail, this should be detected by the tamper detection mechanism.
- Timestamps: Every record in the audit trail should be timestamped to provide a trusted record of when the audit data was received. We note that if the audit service is trusted to record the audit data without tampering with it, then it should also be trusted to append the correct time to the data. Therefore we do not require a secure time stamping service.
- Secure communication: if the audit service operates as a web service then there should be secure communications between the clients and the server in order to ensure tamper resistance, data integrity and authorised connection.

- Secure storage on untrusted media: Since an audit trail may be viewed on untrusted machines, the security mechanisms should ensure persistent and resilient storage of the audit trail, and ensure detection of tampering of the audit trail – modification, deletion, insertion, truncation, or replacement. If tampering is detected, the audit service should be able to notify the security auditor.
- Support multiple simultaneous clients. The audit service should be easily and conveniently accessible and it should be able to serve multiple client applications simultaneously.
- Logging efficiency: The computational work and the storage size required by the audit service should be as efficient as possible.
- Contents transparency: the audit service should be able to record any digital content coming from any repository service.
- Authorised reading: Since the audit trail may contain personal or sensitive information, then the audit service should ensure that only authorised applications or people have the privilege to read the audit trail. The audit trail may be encrypted to further protect confidentiality.

Each repository should keep its own audit trail and this audit trail should only be read accessible to authorised people. Once an audit log file is written it should be taken offline and securely stored on physical media. A backup copy should be available at a different physical address to guard against loss due to fire, flooding, earthquake etc. Audit trails should be destroyed after a fixed (published) amount of time.

Only duly authorised personnel should be able to access the log files of multiple repositories, and then only after all legal obligations are fulfilled.

8.2 Transparency and Consent

PII subjects should be able to find out who has accessed their PII and when. This could be enacted by either notifying them at the time of access, or instead, giving them direct or indirect access to the audit trails in order to validate this information. Alternatively, when an audit trail is destroyed, users could be sent a summary of the accesses that took place to their PII and were recorded in the audit trail.

Audit trails should be destroyed after a fixed (published) amount of time and the PII subjects should be informed about this (or be able to inquire if this has been done.)

9 Mapping to Architecture and Protocols

Various types of repositories are depicted in the architecture: policy stores, trust/reputation stores and PII stores. Whilst the primary design aim is to produce secure repositories that are suitable for storing PII assets, nevertheless they can be used as the repositories for the other types of TAS^3 information without any changes to this design. The other types of repository will typically require fewer features than those needed for storing and linking together PII assets.

The repository is designed to be web services enabled and therefore is designed to fit easily into the TAS^3 architecture. Legacy repositories that do not have a web services interface can be connected to a TAS^3 trust network via a SOA gateway. If the legacy repository does not natively support the TAS^3 repository functions, such as sticky policy storage, then the SOA gateway will have to be responsible for providing this.

Every TAS^3 compliant repository MUST conform to the specification given in section C7 of the TAS^3 Architecture document [14].

9.1 Authentication/SSO

A repository user may have an account with the repository and may be able to login directly to it, or may not have an account with the repository. In both cases the preferred mode of operation is for the user to authenticate to the repository via a TAS^3 IdP. This will provide the user with SSO to other services in the TAS^3 trust network and will simplify the process of linking PII assets together. Authentication via an IdP is described in Step 4 of Use Case 1 in Appendix A. The protocols that may be used for IdP authentication are described in section A.1 of [14]. If the user does not have an account with either the repository or an IdP, the user will be invited to register with the repository as a new user, as described in step 4c) of Use Case 1 in Appendix A. The protocol for this is proprietary to the repository and is not standardised by TAS^3. However the link between the repository and the client must either be trusted or confidentiality protected using a strong encryption protocol such as SSL or TLS, in conformance to requirement CR26-SSL in Annex C of [14].

9.2 Data Retrieval and/or Manipulation

After login/authentication, the user may read existing data, upload new data, modify existing data or delete existing data in the repository. Reading data that the user does not own is described in Use Case 8 of Appendix A. Reading or modifying data that the user does own is described in Use Case 2 in Appendix A. Appending information to data that the user does not own is described in Use Case 9 of Appendix A. The protocol for all of these is repository specific and is not standardised in the TAS^3 architecture or protocols. However we expect that in most cases the user/client will be using a standard web browser and the protocol will be HTTP. The only requirement is that the link is either trusted or confidentiality protected using a strong encryption protocol such as SSL or TLS, in conformance to requirement CR26-SSL in Annex C of [14].

9.3 Setting Sticky Policies

After login/authentication the data owner may set the sticky policies on his data. Giving a third party read access to data is described in Use Case 3 in Appendix A. Giving a third party append access is described in Use Case 4 in Appendix A. Granting other types of access is **expressly forbidden** by the TAS^3 architecture. The protocol for setting the sticky policy is repository specific and is not standardised in the TAS^3 architecture or protocols. However we expect that in most cases the user/client will be using a standard web browser and the protocol will be HTTP. The only requirement is that the link is either trusted or confidentiality protected using a strong encryption protocol such as SSL or TLS, in conformance to requirement CR26-SSL in Annex C of [14].

9.4 Linking Data Items Together

After login/authentication a data owner may link PII assets in other repositories to the current compound PII object. This is described in Use Case 2 of Appendix A. The back end web services protocol that takes place between the two repositories must use SSL/TLS mutual authentication according to CR216-MutualEntAn, CR27-Sig and CR28-Vfy of Annex C of [14]. The precise application level content of the protocol is still *To Be Defined*. The front end protocol between the client and the two repositories is standard HTTP redirects or Form-POST (depending upon the size of the content being transferred). The application level content of the protocol is still *To Be Defined*.

9.5 Delegation of Authority

If a user has used role based access controls (RBAC) when creating his sticky policy, then he may need to assign or delegate a role to someone in order to authorise them to access his

data. The assigning or delegating of a role to a third party is described in Use Case 5 in Appendix A. The model for delegation of authority is described in Chapter 6 of Deliverable D7.1 [7]. The web services protocol for accessing the delegation service must use SSL/TLS mutual authentication according to CR216-MutualEntAn, CR27-Sig and CR28-Vfy of Annex C of [14]. The precise application level content of the protocol is still *To Be Defined*.

9.6 Removing an Account and Its Data

After login/authentication a user may wish to delete his data from a repository and remove his account. This is described in Use Case 7 of Appendix A. The protocol for this is repository specific and is not standardised in the TAS^3 architecture or protocols. However we expect that in most cases the user/client will be using a standard web browser and the protocol will be HTTP. The only requirement is that the link is either trusted or confidentiality protected using a strong encryption protocol such as SSL or TLS, in conformance to requirement CR26-SSL in Annex C of [14]. The repository SHOULD confirm to the user, via an out of band means, that the account and all its data are to be deleted, and receive an acknowledgement from the user via the same means, before the deletion actually takes place. Out of band could be via an email message to the user's mailbox, or an SMS to his mobile phone or a written letter to his postal address etc. Once the deletion has taken place the repository MUST remove all copies of the data including all back up copies so that it is not possible to recreate the user's account again.

10 Conclusion

This document describes the concepts and services of a secure repository service so that the user's privacy can be ensured. The document describes how the user may grant access rights to third parties and attach sticky policies to her data (PII asset). It describes how a compound object may be created by securely linking together PII assets held in different repositories in such a way that the repositories can offer proofs of ownership of the compound document and all its component PII assets to the PII recipients who are authorised to access the compound object. Finally this document describes how the services of secure repositories are mapped onto the protocols and architecture of the TAS^3 infrastructure.

11 References

[1] Representational State Transfer (REST) principles. See http://en.wikipedia.org/wiki/Representational_State_Transfer and http://en.wikipedia.org/wiki/REST

[2] HM Govt "Data Protection Act 1998" Available from http://www.opsi.gov.uk/Acts/Acts1998/ukpga_19980029_en_1

[3] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on "the protection of individuals with regard to the processing of personal data and on the free movement of such data". Available from http://europa.eu.int/eur-lex/lex/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:EN:HTML

[4] The BBC "UK's families put on fraud alert". 20 Nov 2007. See http://news.bbc.co.uk/1/hi/uk_politics/7103566.stm

[5] See http://www.digitary.net/

[6] See http://www.openarchives.org/

[7] TAS^3 Deliverable 7.1. "Design of Identity Management, Authentication and Authorization Infrastructure" v1.2 26 April 2009.

[8] OAuth Core Workgroup. "OAuth Core 1.0" 4 Dec 2007. Available from http://oauth.net/core/1.0/

[9] W3C. "The Platform for Privacy Preferences 1.0 (P3P1.0) Specification" Technical Report. 2002. available at http://www.w3.org/TR/P3P/ (accessed 24 October 2008)

[10] Ardagna, C. A., Cremonini, M., De Capitani di Vimercati, S., and Samarati, P. 2008. A privacy-aware access control system. J. Comput.Secur. 16, 4 (Sep. 2008), 369-397.
[11] U.M.Mbanaso, G.S. Cooper, D.W. Chadwick, Anne Anderson. “Obligations of Trust for Privacy and Confidentiality in Distributed Transactions”. Internet Research. Vol 19 No 2, 2009, pp 153-173.
[12] Xavier Huysmans and Brendan Van Alsenoy. "Identity Management for eGovernment. Deliverable 1.3. Conceptual Framework. Annex I. Glossary of terms (v1.07)" 17 Dec 2007. Available from https://projects.ibbt.be/idem/index.php?id=161
[13] ITU-T."Security Architecture for Open Systems Interconnection for CCITT Applications". CCITT Recommendation X.800, 1991
[14] TAS^3 Deliverable D2.1- TAS^3 Architecture, 1 June 2009
[15] 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.
[16] TAS^3 Deliverable 1.2. “Requirements Assessment Report, Version 2” 31 May 2009

12 Appendix 1– Repository Use Cases – Action Sequences

The TAS^3 infrastructure is based on shared web services, where all users potentially have access, directly or indirectly, to all web services (subject of course to appropriate authentication and authorisation). Users access the various web services via their web browsers. Browsers present the PII content of web service repositories to users without any additional client software being needed, except that occasionally browsers may need additional plug-ins to support new types of PII content as they become available. New types of content will arise from the natural evolution and innovation of the World Wide Web. Users and service providers will dynamically chose to use new types of content as it becomes available.

Two types of repository users are identified: PII recipients and PII subjects. PII subjects are the users to whom the PII refers, and they set their own policy for access to this PII. PII recipients are users who have been delegated access to the PII of PII subjects.

TAS^3 repository services are available within a TAS^3 trust network, for the PII subjects to use and personalize. PII recipients may be members of a TAS^3 trust network when they are delegated access to a PII asset or they may not be. If they are not, the TAS^3 repository holding the PII asset may invite the PII recipient to join the TAS^3 trust network before she is granted access to the PII asset. The PII recipient is free to decide whether to join or not.

Action sequences for the following use cases (tasks) are described.

PII Subject Use Cases

- Use Case 1. PII Subject Registers at a Repository and enters his PII Attributes
- Use Case 2. PII Subject Creates a PII Asset at a Repository
- Use Case 3. PII Subject authorises a PII Recipient to be Granted Read Access to a PII Asset
- Use Case 4. PII Subject authorises a Trusted PII Recipient to be Granted Append Access to a PII Asset
- Use Case 5. Delegating/Assigning a Role (group membership) to someone.
- Use Case 6. PII Subject viewing and correcting his PII Asset in a Repository
- Use Case 7. PII Subject De-Registering from a Repository.

PII Recipient Use Cases

- Use Case 8. PII Recipient (read) accessing a PII asset and its embedded contents.
- Use Case 9. PII Recipient annotating a PII asset.

The following sections provide more detail for each use case.

12.1 PII Subject Use Cases

12.1.1 Use Case 1. PII Subject Registers at a Repository and enters his PII attributes

In this use case, the user is contacting an online repository for the first time, and is registering there as a new user and opening an account. He optionally may want to refer to another existing repository (authoritative source) containing authentic information about him, for inclusion by reference in this repository. Note that we do not consider the use case of a PII subject registering with an Authoritative Source since this process is typically carried out completely out of band, usually via some face to face meeting between the two parties.

1. The user navigates to the welcome page of the Repository.
2. The user clicks on the *New User Registration* hotlink.
3. The user is shown various terms and conditions which he must agree to before he is allowed to create a new account at the repository[5]. For example, if the repository has its own privacy policy, this is displayed. If the repository has over-riding conditions, for example, that it always collects links between repository users and releases these to law enforcement upon request, these are displayed. After the user has agreed to all these terms and conditions, he is sent to the Create New Account page.
4. The user is presented with the option to either select an IdP (trusted by the Repository to authenticate users) from a picking list, or click the *I have an Information Card* button (if the user has a set of Information Cards on his PC), or fill in the details of a new user account (if the user either does not have or does not wish to use an existing IdP account).
   a) If the user selects an IdP from the picking list, the user's browser is redirected to the login page of the IdP, the user authenticates to the IdP and is then redirected back to the My Profile page of the repository.
   b) If the user selects *I have an Information Card*, the user is returned to the Card Selector in his browser, selects the card he wishes to be returned to the repository, enters his login credentials for the chosen IdP, and is then redirected back to the My Profile page of the repository.
   c) If the user chooses to fill in the details of a new user account, the options here will be chosen by the repository. Typically it will entail the user entering a username (which might be his email address), a password, and possibly his email address. The repository may validate the email address by sending a secret to it before allowing the user to login, or it may allow the user to login straight away using his chosen username and password. After the user has successfully logged in, the user is directed to the My Profile page of the repository.
5. When directed to the My Profile page the only information that the page will already contain will be the account name/identifier for the user, which was either returned by the IdP (4 a) and b)) or entered by the user himself (4 c)), plus any attributes that the IdP has returned. All of these will be greyed out, since the user is not allowed to change them himself. The actual contents of My Profile page is repository specific, but it allows the repository to collect, directly or indirectly, various attributes of the user. Some of these attributes may already exist at other authoritative sources, whilst the user himself may be authoritative for some of the attributes. Attributes in the latter category may be entered directly by the user into the My Profile form. It is a local matter whether the repository chooses to verify these e.g. by sending a message containing a secret to the user's postal or email address. Attributes in the former category will be indirectly collected by reference to the authoritative sources, using the steps outlined below.
6. The user is asked to select or enter the DNS name of the authoritative source containing the user's PII attributes (this could be from a picking list of trusted authoritative sources or a free form field for the user to type in). **NOTE. It is a repository specific issue which authoritative sources are trusted by this repository. The repository may**

[5] Precise details to be completed by legal experts of the project

have been provided with a list when it joined the TAS[3] network, or it may have created its own list by private mechanisms.

7. The user is redirected to the authoritative source and may be asked to authenticate, unless SSO is in operation. The user may be transferred immediately to the page containing his attributes, or he may have to navigate to the page, depending upon the particular authoritative source and its contents.
8. Once at the page containing his attributes, the user clicks on *Link My Attributes.* This causes the authoritative source to generate a fixed proof of ownership URL which is transferred directly back to the original repository for it to automatically embed in the user's My Profile page[6]. The user is then redirected back to his My Profile page. Note that the proof of ownership URL should be greyed out and should not be editable by the user. Once back at the My Profile page the user can continue to add further PII attributes or log out or start to add PII asset to the repository (as in Use Case 2).

12.1.2 Use Case 2. PII Subject Creates a PII Asset in a Repository

In this use case the PII Subject creates his own PII Asset, such as a CV, in a repository, and during the process embeds a link to another PII asset that is held in an authoritative source.

1. The user navigates to the welcome/ login page of the repository. The user is presented with several options: to directly login by entering a repository specific username and password, to select an IdP (trusted by the Repository to authenticate users) from a picking list, or click the *I have an Information Card* button (if the user has a set of Information Cards on his PC). After choosing one of these the user is successfully logged in. (See step 4 of Use Case 1 for more details).
2. The user selects the Create New Document button.
3. How a new document is created is repository specific. It may allow the user to directly type in text, to upload files from his PC, to copy URLs to documents residing elsewhere etc. At this point, the user may also attach the sticky security policy to the document, or such policy may be created at later stage (see Use Case 3). There should at least be the following TAS[3] button: *Create Proof Link to Embedded Document*. This button will produce a proof of ownership link to a pre-existing PII asset belonging to the PII Subject. For example, it might be used to create a link to the PII subject's university degree certificate. When the user selects the *Create Proof Link to Embedded Document* button the following occurs.
4. The repository displays a list of trusted authoritative sources with which it has trust relationships[7]. Note that it is not possible to create a proof link with an unknown or untrusted authoritative source. The user selects the source containing his PII asset and the repository redirects his browser to that site.
5. The user may be asked to login to the authoritative source, in a procedure similar to step 1 above, or if the repository and authoritative source both trust the same IdP that the user has already used, and SSO is in operation, the user will be automatically logged in.
6. The user navigates to his PII asset in the authoritative source, and selects the *Create Proof Link to This Document* button. The authoritative source creates the proof URL and returns it directly to the repository, for it to embed in the user's composite PII document. The user is then redirected back to the repository where he sees his proof link embedded in his composite PII asset. This process is shown in Appendix 2 Figure A2.1

---

[6] Note that there have to be safeguards against the user navigating to an authoritative source to which he has read (but not write) access and then pretending that the contents of this page are his own attributes, when they are not. E.g. he could navigate to the page containing the photograph of a film star and ask for this to be embedded in his own Profile as his photograph. Consequently the authoritative source should only show the Link My Attributes button to users who are the owners/subjects of the attributes being displayed. The proof of ownership URL should then be transferred directly by a web services call to the referencing repository, for it to embed directly into the user's profile.

[7] Here we don't say how this list is created. It could be by manual configuration, or a list automatically distributed by the trust network, or dynamic trust negotiation between service providers.

7. If after navigating to his PII asset, the user wishes to set an access control and privacy policy on the asset before creating a proof link, then the user may follow steps 6 to 12 in Use Case 3 after which the *Create Proof Link to Embedded Document* button will be displayed.

12.1.3 Use Case 3. PII Subject authorising a PII Recipient to be Granted Read Access to a PII Asset

In this use case, the user is setting up read and privacy access rights to a PII asset. There are three variants of this use case, depending upon whether the intended PII recipient is able to authenticate to the repository or not, and if he can authenticate, what type of sticky policy (RBAC or DAC) is created. If the PII recipient can authenticate, then the asset repository will need to know either who the recipient is (for DAC), or what role the recipient has (for RBAC), so that this can be stored in the sticky policy of the PII asset. If the PII recipient is not able to authenticate to the asset repository, then the repository will create a secret URL (i.e. bearer token) which can subsequently be used by anyone who possesses it, to access the asset. At the end of the use case the user may choose to create a proof link to her PII Asset, so that any recipient can know that the document really belongs to her.

1. The user navigates to the URL of the PII asset repository. If the user is already logged into the TAS^3 trust network and SSO is in operation, go to step 5.

2. The precise details of what happens at login are repository specific. The user may be redirected automatically by the repository service to its IdP discovery service, or the user may have a choice of selecting between Information Cards and a list of trusted IdPs, or the user may directly enter his username and password into the repository.

3. In the Information Card (CardSpace) model, the user is redirected to the Identity Selector in the user's browser. The user picks the Information Card, and is presented with the login screen. After a short period the user is redirected back to the repository service and is logged in.

4. In the Liberty Alliance model the user might be redirected to a Where Are You From Service, from which the user picks his favourite IdP, or it might be a picking list within the repository itself. The user is redirected to the IdP to authenticate, and after successful authentication the user is redirected back to the repository service and is logged in

5. The user navigates to his selected PII asset. (This assumes the user has a collection of PII assets in this repository). If the user only has one PII asset the repository may present this immediately to the user after successful authentication.

6. The user selects the *Grant Read Access* button and is presented with four choices "Anyone", "Existing User", "Group" and "Anonymous User[8]".

7. If the user selects "Anyone", the interface might pop-up a warning "do you really wish to allow anyone at all to read this information" before creating the sticky policy which grants public read rights to this PII asset. Goto step 11.

8. If the user selects "Existing User", the user is shown either a picking list of existing users at the repository, or a picking list of IdPs trusted by the repository, or a search screen allowing the user to search for an existing user, at the option of the repository. If the user picks an IdP, the repository will then ask for the identity/identifier of the user at the IdP. (The repository may contact the IdP to verify the identity/identifier or may relay a picking list from the IdP to the user, or may perform a search of the IdP. The full mechanics of

[8] This is not necessarily the most appropriate user friendly string which signifies that a secret URL is to be created. If readers have a better suggestion, please don't hesitate to suggest something.

this are still to be worked out). If the user can identify an existing user, then the user IdP/identifier combination is entered directly into the DAC sticky policy for this PII asset. Goto step 11. If the user/repository is not able to identify the PII recipient, then the repository may display a message "sorry unable to identify the user" and could then migrate the user to the Anonymous User page described below.

9. If the user selects "Group", the user is shown a picking list of groups[9] that have been defined in the ontology for the TAS[3] network. For example the list might show "NHS:Doctors, Employers, Kenteq:Facilitators, My Friends, Project Managers". The user can select an arbitrary set of these groups, and only PII recipients who are members of the combined set of groups will be granted access to the PII. The corresponding sticky RBAC/ABAC policy is created. Note that if a particular group membership can be assigned by different Sources of Authority, then the user will additionally be shown a picking list of the Sources of Authority that he may wish to trust[10]. For example, if the user selects Project Managers, then he might be shown the set of organisation in the TAS[3] network who set the PM role within their organisations, e.g. "University of Kent, KU Leuven, Synergetics, Kenteq, Risaris, etc." and the user may choose Kenteq. Then only PMs assigned by Kenteq would be granted access to his PII. If the user chooses "My Friends" or another group for which the user is himself the Source of Authority, then the user will subsequently need to assign the My Friend (or other group role) to the users who are to become the PII recipients. (See Use Case "Delegation of Authority".). Goto step 11.

10. If the user selects "Anonymous User", the user is then asked to set the **sharing and security** properties of the secret URL that will eventually be created for this PII asset. This will create a sticky policy for the secret URL. Note that in this case, since the PII recipient may remain anonymous, the policy cannot be a traditional RBAC, ABAC or DAC policy, since these rely on the user being authenticated.

    10.1. The user can select from a range of security options for the particular PII asset. These may include:

    - **Number of accesses allowed:** allows the user to set the number of times the asset may be viewed by the intended PII recipient(s). Valid entries may be 1, unlimited, or any integer.

    - **Time when accesses allowed:** allows the user to set a time period during which the asset may be viewed by the intended PII recipient(s). Valid entries may be "always", or until a given date and time. More sophisticated policies may include times of day, or days of week etc.

    - **Activate embedded URLs:** allows the user to control whether or not the person viewing the asset will be allowed to dereference any URLs embedded within the asset, or whether access to these will be blocked by the repository service removing the hotlinks before the asset is returned to the PII recipient. Note that even if the user allows the embedded URLs to be activated by the PII recipient, there can be no guarantee that the latter will actually be able to view their contents, since each of these embedded PII assets/URLs will have their own sticky policies as per the current one. For example, the time period of an embedded secret URL may have expired, or the sticky policy may require the PII recipient to authenticate and prove he is a member of a specific group.

[9] These are either attribute values or combinations of attribute types and values.
[10] The list of trusted Sources of Authority is provided as part of the TAS3 Network meta data. When an organisation joins a TAS3 trust network, it must provide the Source of Authority for the roles that are allocated within its organisation.

10.2. Once the user has set his access policy for anonymous users the user next sets the privacy policy in step 11.

11. The user may then set the allowed uses/purposes to which his PII may be put, by selecting the *Restrict Usage* button. The user is presented with a set of options from which he can select any number. The purpose options listed here are taken from the P3P policy specification [9]. We may wish to amend this set later. These are "Only for current activity for which the data was provided", "Web Site Administration", "Research and Development", "Tailoring", "Pseudonymous Analysis", "Pseudonymous Decisions", "Personal Analysis", "Personal Decisions", "Web-based Marketing", "Historical Preservation", and "Telemarketing".

12. The user may then set the retention period which any PII recipient must obey, by selecting the *Retention Period* button. The user will then be presented with the following mutually exclusive options: "No Retention", "For stated usage purposes only", "No longer than" and "Indefinite". If "No longer than" is selected a calendar is displayed which allows the user to select a date by which the PII must be deleted by the PII recipient.

13. Finally the user may ask the repository to create a fixed proof URL for the asset. If the user has set a DAC or RBAC policy for a PII recipient, the page will display a *Create Proof Link to This Document* button. This will create a proof of possession URL which is tied to the user's identity. Goto 14. If the user has set a policy for an anonymous PII recipient, then the page will display a *Create Link to This Document* button and a *Create Proof Link to This Document* button. Both of these will create secret URLs for the PII subject to copy and send to the PII recipient. The latter will contain proof of possession of the PII asset by the PII subject by tying it to the user's identity (goto 14), the former will not, so end of use case.

14. The user is asked to enter the identity (free form string) that he wishes to be inserted into the fixed proof URL. The user is free to choose any identity he chooses. This will be displayed by the repository to the PII recipient when she accesses the fixed proof URL (see Use Case 8).

12.1.4 Use Case 4. PII Subject authorising a Trusted PII Recipient to be Granted Append Access to a PII Asset

In this use case, the user wishes to share access to a PII asset with a trusted PII recipient such as a referee, and enable her to annotate the contents of the document i.e. append access (see use case 9). This use case is very similar to use case 3, only this time the user wishes to give read and append access rights to the PII recipient.

1. The first steps are identical to steps 1 to 5 of Use Case 3.

2. The user selects the "Grant Annotations" button and is presented with three choices "Local User", "Group" and "Remote User".

3. If the user selects "Local User", the user is shown either a picking list of existing users at the repository, or a search screen allowing the user to search for an existing user, at the option of the repository. If the user can identify an existing user, then the user identifier is entered directly into the DAC sticky policy for this PII asset. If the user/repository is not able to identify the PII recipient, then the repository may display a message "sorry unable to identify the user" and could then migrate the user to the Remote User page described below.

4. If the user selects "Group", the user is shown a picking list of groups exactly as in step 9 of Case Study 3 and an RBAC sticky policy is created.

5. If the user selects or is migrated to the "Remote User" option, the user is then asked to enter the name of IdP and the remote user's unique identifier at the IdP[11]. A DAC sticky policy is then created for this PII asset.

6. Finally the user is allowed to create some constraints in the sticky policy that has just been created. The user is presented with the following options:

   6.1. **Annotations are erasable (T/F):** if this is set to false, then the annotations which are made by the trusted PII recipient cannot be deleted by the user, but only by the person who made the annotations.

   Note that the annotations should never be modifiable by the user, just as the asset itself should never be modifiable by the trusted PII recipient.

   6.2. **Require proof (T/F):** if true, the PII recipient will be required to create a proof link between the annotations and the PII asset. If false, an unprovable (unfixed public) link or proof link may be created at the option of the PII recipient.

12.1.5 Use Case 5. Delegating/Assigning a role (group membership) to someone

In this use case, the PII subject either delegates one of his existing roles or assigns a role for which he is the source of authority, to a PII recipient, enabling her to access a PII asset controlled by his RBAC policy. The PII subject uses a delegation web service since this has a policy to control delegations, and it is trusted by other components of a TAS^3 network.

1. The user navigates to the delegation service and is invited to login as in all previous use cases.

2. The user is presented with three choices of who to delegate a role to: "Local User", "Group" and "Remote User".

3. If the user selects Local User, he is presented with either a picking list of existing users known to the delegation service, or a search screen allowing the user to search for an existing user, at the option of the delegation service. If the user can identify an existing user, continue at step 6. If the user/service is not able to identify the delegate, then the repository may display a message "sorry unable to identify the user" and could then migrate the user to the Remote User page described below.

4. If the user selects "Group", the user is shown a picking list of groups that have been defined in the ontology for the TAS^3 network. The user can select an arbitrary set of these groups, and only users who are members of the intersected set of groups will be eligible to be delegated the new role. If the selected group memberships can be assigned by different Sources of Authority, then the user will additionally be shown a picking list of the Sources of Authority of each group he has selected. For example, if one of the selected groups was Project Managers, then he might be shown the set of organisation in the TAS^3 network who can set the PM role within their organisations, e.g. "University of Kent, KU Leuven, Synergetics, Kenteq, Risaris, etc." and the user may choose Kenteq. Then only PMs assigned by Kenteq would be members of the group eligible to be delegated the new role. Continue at step 6.

5. If the user selects or is migrated to the "Remote User" option, the user is then asked to enter the name of the IdP and the remote user's unique identifier at that IdP.

[11] The user must have some way of uniquely identifying the remote user to the repository. A globally unique identifier, or a public key could be used instead of an IdP name and local identifier.

6. Now that the potential delegate (or group of delegates) has been chosen, the user is invited to pick one of his roles that is to be delegated to the delegate(s), from a picking list displayed to the user. For example, the user might choose “My Friend”, or “WP6 member”. The user is allowed to specify the duration of the delegation, by completing *From* and *Until* fields. Helpful calendars could pop-up making it easy for the user to complete these fields. Finally the user is allowed to say if the delegate(s) may further delegate the role to someone else whilst their role is active.

7. The delegation service processes the user’s request and responds by saying either “delegation successful” or “delegation constrained” or “delegation failed”. In the middle case the delegation service will say how the user’s delegation request has been constrained, for example, if the requested duration of the delegation was longer than the delegation policy allowed or the user’s own delegated privilege lasts, then the delegation may be constrained to last for a shorter period of time.

12.1.6 Use Case 6. PII Subject viewing and correcting his PII Asset in a Repository

In this use case, the PII subject is viewing PII information about him that is held in a repository that is controlled by someone else i.e. the PII subject is not correcting data that he has created himself in a repository, since this use case is trivial and is expected to be supported by all repositories. Rather the PII subject does not have write/delete access to the repository in which his PII asset is held. There are two variants of this use case, one in which the PII subject has read and append rights to his PII asset, the other when he only has read rights. In the former case, this is identical to use case 9, in the latter case it is identical to use case 8, so neither will be described further.

12.1.7 Use Case 7. PII Subject De-Registering from a Repository

In this use case the user de-registers from a repository and deletes all his data that he has stored there. (Note this is not a use case for a PII Subject attempting to delete his PII from an authoritative source to which he does not have write access.)

1. The user navigates to the repository and is invited to login as in all previous use cases.

2. The user displays all his documents that are held in the repository, and marks them all for deletion.

3. The user goes to the My Profile page and selects all the proof links to his PII attributes in other repositories, right clicks on them and selects Delete link.

4. Finally the user selects the Remove Account button. The system responds “Are you sure” and after selecting Yes, the user is informed “A confirmation email has been sent to your registered address. When you have received it, please follow the instruction contained therein. We thank you for using RePXXX Repository Services and hope that we can be of service to you again in the future”.

5. The user logs into his email account and sees an email message from RePXXX containing a secret URL which he clicks on. This returns him to the repository site, whereupon he is displayed the message “We confirm that your account and all its data has been deleted from our system”.

12.2 PII Recipient Use Cases

12.2.1 Use Case 8. PII Recipient (read) accessing a PII asset and its embedded contents

In this use case, the PII recipient receives a URL that points to a compound PII object and uses his web browser to access it. The PII recipient is not necessarily aware whether the URL is a public URL, a secret URL or a fixed proof URL. This will become evident once the URL is de-referenced.

1. The user navigates to the repository using the provided URL. The repository inspects the URL and if it is a fixed proof public URL or public URL goto step 2. If it is a secret URL or fixed proof secret URL goto step 3.

2. If the URL is a public URL the PII recipient will have to authenticate herself to the repository. She will be invited to login as in the previous use cases. If the user is already logged in to the federation and SSO is in operation, she will not see the login page and this step will be bypassed. Goto step 4.

3. If the URL is a secret URL it means the user is probably unknown to the federation and is unable to authenticate to the repository. The repository asks the user if she would like to register as a new user of the repository. If she answers No, then continue at step 4. If she answers Yes, then goto step 3 of Use Case 1 to register as a new user, and return to step 4 after registration has been completed.

4. The repository checks the sticky policy of the compound PII document or the policy of the document at the secret URL, and if the user is authorised to read the document goto step 5. If the user is not authorised, she will see the error message “sorry you are not authorised to read this document”.

5. If this is any type of fixed proof URL the repository will say to the user “The information in this document is the personal information of <XXX>. Do you wish to continue?” where <XXX> is the ID that was provided by the PII subject when the fixed proof URL was created. It has been obtained by the repository inspecting the contents of the fixed proof URL. If permanent ID mappings are being used, the repository will also say “The local identifier of the owner is <YYY>”.

6. If the user’s computer is able to enforce sticky policies and has a TPM, goto step 7, otherwise the user is shown the privacy policy of the compound PII document and she will have to accept the terms and conditions of this policy before she is allowed to proceed.

7. The document is displayed to the user, along with any embedded URLs (providing the user is allowed to see them). When the user clicks on any of the embedded URLs she will be redirected to the appropriate asset repository which will then control access to the embedded asset in exactly the same way as the current repository service has just done. The user will need to be authenticated and authorised. This shows the benefit of using federations and SSO, since the user will not need to authenticate a second time to the other repositories, but should be granted read access straight away. If the embedded URL is a fixed proof URL, the referred to repository will return one of the following the messages

   7.1. For URL back pointers: “The information you are about to view has been directly linked from document <ZZZZ> by its owner. If you are not viewing the document from <ZZZZ> then you should be aware that this link is not the original one created

by its owner. Do you wish to continue[12]?" where <ZZZZ> is the URL extracted from the fixed proof URL and should match the URL of the referencing document.

7.2. For Permanent IDs: "The information in this document is the personal information of <XXX>. It is intended to be viewed only from within a document owned by <YYY>. Do you wish to continue?", where <XXX> is the ID of the owner of the embedded document and <YYY> is the ID of the owner of the referencing document.

8. The repository will keep an audit trail of who has accessed the compound PII object and when. This information should to be made available to the PII subject upon request.

12.2.2 Use Case 9. PII Recipient annotating a PII asset

In this use case, a PII recipient has received a request to review a PII asset, or compound PII asset, and to annotate it with her comments. The initial stages of this use case are identical to Use Case 8. Once access has been granted, the PII recipient is presented with a Wiki style of page instead of a normal "read only" web page. The referee is able to write to the page and add annotations, but is not able to delete any of the existing content.

If the PII recipient has already created her annotations as a separate document in a repository (the same or a different one), then she may wish to embed a reference to this document into the current PII asset. She will do this by selecting the TAS^3 button: *Create Proof Link to Embedded Document*. This button will produce a proof of ownership link to a pre-existing PII asset belonging to the user. When the user selects the *Create Proof Link to Embedded Document* button the following occurs.

1. The repository displays a list of trusted authoritative sources with which it has trust relationships. Note that it is not possible to create a proof link with an unknown or untrusted authoritative source. The user selects the source containing his PII asset and the repository redirects her browser to that site.
2. The user may be asked to login to the authoritative source, in a procedure similar to step 1 above, or if the repository and authoritative source both trust the same IdP that the user has already used, and SSO is in operation, the user will be automatically logged in.
3. The user navigates to her PII asset in the authoritative source, and selects the *Create Proof Link to This Document* button. The authoritative source creates the proof URL and returns it directly to the repository, for it to embed in the composite PII document. The user is then redirected back to the repository where she sees her proof link embedded in the composite PII asset.
4. The PII recipient may need to alter the access rights on her embedded document. She can do this by following Use Case 3 for her document.

If the PII asset to be reviewed contains any embedded URLs, then the PII recipient may click on them and be redirected to the asset's repository. Gaining access to this repository is then a repeat of Use Case 8 or 9, depending upon whether the PII recipient has read or append rights as determined by the sticky access control policy of the embedded asset.

[12] It should be possible to build controls into the protocol that forbid a referenced document from being retrieved from any location except the originally intended one, in which case the message is not needed and the system can enforce the fixed proof URLs without user involvement.

13 Appendix 2. Proof of Ownership Using Permanent IDs

The following diagrams depict high level data flows to show how a user can create a compound document in which the URLs provide proof of ownership that the referenced documents belong to the owner of the referencing document.

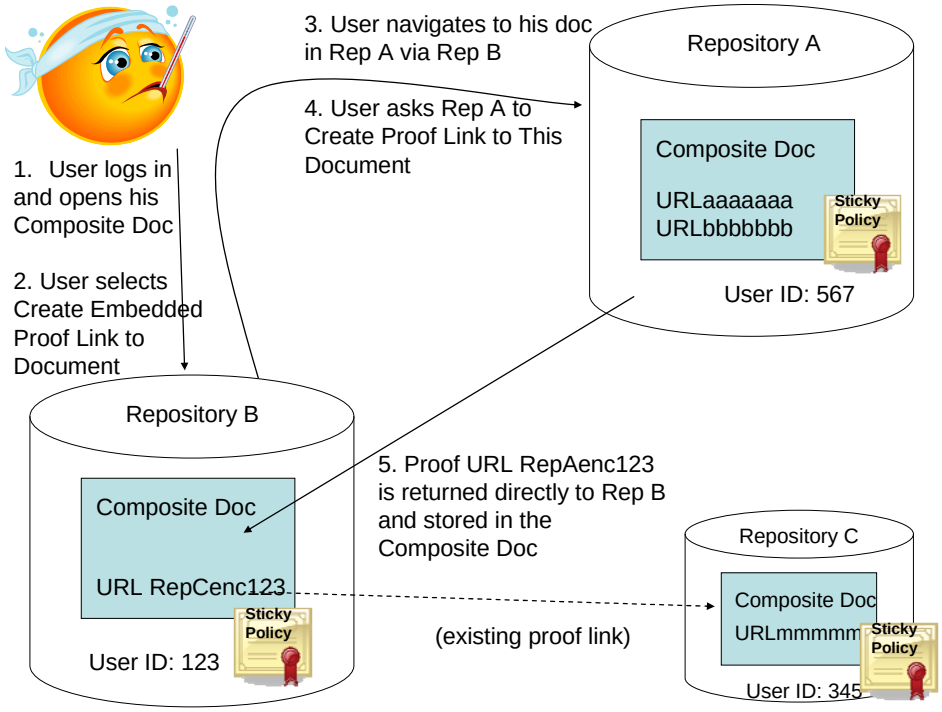

Figure 13.1 Creating a Link to an Embedded PII Asset

In figure 13.1 the user logs into his Repository B, which already contains a high level overview of his electronic medical record (EMR) stored as a Composite Document. The Composite Doc already points to a distributed record at Repository C (which might be at a hospital), and the user knows that additional distributed records also exist at Repository A (say a dental surgery). The user wishes to link the records in Repository A to his existing EMR in Repository B. The following steps are involved.

1. The user logs into Repository B, either directly or via an IdP (not shown here – see Figure 13.2). The user's login ID at Repository B is "123".
2. The user navigates to his EMR Composite Document or might be sent there immediately after login. The user has already created his sticky policy for this document. The user clicks on the "*Create Proof Link to Embedded Document*" button.
3. The Repository already has a configured list of trusted repositories and shows this to the user. The user selects Repository A and is redirected there by Repository B. The user may have to login to Repository A or SSO may be in operation. The user's login ID is "567". The user navigates to his document in Repository A.
4. The user clicks the "*Create Proof Link to This Document*" button at Repository A.
5. Repository A makes a web services call to Repository B, creates a fixed proof URL to the user's document in Repository A and passes this to Repository B. Repository B is expecting this connection and embeds the fixed proof URL into the user's document in Repository B. The user is then redirected back to his Composite Document in Repository B, and the URL (which is greyed out and cannot be edited) is visible to the user, who can then add appropriate text to the URL (such as Here are my Dental Records) for the benefit of the reader. The IDs of the user should be encrypted in the fixed proof URL, using a secret key of Repository A, in order to protect the privacy of the user. When

Repository A returns the PII asset to a recipient who uses the URL, it can decrypt the IDs and provide the identifiers of the user to the recipient.

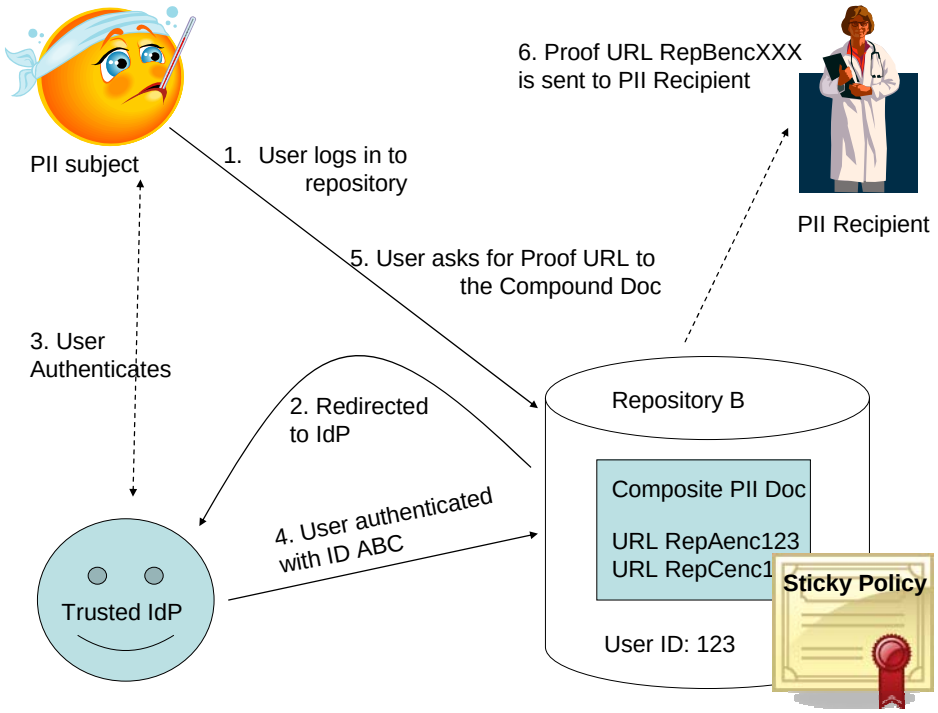

Figure 13.2 Creating a Fixed Proof Link to a Top Level PII Asset and giving it to a PII Recipient

In figure 13.2 the user is creating a fixed proof link to his Composite Document EHR in Repository B and giving it to a doctor to access.

1. The user logs into Repository B via a trusted IdP. The user is either shown a list of trusted IdPs or may be re-directed immediately to the one and only trusted IdP for this user/repository combination.
2. The user is redirected to the login page of the trusted IdP.
3. The user enters his login credentials and authenticates to the IdP.
4. The IdP redirects the user back to Repository B where he is now logged in and known as User ID "123". The user may be redirected immediately to his one and only Composite Doc or he may have to navigate to it if he has several.
5. Once the user is viewing his chosen Composite Doc he selects the "*Create Proof Link to This Document*" button. The user is asked to enter the identity that he would like to be displayed to the PII Recipient, and the URL is created.
6. The URL could be displayed for the user to cut and paste and pass to the PII recipient, or the Repository could ask for the email address of the PII recipient and could then email the fixed proof URL to her.

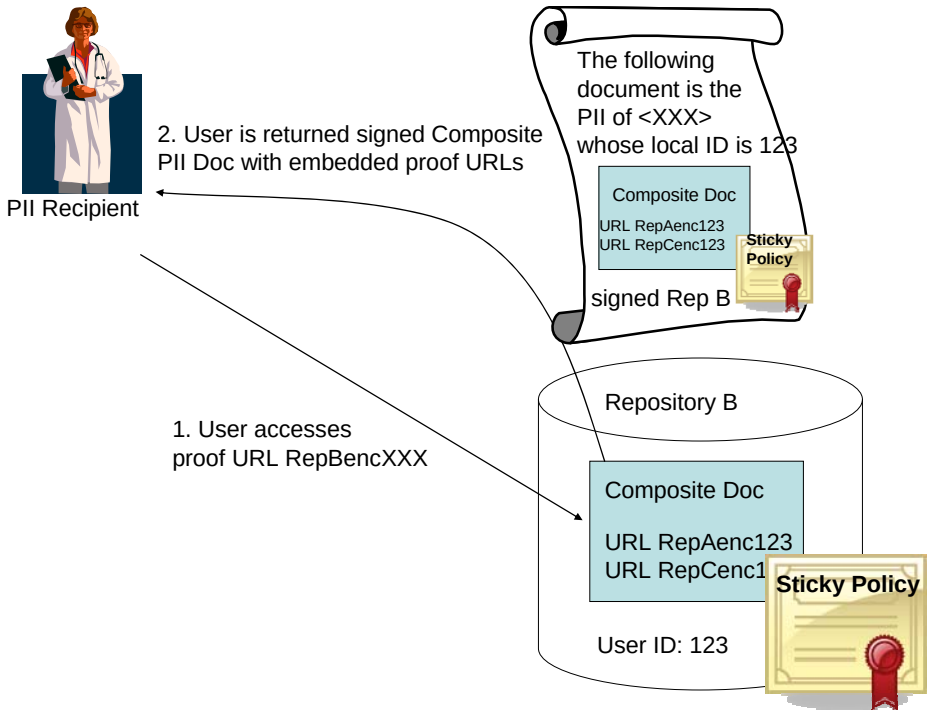

Figure 13.3 PII Recipient Accessing Composite PII Document

In Figure 13.3 the PII Recipient is accessing a compound document by using a fixed proof URL that has been given to her.

1. The user either provides a secret or public fixed proof URL. If a public fixed proof URL, she is asked to login to the Repository B either directly or via a trusted IdP, and her access rights are checked before the document is displayed. If the user provides a secret fixed proof URL she may be asked to register with the repository first (not shown), otherwise she is sent the document immediately.
2. Repository B decrypts the identifiers in the URL and informs the PII Recipient that the document is the personal information of user XXX (where XXX is the identity provided by the user when the fixed proof URL was created.). She is also told that the local identifier of the user is "123". The document is signed by Repository B.
3. The document contains embedded fixed proof URLs to Repositories A and C. The PII recipient clicks on the Repository A URL (figure 13.4). If the fixed proof URL is a public URL, the PII recipient will be asked to login to Repository A (unless SSO is in operation) after which her access rights will be checked against the sticky policy before the document will be displayed (assuming the sticky policy allows it). If the fixed proof URL is a secret URL the PII Recipient will be taken straight to the document (again providing the access rules allow it).
4. Repository A decrypts the user's mapped IDs, and tells the PII Recipient that the embedded document belongs to the user whose identifier is 123, and whose local identifier is 567. The message is signed by Repository A.
5. The PII recipient could then recursively click on URLs embedded in the Composite Document held by Repository A and be returned the chain of linked IDs which prove that all the embedded documents belong to the same original user XXX.

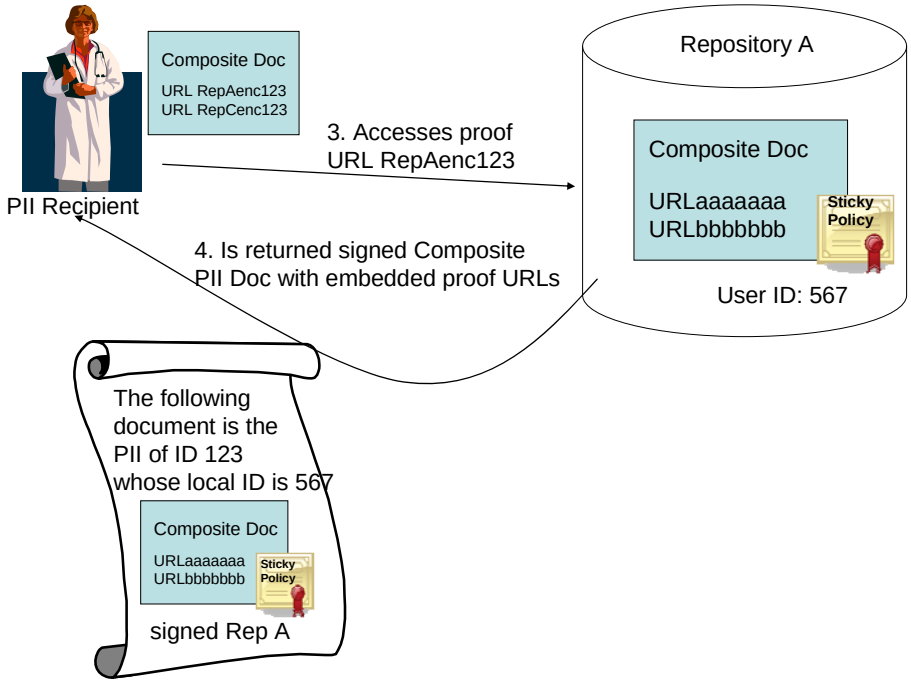

Figure 13.4 PII Recipient Accessing an Embedded PII Asset

Document Control

Amendment History

Version	Baseline	Date	Author	Description/Comments
1.0		6 APRIL 2009	David Chadwick	INITIAL VERSION FOR INTERNAL REVIEW
1.1		12 May 2009	David Chadwick	FINAL DRAFT AFTER INTERNAL REVIEW
1.2		25 May 2009	David Chadwick	Minor modifications due to discussions at VUB meeting
1.3		31 Dec 2010	Danny De Cock	Minor modifications