Analysis of Existing Web Protocols for Trusted Contents

Deliverable: D 1.1.1

Editors:	Jan Camenisch (IBM)
	James Riordan (IBM)
	Sandra Steinbrecher (TUD)
Reviewers:	Stuart Short (SAP)
	Arnold Roosendaal (TILT)

Identifier:	D1.1.1
Type:	Deliverable
Class:	Public
Date:	31 August 2008

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n^{o} 216483.

Members of the PrimeLife Consortium

1.	IBM Research GmbH	IBM	Switzerland
2.	Unabhängiges Landeszentrum für Datenschutz	ULD	Germany
3.	Technische Universität Dresden	TUD	Germany
4.	Karlstads Universitet	KAU	Sweden
5.	Università degli Studi di Milano	UNIMI	Italy
6.	Johann Wolfgang Goethe - Universität Frankfurt am Main	GUF	Germany
7.	Stichting Katholieke Universiteit Brabant	TILT	Netherlands
8.	GEIE ERCIM	W3C	France
9.	Katholieke Universiteit Leuven	K.U.Leuven	Belgium
10.	Università degli Studi di Bergamo	UNIBG	Italy
11.	Giesecke & Devrient GmbH	GD	Germany
12.	Center for Usability Research & Engineering	CURE	Austria
13.	Europäisches Microsoft Innovations Center GmbH	EMIC	Germany
14.	SAP AG	SAP	Germany
15.	Brown University	UBR	USA

Abstract

This document aims at analyzing the existing web protocol with respect to the support they offer for realizing trustworthy content on the Internet as well as what limitations they pose for such a goal. To this end, we first lay out the principles of how we think one could achieve trustworthy content. In a nutshell, solution proposes to add metadata to the content that allows users to assess to what extend the content is trustworthy. The basic idea here is that entities who process the content (generate, modify, distribute,...) bind assertions about to the content to it. These assertions together with additional knowledge about the processing entities (which essentially are again assertions made by some entity), should allow users to assess the trustworthiness. We then list the relevant existing web protocols and discuss them in context of our architecture and solution sketch. Our conclusion is that the limitations of the current web protocols are not severe but that substantial work is required to implement our architecture.

List of Contributors

Contributions from several PrimeLife partners are contained in this document. The following list presents the contributors for the chapters of this deliverable.

Chapter	Author(s)
Abstract	Jan Camenisch (IBM)
Introduction	Jan Camenisch (IBM), James Riordan (IBM), Sandra Steinbrecher (TUD)
Scenario, Architecture and Requirements	Laurent Bussard (EMIC), Jan Camenisch (IBM), Phil Janson (IBM), Benjamin Kellermann (TUD), Ronald Leenes (TILT), James Riordan (IBM), Sandra Steinbrecher (TUD), Rigo Wenning (W3C)
Gap Analysis w.r.t. Existing Web-protocols	Patrik Bichsel (IBM), Carine Bournez (W3C), Laurent Bussard (EMIC), Jan Camenisch (IBM), Phil Janson (IBM), Benjamin Kellermann (TUD), Ulrich Pinsdorf (EMIC), James Riordan (IBM), Thomas Roessler (W3C), Sandra Steinbrecher (TUD), Rigo Wenning (W3C)
Conclusion	Jan Camenisch (IBM)

This deliverable was rendered from HTML pages using Prince XML from YesLogic Pty Ltd. YesLogic has donated a license of Prince XML to W3C.

Table Of Contents

Chapter 1

Introduction

1.1 Problem Statement

Our society and economy increasingly depends on the Internet as a communication backbone. The Web has become a primary repository of information, easily found thanks to progress in search algorithms and data analysis. Moreover, Web 2.0 technologies that allow users to easily create original content and annotate existing content, are changing the dynamics of the Web: from being a space with considerably more content consumers than content producers, the Web is becoming a platform where every information consumer is potentially a producer as well. User contribution is at the core of many services available on the Web and as such, is deeply built into those service architectures. At the other end of the spectrum, websites such as Wikipedia (and in general, wikis) are entirely based on content contributed by multiple users and modifiable at any time by any of them. And, with the mashup concept, information can be combined from multiple sources and presented in novel consolidated forms.

Individuals and organizations increasingly depend on this distributed information, but they face severe trust limitations. In the Web 1.0, it was already difficult to decide if online sources could be as trusted as off-line sources such as books. With Web 2.0 the question of trust in online content becomes even more important - users have increasing possibilities on the Internet not only to consume but also to publish information. In general, users cannot rely on the information correctness. They cannot be sure that the information will be accessible in the future, whether it is legal to use it, and who would assume liability in case the information is incorrect. Users of the Web are not protected against lies and misinformation aimed at destroying reputations - think of the recent cases of intentionally false articles in Wikipedia (e.g., BBCNews [BBCNews]), or stock price manipulations through misleading newsgroup postings (e.g., CNet [CNet]).

In fact, with the highly dynamic information flow enabled by the Web, information often takes a life of its own as it can be, for example, published, edited, copied, aggregated, or syndicated; it eventually becomes detached from the context in which it was created and evolves separately. Users do not have cues to determine whether they can trust the information or not. Personal ad-hoc strategies to deal with this, such as trusting only certain websites, are no longer appropriate when the information is dynamically assembled from

multiple sources or is passed around, republished, or when the website itself does not perform editorial tasks but entirely relies on its users.

This deliverable is an analysis of web authentication protocols from the standpoint of trusted content. Web authentication protocols concern the establishment of channels between two parties which allow at least one of the parties to form justifiable beliefs about properties of the other party. Such properties include identity, membership in a group, or possession of a credential.

Trust in content is typically derived through trust in people, institutions, qualifications or processes. A channel providing basic integrity mechanisms provides a weak binding between the content delivered and the receiving party, hence a weak binding to a potentially trusted party. The latter may be leveraged, to derive beliefs, hence trust, about the content delivered over the channel.

In order to provide the context for the analysis of web authentication protocols, we begin with a clarification of what we mean by trust and trusted content.

1.2 Trust, Trusted, and Trustworthy

"Trust" is a difficult quality to define precisely due to the fact that there are often implicit qualifiers which are determined by the context in which the word is used and that the word is used to mean many different things (even within a single context).

An example of implicit qualifiers is in the statement "I trust my doctor" with the context of an upcoming knee operation: this might actually mean that I believe that my doctor will act in competent accordance with current medical practice in matters concerning my health. By contrast, the statement "I trust my teenage son" with the context of leaving him alone for the weekend, might mean that I believe that he will act responsibly with respect to his own health and well-being in addition to the assets left in his care. In each case, I am trusting somebody to behave correctly but the actual actions or behaviors determined by "correctly" are completely different. Confusion and ambiguity often arise from the fact that the behaviors are determined from context rather than explicitly specified.

We note further that these two examples illustrate the most common difference in the usage of the words "trust" and "belief": negative consequences for the person trusting if the belief proves false.

Definition is further hindered by the gap between "trusted" and "trustworthy". Trusted naturally means that one party trusts another party or entity (with implicit qualifiers). Trustworthy, by contrast, means that one party or entity satisfies a set of conditions (with implicit qualifiers) so as to justify well-founded trust on the part of another party. There is finally the issue of whether trustworthiness can be evaluated. Trusted does not imply trustworthy. Trustworthy does not imply trusted. The ability to evaluate trustworthiness varies with environment.

To ensure clarity we provide an example with bridges.

- A solid and well built bridge that simply is unknown to a particular party is trustworthy but not trusted (as it is entirely unknown to the party).
- A bridge on the verge of collapse which is crossed by a particular party is trusted (e.g. the party believes it is safe and relies upon this belief) but not trustworthy.

- A bridge whose structural members cannot be inspected due to its design, for example if load bearing members are inaccessible due to a facade, might be trustworthy or might not be; we have no way of evaluating the trustworthiness.

A good discussion of trust is available at the Stanford Encyclopedia of Philosophy [Trust].

1.3 Trust in Content

As with trust in a person, institution, or process, trust in content has implicit qualifiers. Generally speaking trust in content means belief that the information is true, accurate or at least as good as possible and reliance upon this belief. The realm of trust, analogous to trusting a doctor in medical matters, is generally determined by the topic of the information itself.

As before, content may be trusted or trustworthy without implication in either direction. The ability to assess the trustworthiness of content varies. A particularly nice example is provided by trust in a mathematical statement. It may be:

- derived wholly from a proof (assuming sufficient skill on the part of the proof consumer),
- derived from people ("I know X and X said there is a valid proof"),
- derived from qualifications ("These three Fields medalists said there is a valid proof"),
- unknown (there is no known proof),
- or even unknowable.

The point of "trust in content" is enabling consumers to assess (correctly) the trustworthiness of content. Such enabling involves a combination of technical mechanisms, psychological insights, and user education.

1.4 Enabling Trusted Content

Trust in content is most often derived from trust in a person, entity, or process. As such, there needs to be a binding between content and the person, entity, or process from which trust may be derived.

There are two standard approaches to address this binding. The first, more commonly used, consists of trusting the deliverer of the content. These include online news sites, corporate Web sites, and main entries in a blog. The second approach is to include meta-information along with the content that the user may use the assess properties of the content. Such meta-information includes digital signatures on the content itself and the links to external authoritative sources that are encouraged in Wikipedia (note that the external links tend to follow the first approach).

The former approach offers the advantage that it is rather easy although it has the disadvantage of numerous security problems; trusting a deliverer of content, such as a Web server, means trusting an entire trust chain including the administrators of the system (trust for both non-maliciousness and competence), the developers of the system, the transport mechanism, larger-scale infrastructure providers such as certificate authorities and DNS (Dynamic Name Service) providers, and so forth.

The latter approach offers the advantage that trust mechanisms may be built as a network so to eliminate single points of failure. Some, such as digital signatures, have the advantage of binding directly to the serialization of the content (cf the meaning or presentation to the user of the content) hence ensuring integrity. Generally the disadvantages are cost and significant complexity (both technological and social).

A deliverer-based approach requires infrastructure that is both trustworthy and trusted. Activity 6 focuses on the infrastructural basis for trusted content. Specifically, it investigates technologies such as, but not limited to, trusted computing and smart cards. The infrastructural components covered by this can, in the context of trusted content, be applied to authenticating both users and content, storing and securely handling both identity information, meta-information, and ultimately arbitrary content that needs to be treated with special care.

One of the most significant problems in delivering trusted content to end users, is the lack of trustworthy platform on which to interact with the content. If the end platform is not trustworthy, then there is no reason to believe that the user is produced, and perhaps signed, content is the same as the which the users believes it produced. The potential of trusted mobile devices for authentication is also within the scope of Activity 6. As the trusted platforms investigated are more flexible and powerful than conventional smart cards, growing beyond simply handling authentication and authorization based on identity information, it becomes feasible to treat trusted environments as full-fledged platforms for identity management, rather than just authentication platforms. Also, trusted environments become more and more widespread within mobile devices such as mobile phones, as several solutions for realizing more secure SIMs have emerged. This makes it possible to harden mobile phones, a class of device that has a very large user base, to a point where they can serve as a trusted platform for identity management and content-based transactions.

In addition, on the service side, the application of the trusted infrastructure to the enforcement of privacy constraints is investigated, with a special focus on service composition. The technology examined here includes the combination of privacy policies for controlling the handling of and access to content at and by the individual services. One example is, once again, identity management, where the infrastructure could enable a trusted distributed personal rights management, giving the user strong control over the handling of his personal information by service providers.

Section 2 discusses trust in content in the larger context including binding via the addition of meta-information.

Chapter 2

Scenario, Architecture and Requirements

2.1 Examples and Characteristics

This section describes a few example scenarios and discusses the requirements for trusted content for them.

2.1.1 Trusted Content & Business Workflows

In this section we consider the case where a document is not hosted on a single server, but travels from place to place. Still, we want to enable a reader or user of the document to judge to what extent the (contents of the) document is trustworthy and how the information can be used. The problems that appear in such scenarios are slightly different from scenarios where the document is hosted on a single server which takes care of the document modifications.

To illustrate this, let us consider the following example of a work flow of a patient's record in a hospital. The record basically travels with the patient. Each doctor or nurse should be able to read the record (or at least to read the parts of it that they are allowed to read) and to modify the record, e.g., to add the results of a test or their analysis to the record. Also, as tests are performed, the record or rather parts of the record may be sent to other parties (possibly in parallel) who perform additional tests and interpret the results. For instance, a blood sample is taken and sent for analysis to various labs. The results, and possible interpretations need to be added back (or merged) into the main record. Another example could be that an x-ray is taken and, for a diagnosis, needs to be sent to an off-site specialist. For this diagnosis, the specialist might require further data about the patient, but naturally, only the relevant data out of the record should be sent to this specialist.

In summary, we derive the following requirements:

- Accountability of, and information about, actors: all changes to the documents should be recorded so that the actors can be held accountable for the changes they performed.
- Privacy for actors: Users (e.g., readers and editors) of document might only be able to see a partial profile of previous actors, e.g., a nurse might only see that the x-ray diagnosis was done by a doctor with required expertise, but not the name, while an investigator might be able to see the identity of the doctor. Thus, the system should also record which part or version of the document an actor has seen (e.g., only the x-ray and not the blood test results) when making a diagnosis.
- Use policies: These policies shall define who is allowed to access, use, or modify which parts of the document and for what purposes. For instance, a policy might say that the x-ray diagnosis might only be read or used if the results of an additional blood test has a certain outcome.
- Splitting and merging of documents: In work flows it will often happen that documents are split and that later different versions of the documents need to be merged again. We require that these operations can be performed such that the actor information and use policies are respected and kept intact. We further requires that if two documents are merged, then it is ensured that the right documents are merged (e.g., it should not be possible to merge the x-ray diagnosis for a patient with another patients record). In some cases, we require that if a document is split into two non-overlapping parts, then one shall not be able to tell whether or not these parts stem from the same or different documents. This latter requirement might not be compatible with the previous one - but one could imagine that only some designated parties could figure out whether or not two documents are related (and thus that they can be merged).

We note that many of these requirements are easily satisfiable if the document is hosted by a trusted third party. Our goal, however, is that no such trusted party is used but rather that these requirements be met "by the documents themselves."

2.1.2 The Web 1.0

Web 1.0 retroactively refers to the Web before Web 2.0, i.e., when there were mainly static web pages. We call Web 2.0 the recent set of web technologies where content publishing is simplified, content is more dynamic, and users can easily collaborate, e.g., Blogs, Wikis. Although, there is no clear border line between Web 1.0 and Web 2.0, a few of the issues for trusted content already arise in the typical Web 1.0 scenarios. We discuss these.

In Web 1.0, there are two different situations regarding trusted content in corporate scenarios: 1) Intranet and internal publishing/consumption of content; 2) Public Website and external consumption of content.

In the intranet case, trusted content is hosted by a corporate service, which is trusted by all parties. This service is in charge of authenticating content authors and keeping track of their identity, e.g., by displaying the name of the author of each document. Since anonymity is not expected in such a context, providing the identity of content author is not an issue. Since the Intranet is a closed world, the identity of the author is generally sufficient to decide whether content is trusted. Additional information on the author, e.g., role, is generally available within the Intranet.

Content published to public Websites is generally authored by the corporation and not linked to a particular employee. In this case, content is evaluated based on the corporation identity and attributes using, for instance, a public key infrastructure and transport layer security.

As shown in the next section, Web 2.0 breaks this simple model by, for instance, letting external parties contribute to corporate Website content, e.g., through a blog.

2.1.3 The Web 2.0

Blog

A "blog" (a portmanteau of Web log) is a sequence of generally short articles, often called entries, published on the Web. The entries are produced by an individual or collection of individuals who are the blogs author/s and are often connected by a theme. The entries may consist of text or multimedia, as in a video blog (vlog) or a photographic blog (photoblog).

An important difference between a blog and more traditional forms of publication is the low cost of production and distribution of content. A direct consequence of this is a large base of content producers and of topics addressed. As of the writing of this document, the blog indexing and searching site technorati [Technorati] tracks 112.8 million blogs.

While this proliferation has undoubtedly resulted in an increase of the amount of content available, it has rendered difficult the evaluation of the trustworthiness of the content by the content consumer.

Wiki

A wiki is a collection of Web pages, which can be edited online by its users. The objective of a wiki is to support users in collaborative creating and editing of common shared contents. It is possible to link content and to comment on it. Wikis also provide history functionality so that it is easily possible to reset pages to former versions. While some wikis are accessible and editable without authentication, others have more or less fine-grained access control.

A problem of information published in wiki systems is trustworthiness. Especially publicly-editable wiki systems can easily be tampered with. On the other hand the advantage of public edit ability is that malicious content can be corrected by a larger set of users. To prevent misuse or vandalism, most wikis try to adopt the strategy of making damage easy to undo rather than attempting to prevent it in the first place. A widely known example for this is the history function with which one can return pages to older versions. To try to assess the trustworthiness of an editor, there are different strategies. Captchas are sometimes used to prevent automatic account creation as well as to ensure that wiki contributions are made by real (human) users as opposed to automated and presumably rogue computer systems. Additionally, to make a contribution to the the English Wikipedia, a user account has to be at least four days old, before that user can rename pages so as to prevent rogue automated programs to do so. Another example is the the Portuguese Wikipedia where a user has to make a certain number of edits to prove his trustworthiness and usefulness as an editor. The German version of Wikipedia is currently testing an extension called *Flagged Revisions* which lets trustworthy authors assign *sighted* or *certified* tags. The conditions for an author to be able to assign *sighted* tags are restrictive in days of having an active account as well as number of edits. The *certified* tag can currently not be assigned as the conditions are being worked upon.

Internet forums

Internet forums (also: bulletin boards) are Web sites, where users meet to state and discuss their opinions about topics which they are interested in and which fall into the domain the forum is dedicated to. A single forum entry of a user is called "posting". Several postings that refer to the same subtopic are grouped into a "thread", thus, an Internet forum usually consists of a number of threads. In order to write own postings, users often have to create an account for the Internet forum that requires at least the indication of a (nick)-name and a valid e-mail address. The integrity of content a user posts in Internet forums is only protected by the authentication measure the forum uses (usually username and password). Unlike in the case of wikis other users usually cannot edit a user's entries, unless they have a special role in the forum such as being a moderator or administrator. A popular example of internet forum software is phpBB [phpBB]. How trustworthy content is cannot be decided directly but some forums implement reputation systems to rate the content of a posting.

Social networks

Social networks are environments in which users engage in social interaction. Social network sites (SNS) are a sub-species of social networks that are commonly defined (Boyd & Ellison, 2007) as web-based services allowing individuals to (1) construct a public or semi-public profile within a bounded system, (2) maintain a list of other users with whom they share a connection, and (3) view and traverse their list of connections and those made by others within the system. The nature and nomenclature of these connections may vary from site to site. What sets social networks apart from other kinds of social media, such as Internet forums, is that users can articulate and visualize their social networks. The public display of connections is therefore a prominent feature of SNSs. The given definition of SNSs excludes social networks such as Second Life which are not web-based.

Profiles on SNSs usually have a public part that is visible to everyone and a closed part that is only visible to visitors who belong to a certain social circle of the profile owner. A common way of controlling access to such closed parts of the profiles is by distinguishing between friends, friends-of-friends and everyone else. Friend relations can be uni-lateral or bi-lateral. Examples of popular SNSs that fall in the category of systems described above are the US based MySpace [MySpace] and Facebook [Facebook], and Hyves [Hyves] which is popular in a number of European countries. There are also special purpose SNSs, such as a network for professionals LinkedIn [Linkedin].

In order to create a profile on a SNS an individual has to register for an account. Usually there is no identification and authentication of the individual registering for an account. Many SNSs require the user to provide complete, truthful and accurate information concerning personal details such as name and e-mail address (e.g., user agreement of Hyves [Hyves Agreement]). There is, however, no guarantee that a profile actually belongs to the person presented in the profile. The trustworthiness of information provided in a profile is therefore unknown to the visitor. Many SNSs allow other users to post comments on a profile site (microblogs) in the public part of the profile without authentication. The trustworthiness of such information is equally uncertain.

Mash-up

Mash-ups are Web applications that combine data and / or functions from more than one source into a single, integrated application. Mash-ups make use of application programming

interfaces (API) published by many service providers. Those APIs are based on Web standards and allow use of the functionality of the services without being an expert in programming. Many technologies are coming up that allow the creation of mash-ups. Examples are Yahoo! Pipes, Popfly, Openkapow, Dapper, Serena Software and many more (Look at WP6.3 contribution to deliverable D6.2.1 for more information). Those technologies differ in the way the mash-up is executed. Some of them are executed on the client side in the browser while others run on a server of the technology provider.

Server-side mash-ups offer advanced features for graphically combining services but result in standard Web 1.0 pages as seen by end-users. Client-side mash-ups require more advanced features in the browser since the mash-up is a downloaded piece of code (JavaScript, Java applets, or .NET code with Silverlight 2.0). There is no well defined taxonomy for mash-ups and they are often classified on their complexity: *presentation/consumer mash-ups* bring information from more than one source into a common user interface (UI) ; *data mash-ups* extract data from multiple sources and combine it; *logic/business mash-ups* combine data, resources and processes.

There is a clear functional overlap between business mash-ups and traditional workflow applications.

However, from the standpoint of trusting content, there is a major difference: Classical workflow applications are assembled by professional IT organizations under the umbrella of some corporate "cathedral" rules and funding. Thus they usually provide some degree of trustworthiness. By contrast mash-ups, because they are so easy to assemble, can be the result of some bazaar-like, spontaneous, unsupervised, unfunded, unauthorized - even illegal - skunk work. They can combine internal as well as external processes into some novel internal or even external service offering. They pose a serious challenge to corporate IT and audit departments because they may sprout up in the wild, anywhere, at any time. As such they not only expose the company unknowingly operating them to potential immediate legal troubles as they may happen to infringe some external regulation; in addition they may even challenging that company over the longer run if some of its departments - or even clients or partners - come to rely on them while there is no committed strategy to ever maintain them, and even less to guarantee the trustworthiness of their output.

2.1.4 Abstraction

In all examples given for trusted content the following actions/actors were relevant:

- Producing Information: There is an entity producing some information, usually the author of the information.
- Processing and providing information: There is some system (probably controlled by other entities than the author) processing the information that authors produced and providing it to consumers.
- Consuming Information: There are some other entities who want to consume the information processed by the system and produced by the authors.

All the entities have to take measures to ensure that the content they produce, process, provide or consume is trustworthy.

2.2 Integrity of Content

Integrity, when juxtaposed with the word "information", generally addresses a process affecting the representation (of the information) rather that the information itself. One speaks of the integrity of a channel through which information is delivered or the integrity of a data store.

At the level of the representation of information, there are many cryptographic constructions that offer strong integrity guarantees.

Unfortunately, the passage from guaranteeing integrity of byte sequences representing information to guaranteeing the integrity of human-consumable forms of information, such as a picture or text annotations, is more problematic.

2.2.1 Binding Contents to Actors

In this section we consider the requirements for binding contents to actors by clarifying what we mean by content and actors. With this clarification, the actual binding mechanisms are discussed in linkability of information to actors (see chapter 2).

As discussed, trust in content is generally derived from trust in a person, entity or process. This generally results from the trusted person, entity, or process (henceforth actor) producing an annotation referencing some (external) content. The actor might make an annotation stating "I authored this", "Bob authored this", "I agree with this", "Following this instruction will likely result in mortal injury", etc.. In this setting the annotation (meta-information) is the content and the external content is just a reference.

In a physical world, references are somewhat more stable than on the web. It is unlikely that accolades for Kant's "Critique of Pure Reason" could be widely exploited by substituting Kant's text with that of an attackers choosing. By contrast the volatility of references on the web is very problematic.

Reference to Content

One of the fundamental building blocks of the web is that of a Uniform Resource Identifier (URI) and the related and widely confused URI type called URL (URI offering location), URN (URI that is merely a name), and URC (URI that is a citation). Clarification of these types is available at http://www.w3.org/Addressing/ [W3C Addressing]

These identifiers are fundamentally references rather than the content to which they refer.

References are generally problematic for purposes of security: if the referenced content changes while the references remain fixed, then the trustworthiness of statements containing the references is at best dubious (and can simply be meaningless).

As such, the question arises how a reference to content can be made securely.

The simplest solution is to use a cryptographic hash of the content and then use the resulting hash as the reference. This approach has several limitations:

- The reference needs to support the operations we want to perform upon the content, such as selecting a subset. For example, if we wish to refer to an entire book, we may wish our reference to apply to each chapter as well as to the whole book; hence, for example, an endorsement of the book would become an endorsement of each chapter.
- If the content gets modified (e.g., a chapter gets added) one needs to be able to (re-)generate the reference which is not possible to do so efficiently from just the hash value.
- Given the reference, one needs to be able to retrieve the content and display it to the user.
- Content often has links to other content. Sometimes this linked content is part of the original content, as often is the case with images, while other times it is not.
- The same content might change in one way or another as it is loaded at different times or from different locations (page counters, language matching, banner ads and so forth).

These possibilities, while practically reasonable, offer significant security challenges.

There are additional problems with trusted displays but this is a orthogonal issue. For our purposes assume that the user has access to at least a platform such as a laptop or mobile phone he or she trusts to behave according to a certain specification.

Actors

The actor needs to be referred to by some kind of identifier so that properties such as "expert on topic X" of the actor can be established. Requirements for such identifiers might come from, e.g., reputation systems for which it is probably necessary that different contributions can be linked to the same actor. We note that our use of actor, does not imply the ability to resolve the identity of the actor. Actors may be resolvable to individuals but may also be credential bearing pseudonyms or even automated mechanisms wherein there is no identity (e.g., Google page rank).

2.2.2 Trust Derivation Means

When a user trusts some content or the author of some content, several models can apply:

- A user may trust the author directly in the context of the content; social sciences call this **inter-personal context-specific trust**.
- A user may not know the author directly but through a social network where an information flow exists about the author or the content. This information flow is propagated, either authorized and authenticated by the source of the information flow or in the form of gossip. For this reason, all connections within a social network, through which the information flows, have to be weighted by the one who receives the information flow. There exist numerous trust models that show how this weighting can be done, as outlined below.
- **System or context trust** may be generated by the system used for the content, the social infrastructure it is embedded in, and a legal infrastructure around it. Trust can especially be generated by the providers establishing the infrastructure that provides the content. This means they have to guarantee to their users that only authors with a certain qualification are allowed to publish content within their system. This type of trust in a service or resource provider is also called **provision trust**. For the Liberty Alliance project the term **business trust** is used to describe mutual trust between

companies emerging from contract agreements that regulate interactions between them.
- Specification of the exact reason why we choose to trust a party is intractable in the general case; indeed, it is often seemingly impossible to specify why a trust decision was made in individual cases. Trust policies that can be precisely specified tend to concern simple problems in a limited and well-specified domain.

2.2.3 Attributes about Actors

Statements about actors are often referred to as attributes but this is a matter of language usage rather than meaning.

Assessing whether one can justifiably believe in, trust, or rely on an statement made by an actor often involves believed attributes about the actor. Generally, such attributes can be expressed as statements about the actor: the actor is an expert in a given field, the actor has a particular qualification, the actor is a known and respected institution, the actor is a scoundrel or a fool, et cetera. Generation of belief in such statements is recursively the same process. Trying to formalize the practicalities this process, the assessment is a computation on a large (inconsistently) labeled, self-referential, directed poorly-specified semi-graph. Such computations are, in general, intractable.

As such we begin with certain actors about whom we have beliefs (this is a good university that tends to grant valid degree, this degree is meaningful, this person has demonstrated a certain behavior, this process is valid,...) and iteratively assess statements.

2.2.4 Data Handling

One aspect of trusting information pertains to what a consumer or user of information allowed to do with the information (what rights are given to the user/consumer). This includes the following.

- **Access** The most basic right that can be given is probably whether or not a user is allowed to read the information.
- **Store** Next, a user may or may not be given the right to store the information (or only for a limited time).
- **Use** The user might be given the right to reuse or modify the information, i.e., to quote (or reproduce) it, e.g., by including it in a text of her own.

The latter right is of course the most complex one when it comes to consequences of use. Questions that need to be addressed here include the following ones:

- Can the document be modified? This includes, e.g., what parts are allowed to be modified by whom and how.
- What part of the information can be quoted (only the whole text, only a given paragraph, etc)?
- If only part of the information is quoted, what trustworthiness of that part can be derived (e.g., do all assertion carry over and to what extent do they still hold)? Moreover, it has to be determined how the assertions carry over (e.g., if only a part is quoted, then an actor might not want to give her identity anymore, but only show that she was qualified to make a certain assertion).
- What rights are given to the user of the derived work?

In conclusion, there are a number of quite complex policy questions arising here. For the scope of this document, however, it is only important to note that there are these kind of policies that should be attachable to the information.

2.2.5 Access Control

The main target of access control is to ensure the integrity, confidentiality and availability of information. Access control describes which kind of access is granted to whom. The overall access control is mostly expressed as a list of several access rules called access control policy.

An **access control policy** is fourfold and states

- what kind of access is granted,
- to which data,
- under what conditions, and
- to whom.

Obviously, rules which define the access may also include any boolean combination of these criteria.

Action (what kind of access)

Usually, two different access levels are distinguished:

- **read access**, which grants a user the most elementary right to read the data and
- **write access**, which gives a user the right to modify or delete data. Thus, there might a number of sub-cases of write to be specified.

In some access control systems, write access includes read access. Modifying information without reading them is sometimes allowed, e.g., for appending to a file. Sometimes, an access control action does not include the right to give the same access control to other persons (other objects). This is, however, typically not enforced (one could make a copy and then make that copy available to other parties); there could be different shades, e.g., just informing the user that such a right is not given or, ultimately by enforcing at the operating system (and possibly with hardware mechanisms). In some circumstance, the system may automatically allow these transfers, for example a read access to a specific person together with the data handling policy to copy and forward the data to other persons is always like the possibility to transfer the read access.

Object (which data)

The object of an access control policy is the data content itself, e.g., a specific Web site content. Furthermore one can distinguish if access is given either to

- the **data** itself, or
- to any (specific part) of the **meta data** associated such as the history or the access time to the Web site.

For example, deleting the latest version of a Web site is expressed by writing to the history information of that Web site.

Conditions

Access to a piece of information depends on different criteria including daytime, number of accesses in a certain time, the context of the access etc.

Persons

The specification of who can access content (and with which rights) can be specified as

- **Persons** through their identities, or also by
- **Credentials**, e.g., the roles or attributes that the person needs to have.

2.2.6 Level of Trusts in Server

Trust mechanisms based on web authentication protocols rather than signatures on data, typically require trust in the server to maintain the integrity of the data and to correctly attach it to meta data (e.g., attributes about the producer). We note that trust in a server implicitly requires trust in its administrators, its operating system, its configuration, and its maintenance (an insecure server is not trustworthy even if all of its official administrators are).

Trust to make appropriate information available

We trust the server in both a positive and negative sense: to provide the requested content and not furnish content that it should not. The consequences of failure in either way depend upon the particular application. Some applications require low security or privacy features but high availability (e.g., a signal system in a railroad network); undetected failure to provide information is potentially catastrophic while disclosure of information to an unauthorized party should not be particularly problematic (the trains are observable anyhow). By contrast, an audit system searching for financial fraud might have low timeliness requirements but contain private details of many parties.

Analysis of consequences of failure should determine the appropriate measures required to protect the system.

2.3 Solution and Architecture

Integrity of content requires integrity at several levels of abstraction starting with the low-level integrity of the representation of the content (byte sequences) up to the presentation of the content to the entity consuming it.

2.3.1 Low-Level Integrity of Content

This is to say that the state of the art is able to ensure integrity at the level of byte sequences. There are recent concerns about the reliability of hash functions [WYY05]. The National Institute of Standards in the United States (NIST) is attempting to address these concerns in a similar manner to the Advanced Encryption Standard (AES). Information is available at Website of NIST's Cryptographic Hash Project [NIST], although an overview of this effort is significantly outside the scope of this work.

One way to bind content, assertion by an actor, and the actor together is by means of a signature scheme. In this section we briefly discuss the requirements on the signature algorithm arising from this and to what extent these are fulfilled by existing algorithms and standards.

Application-level integrity

As a general rule, the complexity of the transformation from byte sequence to human-consumable form negatively affects the possible guarantees that can be made about security. Note that the complexity should be measured in terms of the maximal complexity that the transformation can address rather than the particular complexity of an example. For example, the evaluation of the complexity of the "simple" rendering of ASCII text in a browser should include the full complexity of HTML and its numerous variants including provisions for malformation, Unicode, XSLT, CSS, Javascript, Adobe Flash, and so forth.

Integrity of information presented to a human agent requires both the integrity of the information and the integrity of the entire process, collected components and application by which that information is presented to the user.

2.3.2 Linkability of Content to Actors

As discussed, assessing the trustworthiness of content requires one to verify whether or not (and to what extent) an actor is linked to some content, e.g., whether person X indeed endorsed a book or whether person Y is indeed the author of the book. In the latter example, the content we are considering is "I have authored this book" where "I" refers to the actor and "this book" refers to some other content. There are basically two approaches such a link can be established and later be verified (and they can of course be used at the same time). In the first one, the actor somehow states the assertion her or himself. In the second approach, the link is asserted by a third party and hence requires trust in that party (and thus is a weaker form of binding). In fact, the second approach is essentially the same as, again, there is an actor (the third party) that needs to be linked to content (in this case "X wrote this book"). Still, we are left with finding a way to link an actor to content. This can be done either on-line (e.g., using a secure authenticated channel) or off-line (using digital signatures).

On-Line Approach

The idea here is that to verify a link of an actor to some content, one establishes an authenticated and secure channel to the actor and retrieves the content from the actor. Establishing a secure and authenticated channel seems to work well currently on the Internet provided that the identity of the actor is known and can be verified. Examples include TLS and required certification mechanisms. However, if the actor needs to be provided some form of anonymity, then this approach does not work. Also, the approach assumes that the server of the actor is trustworthy, i.e., that only content either bound to the actor (or the represented actor) can be retrieved. Also, the approach assumes that the server of the actor is trustworthy, i.e., that only content that shall be bound to the represented actor can be retrieved and also that content that is bound to the actor can be retrieved. These issues regarding trustworthy servers are treated by PrimeLife Activity 6.

Off-Line Approach

In this approach, the actor employs a digital signature scheme to sign the content. This requires that:

- One can verify the actor's signature, i.e., that some identifier of the actor is bound the public key (verification key) of the used signature scheme. Typically, this is addressed by a public key infrastructure, which in general is not in place today.
- One can trust that the actor uses a platform so that only content that is to be bound to the actor gets signed. Here again the problem of presentation of the content arises.

We note that in cases where we want to provide some form of privacy, the signature should not be linked to an identifier of the actor but rather to a pseudonym or a one-time identifier. This, however, is only useful if it is possible to, later on, learn something about the context of the actor, e.g., that the actor holds a degree in engineering. Indeed, possible solutions here are anonymous credential systems which combine the binding of the actor to the content with (asserted) attributes about the actor. For instance, an actor can state that she is authored this book and sign this statement as "a person holding an engineering degree from university X". Using University's X public key, a user can then verify that statement. We refer to the next section for this.

Differences and Open Problems

The on-line approach is potentially more flexible than the off-line one. That is, the former approach allows for dynamic content easily, while in the off-line approach, special care needs to be taken if the content should be modifiable (within some defined limits possibly specified by policies). Designing a signature scheme that allows for such things is an open problem PrimeLife will investigate.

When it comes to privacy the two approaches differ in the properties they provide. As mentioned, the on-line approach can only offer very limited privacy whereas the off-line approach can offer better privacy when using the signing capabilities of anonymous credential schemes. However, when an actor is anonymous or known under a pseudonym, it must be possible to learn attributes about the actor in order to be able to assess the trustworthiness of the actor or the content linked to him or her. We refer to the next section for a discussion on how the context of an actor can be provided in a privacy friendly fashion.

2.3.3 Context about Actors

Following Dey [Dey01] "context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves".

Several actors, and each actor's context, are relevant in deriving trust from content: the producer of the content, the consumer of the content, and other parties referencing or annotating the content. The producer is influenced by the context in which the content is produced and the consumer evaluates the context of the producers and annotators of the content relative to his or her own context.

This needs context-awareness of the system providing the trusted content to its users. "A system is context-aware if it uses context to provide relevant information and/or services to

the user, where relevance depends on the user's task" Dey [Dey01]. Possible contexts of authors are their profession, the time they created the content in, etc. Possible contexts of consumers are their needs regarding the content (e.g., scientific relevance, vital importance, casual reading).

To assess the trustworthiness of content requires the knowledge of the context of the actors related to that content. Such knowledge can be established and communicated by several means. The two most prominent ones are probably credentials and reputation.

Credentials

Asserting an attribute of a user can basically be done in two ways.

- The assertion can be stated on-line, i.e., the third party (asserting party or identity provider) confirms upon request that it asserts a certain attribute value for a given actor.
- The assertion can be stated in an attribute credential, i.e., the third party signs a statement asserting a certain attribute value for a given actor.

From a privacy point of view, one of course prefers that the identity provider does not learn who is interested in knowing what attribute is valid about which actor. Certainly, when using the first approach, the identity provider always learns all this information (except possibly the requesting party's identity if anonymous communication is used). Also, depending on how an identity, pseudonym, or partial identity is linked to an assertion, all assertions made by the same actor could be linked together.

In the second approach, this information is not directly leaked to the identity provider as the attribute credential can be distributed with the identity or pseudonym of the actor. The attribute credential could also contain a signing key of the actor thereby allowing the actor to cryptographically bind the assertion she makes about a piece of information directly to her attributes. However, distributing the attribute credentials with the information and assertion made by the actor will enable one to establish whether or not two assertions where made by the same actor if one uses traditional attribute credentials such as X.509 certificates. By using so-called anonymous credentials, this can be avoided. That is, such systems allow the user to randomize her attribute certificate for use so that different uses cannot be linked to each other. Moreover, the user can decide to selectively drop attributes contained in the credentials, thus revealing only, e.g., that an assertion has been stated my a person living in given city dropping all other address details.

Reputation

To help users to estimate the quality of content, e.g. from the skills of the actors who provided the content, reputation systems have been designed and established that collect the opinions others announced about the quality of the content or the skills of its author(s). Many providers collecting information in Wikis or communities make use of reputation systems to help users estimate the information's quality.

Contents or authors earn reputation from users who rate them and thereby influence their reputation. Raters usually give subjective ratings that are influenced by their personal estimation of the usefulness of the content or the author's contributions in general. Thus, to use such a rating one must not only trust the rater's honesty but also map the rater's subjective rating to one's own view.

A reputation network is a social network that can be modeled as a graph with its nodes describing the users being part of the network (as raters, actors or consumers) and the directed links between them, representing the information flow from a rater to possible consumers. The information flow presents a subjective rating from the source of the rating. There exist numerous trust models for defining how trust in a social network might be implemented. Both the sources and the context of some rated information (including all those who created, stored, evaluated and propagated reputation) are weighted according to the trustworthiness they have for the rater of a reputation.

After its creation, reputation has to be stored somewhere. This might be either at user devices or at some central reputation servers designated for this purpose. The storage at the user devices might be either

- local to the target object that received the reputation rating or
- distributed among the devices of users who either themselves rated the target object or collected the rating from others.

The reputation stored can be evaluated by other users of the reputation system if there is an information flow in the reputation network towards them. There exist numerous metrics on which ratings can be based.

Due to the fact that most users do not only collect information from one Web site or community but make use of various sources of information on the Internet there is a need for interoperable reputation systems independent from one single provider to allow comparing information from different sources.

2.3.4 Trustworthiness of Content

Policy

One of the main mechanisms of trust derivation is through assessment of trustworthiness. As noted before, the general case of assessment is intractable. Sometimes, such policies are embedded or difficult to specify or configure (this difficulty is often intentional to prevent attacks fooling users into believing a certain content is trustworthy). Many policies found do not address the accuracy of the information but rather its authenticity, provenance, et cetera.

Here are some examples:

- We might trust our computer vendor to provide patches which are free of Trojan horses or other malicious logic.
- We might trust a particular online product review site to offer unbiased product evaluations.
- We might trust our browser to establish a channel having certain integrity and confidentiality properties if and only if displaying a little symbol of a closed padlock.
- In the previous case, we are also trusting that the certificate authority followed some process that binds the legitimate web site owner to the party seeking certification.

The last point is particularly interesting in that the evaluation of the trust hierarchy in a public key infrastructure is an example of an implemented trust policy; there are some problems as stated in the X.509 Guide [X.509 Style Guide].

Trust Heuristics

Whenever a user has to derive trust through his social network, different trust heuristics might be applied to assess the actual trust in a given content/author. Following JIB_06 [JIB07] the following trust heuristics can be distinguished. For each of these heuristics there exist numerous implementations:

- Summation or average:
  - If the recommendations in the information flow form an additive group they can be simply summed up.
  - If the recommendations in the information flow form a field the average recommendation can be calculated. The average recommendation can be calculated using certain weights, e.g. depending on the source or age of the recommendations.
  - If the recommendations are strings over an alphabet summation can be defined as the concatenation of strings.
- Bayesian Systems: In Bayesian systems the possible recommendations are binary (positive or negative). The trust values are calculated by statistical updating of beta probability density functions. To the users the trust values are represented as pairs of the number of positive and negative recommendations or/and as expectation values of the beta probability density functions.
- Discrete Trust Models: Human beings like to use discrete verbal values instead of mathematical measures for describing trustworthiness. A possible implementation considering this is given in AH_00 [AH00] with the trust values "Very Trustworthy", "Trustworthy", "Untrustworthy" and "Very Untrustworthy".
- Belief Models: Belief theory is strongly related to probability theory. The essential difference is that in belief theory for a random value the probabilities do not necessarily sum up to 1. But there is a remaining probability describing the uncertainty of the outcome. Belief theory can be used to describe the current trust value in a content/author considering recommendations given within the social network but there is still an uncertainty about the 'real' trustworthiness of the content that belief theory considers additionally to probability theory.
- Fuzzy Models: Trust an be represented as linguistically fuzzy concepts. Then fuzzy logics can be used to describe whether an author/content is trustworthy.
- Flow Models: If a reputation value is calculated as an iteration through an arbitrarily long chain of members within the social network the trust model is a flow model. The amount of reputation available within the system to assign trust to contents/ authors can be either fixed or may evolve. An example for a fixed amount of reputation is the PageRank Algorithm Google [Google] applies to rate a website.

2.3.5 Infrastructure

Trusted content relies on an identity infrastructure in order to 1) verify the origin of specific content and 2) control who can publish and consume content.

First, verifying the origin of content requires either a form of signature or trust in the service hosting content. In the latter case, the service authenticates authors of entries and provides suitable identity and/or trust information with the entry. In both cases, an authentication infrastructure is required, e.g., a public key infrastructure to check that an entry is by a medical doctor. Moreover, in numerous cases, a trust infrastructure is required as well (e.g., reputation mechanisms).

Next, restricting publication and consumption of trusted content mainly relies on authorization and authentication infrastructures. Authentication is typically based on identity infrastructure such as public key infrastructure (PKI), SAML, or security token services (STS implementing WS-Trust interface). Authorization can, for instance, be based on access control lists (ACL), policy decision points (e.g., XACML PDP), or security token services (STS).

Finally, trusted content must be protected in terms of availability and integrity. An infrastructure dealing with version management, backup, etc. is also required.

Chapter 3

Gap Analysis with respect to Existing Web-protocols

This chapter provides an analysis of the existing web-protocols with respect to trusted content. That is, on the one hand the existing web-protocols and standards are reviewed with regards to what extent they can be employed to enable users to assess the trustworthiness of web content. On the other hand, the web-protocols are also examined with respect to what limitations they pose to realize our goals, e.g., to what extent content in a blog or wiki can be secured.

To this end, we first gather, in the next section, the criteria with respect to an analysis of web protocols and standards in later sections.

3.1 Criteria

3.1.1 Binding Actors, Assertions, and Contents

As discussed in the previous chapter, there are several means (including signatures and authenticated channels) to bind content, and assertions about it, to actors. Criteria hereby include the following:

Privacy
: In some use cases, actors can be anonymous or known under pseudonyms, and only properties about them shall be known. Furthermore, different assertions by the same actors should not (necessarily) be linkable to each other and thus that same actor.

Policy Support
: We have seen that documents can be split, appended, edited, and merged. The binding needs to support these operations. Here, the split and edit operations are probably the most difficult ones to support.

Quality of Binding

A digital signature scheme seems to offer a better quality of binding than information transmitted over an authenticated channel. Thus, this criteria is about the degree of trust needed in different parties such that the binding can be assumed to be correct.

Quality of Actor Identification

This concerns the guarantees on whether the person or process, associated with an actor identifier, did indeed want to make an assertion, i.e., is the signing key of the actor protected by a smart card, or was the actor just identified by through a username and password by some server under the control of a third party.

Uniqueness of Actor ID

Assertions made about an actor need to be linked to the identifier of an actor. In this case, the identifier might be more or less unique (e.g., a three-letter UID versus a cryptographic public key).

Transferability

The extent to which a binding can be verified by a third party. Signatures are typically transferable while on-line statements are not.

3.1.2 Data criteria

Extensibility

Most languages used on the Web are markup-languages that are extensible by nature. However, some proprietary languages are closed and don't allow user-defined extensions. There are several ways to provide data extensibility: model-based or content-based. Both scenarios have similar implications on trust.

Model-based - Open Content

A lot of markup languages have an Open Content model with one or more extension points. This facilitates the binding of policies with data at the model level rather than the data source (server) level.

Content-based - Mash-ups

Mash-ups are combinations of data from different sources, possibly using different data models combined in a rather direct and primitive way, without complex model integration. Each data source may have a different level of trust, therefore rules for combinations of various trust policies are needed. The data formats usually do not include nor preclude the use of a built-in trust mechanism.

Machine processing

Some proprietary data formats make machine processing less easy for purposes like indexation, search, etc. All standard markup languages for the Web are machine processable and even human-readable/human-editable. In most cases, it is possible to process part of the content with missing information on the data structure (e.g. missing schema for an XML document).

Semantic Interoperability

Most data formats used on the Web today already achieve a good level of syntactic interoperability. XML allows one to make sense of data even if the XML Schema defining the meaning of certain elements is not provided with the content. To get to a higher degree of semantic interoperability, the data model has to be further constrained. One way described later in the document is the Semantic Web where a triple-based data model is used to ease the acquisition of new meaning into an existing language thus encouraging a much more federated approach to ad-hoc cooperation and allowing for easy combination of resources found on the Web.

Procedural Interoperability

From a security and trust standpoint, the next valuable step beyond semantic interoperability is procedural interoperability. Often security frameworks depend on a given protocol that is vulnerable to attacks. The data used in such contexts is often tied to the procedural model underlying the final consuming application. Worse, sometimes, the data model expresses the procedural assumptions underlying a certain application framework. Thus the extension and mash-up of such data sources is not trivial and needs engineering work due to the different protocols and formats used. The work on rules-languages in research may be of help to overcome barriers.

3.1.3 Publication Criteria

We review the criteria regarding the publication of content. A number of them were already discussed in the previous chapter, namely

- Privacy (see chapter 3),
- Access Control (see chapter 2),
- Integrity (see chapter 2),
- and Authentication of origin (see chapter 2).

In addition, there are a number of criteria that are not specific to the solution we have sketched in the previous section but rather hold for any means of securing content. These criteria include:

Ability to Archive
: It should be possible to archive content and retrieve older versions, for example, wiki's typically allow one access to old versions of their pages. Obviously users need to be able to also assess the trustworthiness of archived content.

Ability to Reference
: It should be possible to reference published articles whereby the reference should include a specific *version* at a specific *time*. A reference must be unique and provide integrity.

Topology
: Does one specific author writes some content (one-to-many, as usual in Web1.0 (see chapter 2) pages and Blogs (see chapter 2)) or is the content written by a set of authors (many-to-many, as in Wikis (see chapter 2))?

Dynamic vs. static content

One can distinguish between content which changes often (as in Wikis (see chapter 2)) and content which is static (as in classical Web1.0 (see chapter 2) pages or Blogs (see chapter 2)).

Reference vs. Content

Either one can store the reference to an information (typically a link to it) or the information itself or even both.

Reliability of publisher

If we believe, that specific content was written by a specific author (cp. Binding Actors and Contents (see chapter 2)), the reliability of the author may increase the trustworthiness of the article (cp. Accuracy of the Information (see chapter 2)).

Granularity

The granularity of a document may influence its trustworthiness, for example, a trust value can be computed more easily and accurately for fine-grained documents.

Authentication and Access control enforcement

Authentication and access control can be performed in different ways (e.g. technically, legal etc.).

Criteria specific to Mash-up / service composition

Service composition is an important trend in information systems, both in business workflows and for individual use. Sometimes the user of an application or a website does not notice that she is interacting with a mash-up of services. In principle SOA - and thus mash-up technology - is just an architectural pattern for combining information services. Depending on the focus of the mash-up technology the presentation layer is or is not part of the service composition.

Mash-ups are an increasingly popular way to compose services and it is hard to predict how this technology will evolve. Therefore the types of mash-ups described above just give an indication of what is being done at the moment but many unforeseen combinations and variations will need to be addressed in the future. The PrimeLife document D6.2.1 gives a definition and lists some examples of the variety of technologies applying SOA and mash-ups concepts.

In addition to the publication criteria listed in the previous section, we distinguish the three mash-up approaches by means the following criteria:

Data flow

Communication between services in separate sandboxes (e.g. iGoogle) and services in the same sandbox (e.g. Yahoo pipes)

Execution side

client-side (e.g. PopFly) vs. server-side service composition (e.g. Yahoo pipes).

Support for authentication

public, protected, or private access

Trust domain

aggregation of data from services in a single trust-domain or in multiple trust-domains

The actual comparison of mash-ups is in section Service Composition (see chapter 3).

3.1.4 Policy criteria

As we have seen, there are a large number of places where policies need to be employed to provide trustworthiness of content. Although hardly any policies are defined and used today, apart from access control policies, we list the few exceptions later in this chapter.

Composability
Policies need to support the composition of content. Different content items can be merged or only a subset of some content may need to be referred to. Machine Analyzable: In many cases, it might be sufficient for policies to be only human readable, e.g., telling a user that she is allowed to copy or cite contents. Preferably, however, users should get computer assistance in analyzing and handling policies. This includes telling a user that she is not allowed to copy a text only if the user is about to copy text or supporting the user in applying her trust derivation policies w.r.t. some content.

Human Understandable
At some point a user needs to be informed about what a policy means. This can be the case, e.g., when a user is editing her trust derivation policy or when she is assessing the trustworthiness of some content. Also, the extent to which policies can be machine processed is limited and hence at some point the user will have to do some processing by herself.

Performance
Finally, as we want to build a system, the policies must be implementable and decisions based on policies must be taken efficiently.

3.1.5 Criteria for Reputation

Nature of reputation object

Reputation can be assigned to various objects especially one can distinguish between:

- dynamic reputation objects: Objects whose reputation changes over time, depending on how these objects evolve. Authors as human beings belong to this class.
- static reputation objects: Objects whose reputation might evolve although the objects themselves do not evolve. After such an object has collected many reputation ratings, its reputation will usually converge to a certain value. A static object such as a book is a good example of this.

Objects can have different reputation natures like individual reputation for persons, group reputation for groups of persons, product reputation for products or services etc.

Designation of reputation

Reputation can be designated in two possible ways:

- Implicit reputation is linked to the name of the reputed object. Simply by using this name the object achieves recognition by users who associate its name with a certain

reputation or expectation. But this reputation is not formalized by any explicit value. Examples are famous product brands.
- Explicit Reputation is built by explicit ratings from users who try to subsume their experiences with the object. With the help of a reputation systems ratings can be aggregated to form an object's overall reputation.

Ratings can be:

- subjective, i.e. influenced by the subjective estimation of users of the object.
- objective, i.e. assigned by 'neutral' evaluators at some point in time, possibly by aggregation of subjective ratings.

An example of subjective ratings is eBay ratings while examples of objective ratings can be found in P2P systems, e.g. GNUnet (www.gnunet.org) where the reply to a query automatically leads to a positive rating, and a reply can be proven or verified at least at the time it is sent. Ratins generated by humans will of course generally be subjective.

Architecture of reputation systems

Reputation systems can be either

- stand-alone systems, possibly usable within different application.
- integrated systems primarily used and applied as part of one application, even if transferability to other applications may be possible.

After creation of reputation ratings must be stored somewhere. They may be stored locally on users' devices or on central reputation servers designated for this purpose. The local storage might be either

- on the device of the user who received the reputation.
- or on the device of other users, possibly distributed.

eBay is the typical example of central servers while GNUnet is an example of storage at the devices of other users.

A reputation can be:

- global: This means the information flow within the reputation network is complete and every evaluator gets the same reputation
- individual: This means an evaluator only gets a partial view on the available ratings.

Requirements

For securing trusted content with reputation the following requirements hold:

- **Availability of reputation:** Each user of a reputation system wants to access ratings as a functional requirement to allow him to estimate the quality of web content.
- **Integrity of web content and ratings:** Users want web content and ratings to be integer. The reputation information needs to be protected from unauthorized manipulation, both in propagation and in storage.
- **Accountability of authors and raters:** Users want authors and raters to be accountable for the web content they provide or rate.

- **Completeness of reputation:** Users want reputation to consider all ratings given. This holds especially for the storage and propagation of reputation as it should not be possible for entities involved to omit certain ratings that might be of interest to users.
- **Pseudonyms for raters and authors:** Users want to be able to rate and provide web content under a pseudonym to not necessarily allow others to link this pseudonym to their real name. In the real world there are also authors who write under a pseudonym and many services in the Internet also allow the use of pseudonyms instead of real names.
- **Non-linkability of ratings and web content:** Users want to be able to issue different ratings and provide different content without those transactions being linkable. In the absence of such guarantees behavior profiles of pseudonyms (e.g. time and frequency of website visits, valuation of and interest in specific items) could be built. If a pseudonym becomes linked to a real name, as often can happen in a reputation network, a profile would become related to a real name as well.
- **Anonymity of users:** Beneath non-linkability of pseudonyms users also want non-linkability of themselves and their actions. Here this means they want to be able to evaluate reputation anonymously to prevent others from building behavior profiles of their possible interests.
- **Confidentiality of rating:** Although a reputation system's functional requirement is to collect and provide information about an object, some raters might want only a subset of users to know which rating they actually gave to a web content, to all others it should be confidential.

As typical for security requirements some of the requirements above are conflictive. This means not all of them can be fulfilled completelsy but a compromise in the sense of multilateral security has to be made both by the system designers when designing the system and by the users if and how they use the system.

The following requirements hold for rating and reputation algorithms ENISA 07 [ENISA 07]:

- **Accuracy of the reputation algorithm:** The reputation system needs to be accurate in the calculation of ratings. This should also consider long-term performance, i.e. the metric should be designed so that temporary minor misbehavior (due, for example, to temporary failure) does not affecting a reputation significantly. Other aspects include soliciting (truthful) ratings, but also deducing hidden ones (e.g. lack of vote, observable factors). It should be possible to distinguish among a newcomer and an rater with a bad reputation.
- **Trustworthiness of raters:** Possible mitigations include making use of existing social networks, and weighting recommendations according to how trustworthy a rater is (confidence value).
- **Self-correction:** Since reputation is linked to subjectivity of its raters,

if a user disagrees with certain ratings for a certain object, he should correct both the trust values for the corresponding raters in the reputation algorithm as well as the reputation of the object. Another aspect of self-correction might be time-constraints; if the reputation object changes and ratings given a certain time ago no longer fit a rater's view of the current state of the reputation object.

3.2 Authentication & Authorization

We use the following criteria to compare different authentication and authorization protocols from a “trusted content” point of view:

- Anonymity & non-linkability
- Information available to Identity Provider
- User control

We use the following terminology in this section: the **user** wants to access a service offered by a **relying party**, e.g. online calendar. To authorize the user, the relying party may rely on an **Identity Provider** that authenticates the user and issues claims regarding this user.

3.2.1 Session Based

OpenID

OpenID allows users to log on to different web sites with a single digital identity: the OpenID identifier. OpenID is decentralized and enables users to choose the identity provider they want to rely upon. Even if users often use passwords to authenticate to the identity provider, OpenID does not impose any authentication mechanism. Different communication options are available between relying parties and identity providers.

Anonymity & Non-linkability: OpenID does not offer anonymity since the "relying party" knows the OpenID identifier of the user. This identifier combines the name of the identity provider with the user pseudonym. Since the same identifier is used with multiple services, users become more traceable than with usual username/password pairs.

Information available to identity provider: The identity provider is trusted (i.e. chosen) by the user. The identity provider knows the identity of the relying party as well as the identity of the user. The identity provider can thus trace activities of a user each time he starts a new session.

User control: With "checked_setup", the user communicates directly with the identity provider (redirection). CardSpace can be used as identity selector to improve user control. The user can select his identity provider and thus controls who he wants to trust.

WS-*

WS-Trust describes a specific type of identity provider (Security Token Service, STS) and protocols to authenticate and obtain credentials (Security Tokens). WS-Federation defines how multiple STSs can be federated. STSs can be used at client-side to issue identity claims and at service-side to issue authorization claims. In this section we only focus on the client-side STS that issues identity claims regarding a user.

Anonymity & Non-linkability: WS-Trust does not impose a specific type of credential and could be used to query anonymous credentials (e.g. Identity Mixer or U-Prove). However, in most cases, symmetric or asymmetric keys are used to ensure non-transferability of credentials. Anonymity and non-linkability are not ensured in this case.

Information available to identity provider: The identity provider knows the identity of the requester (user). Depending on the setting, the identity of the relying party is known as well. The credentials issued by a STS generally have a short lifetime and the user has to regularly contact the STS to get new credentials.

User control: Identity selectors such as Higgins or CardSpace can be used to let the user control which claims are revealed to a relying party.

OAuth

Open Authentication (OAuth) is an authorization protocol and is thus slightly different from authentication protocols such as OpenID and WS-Trust. We however try to use similar criteria.

OAuth offers a standardized way to ask for permissions. Its main targets are personal mash-ups and single-sign-on with different permissions. In other words, OAuth allows users to share resources (calendar, photos, etc.) on one site (service provider) with other sites (consumers). This form of delegation does not require that the user shares his credentials (to authenticate to service providers) with consumers. This is achieved by redirecting a user to the service provider in order to 1) authenticate the user, 2) grant access to the consumer service. The user can choose what privileges he grants to the consumer. The resulting credential can be reused by the consumer service.

Anonymity & Non-linkability: The current protocol does not support anonymity and non-linkability. Moreover, using anonymous credentials would require deep modifications of the protocol. Indeed, the current protocol requires that 1) the consumer authenticates the user, 2) the consumer gets appropriate “access token” (new one or cached) to authenticate to the service provider and access resources. Finding the right "access token" implies a form of traceability.

Information available to service provider: The consumer service, as well as the service provider, knows what the user is accessing.

User control: For each consumer, the user can control which resource he wants to give access to and the lifetime of this access right.

3.2.2 Request Based

HTTP (RFC 2616, RFC 2617)

HTTP authentication mechanism as defined in RFC 2616 “HTTP/1.1” uses a simple *WWW-Authenticate* header field with a challenge from the server to signal the need to authenticate, and an *Authorization* header field in the request with the required credentials from the client. Similarly, a proxy server would use a *Proxy-Authenticate* header field to ask the client to authenticate, and the client would include a *Proxy-Authorization* header field in the following request. Successful authentication within a *realm* is valid for subsequent requests in the same *realm*, provided that the authentication scheme is stable (i.e. credential does not vary).

RFC 2616 prevents loss of privacy or failure of access controls when using shared caches: reuse of a cached response for another request is forbidden, unless a specific cache-control directive allows to do so (*s-maxage* or *public*) or implies revalidation with new credential even if the resource has not changed (*must-revalidate*, *proxy-revalidate*).

HTTP provides several optional challenge-response authentication mechanisms which can be used by a server to challenge a client to provide authentication information. The general framework for access authentication, and the specification of “basic” and “digest”

authentication, are specified in RFC 2617, “HTTP Authentication: Basic and Digest Access Authentication”:

- In the “basic” authentication scheme, the client must authenticate with a username and a password for the specified realm. The realm value is an opaque string, unique within the server. If the server can validate the username and password for the protection space of the resource, it will serve the resource

in the response. In this basic authentication scheme, there are no optional authentication parameters and, moreover, the username and password are unencrypted.

- The “Digest Access Authentication” proposes a scheme to secure the authentication mechanism. The credential is a checksum of the username, the password, the nonce value (the challenge), the HTTP method, and the requested URI. Using the *domain* directive, the response can include resources from the server (typical use) or other servers (generally virtual hosts) in the protection space.

Anonymity & Non-linkability:

HTTP does not specify anonymous credentials mechanisms. Anonymity and non-linkability are not provided. The user is traceable within a realm or domain.

Information available to identity provider: The service provider knows what the user is accessing.

User control: The user credential is not sent to another party by the service provider, but the user can choose to reuse the credential to access another resource in the protection space.

3.2.3 TLS

The Transport Layer Security [TLS] protocol stems from the Secure Socket Layer (SSL) protocol which aims at the establishment of a secure channel between two communication partners. During the authentication process it makes use of the X.509 public key infrastructure (PKI) to establish the necessary trust relationship. In particular, a client initiating communication with a particular server (e.g. bank) wants assurance about the authenticity of the communication partner. Hence, the TLS handshake protocol requires the server to show its certificate (currently X.509) along with the negotiation of the cryptographic protocols to be used for further communication. The client checks the certificate, meaning that the trust is established using the PKI, before continuing the communication.

A major problem of the TLS trust establishment process is the fact that a majority of users are not able to judge the information about a probably untrustworthy certificate. For example, when a server presents an expired certificate many users choose to trust the server anyway because their goal is to get work done with the specific Web site so they invariably choose to trust the invalid certificate anyway.

3.3 Signatures & Certification

As we have seen one important means to bind content to actors are signatures. Here content includes basically everything that gets signed by an actor, e.g., public key certificates issued by one actor to another actor as well as just text documents. Digital signatures have been

around for quite some time, although they are not really widely used. There are several standards for digital signatures. We discuss the most important ones here.

3.3.1 XML Signature

The XML Signature specification [XMLsig] describes a format that can be used to encapsulate cryptographic signatures of arbitrary content in XML, and to sign XML documents, or subsets of such documents. The specification is designed to be highly extensible: Signed information can be subject to arbitrary transformations before signing. Algorithms - both cryptographic and otherwise - are identified by way of URIs and therefore exchangeable.

3.3.2 X.509

X.509 certificates are used to authenticate in an open infrastructure such as the Internet. Certification authorities (CA) cryptographically sign the public key of entities, which can be individuals or computer infrastructure elements. Before getting a certificate, i.e., the signature of the CA on the public key, an entity has to identify itself to the CA. The latter can also sign the public key on another certification authority which is subsequently allowed itself to sign further public keys of entities.

The certificate verification process is carried out in the following way. Upon being presented a X.509 certificate a recursive resolution of the hierarchy is taking place meaning that it starts with the CA that signed the certificate and proceeds up the chain of CAs. Theoretically, each CA that signed a certificate on a public key of another CA needs to be trusted in order to trust that the certificate is valid. Current implementations rather rely on the CAs assessing the trustworthiness of other CAs before signing their public key. Assuming that these trust relationships are transitive we can conclude that trust in the root CA is sufficient for the derivation of a trust relationship with the holder of the certificate. The trust model is clearly hierarchical.

3.3.3 Summary

The standards that we have discussed here are of rather limited usefulness to our purpose (and the same holds for the ones we did not discuss). They fulfill the basic requirements when it comes to plain signing. However, that is typically not enough, as one requires to have more context of what is signed and what the signature means. These standards were not designed for purposes of trusting Web 2.0 content. Also, non of these standards allow one to address privacy requirements.

3.4 Data Formats

3.4.1 Introduction

The ways in which content can be mixed and mashed -- while maintaining context that affects trust and policies associated with that content -- depends on at least three factors:

- the data format that holds the mash-up
- the data format of the individual pieces

- the mechanism that is used to associate trust and policy metadata with the content in question

Different requirements hold for these different pieces of the technological puzzle.

The mash-up mechanism must be able to leave both metadata and references, that link the metadata to the content they cover, invariant. This invariance possibly needs to extend beyond the meaning of the data, and include the ability of some kind of signature to survive the mash-up process.

The format of the individual pieces must be amenable to signatures and to attaching trust-related metadata.

The format that will surround the eventual mash-up needs to enable users to trust what they see about individual pieces.

What precisely these requirements mean will become clearer as we consider two very different data formats: RDF and HTML.

In this picture, RDF will be a format that enables data-based mixing and mashing (and encounters some challenges along the way). HTML is a format that we often find in both the wrapper and the content format, with some challenges concerning the trustworthiness of the resulting mash-up, and the separation of control between different domains.

3.4.2 Semantic Web

A simple data format: RDF

The Semantic Web provides a common framework that allows data and metadata to be shared and reused across application, enterprise, and community boundaries. Key ingredients of this framework include the simple, yet powerful data model of the Resource Description Framework [RDF]; a standardized query language [SPARQL]; and an ontology language [OWL] that enables machine-readable expressions of the relationships between different concepts. One of the key design principles of the Semantic Web is the use of http URIs as identifiers for arbitrary concepts. URIs can often not be coined without causing unintended conflicts: by dereferencing URIs opportunistically, additional information (e.g., additional assertions in plain RDF, or ontological information uttered in OWL) can be automatically learned about the concepts that are described by a particular resource.

For the purposes of this discussion, RDF itself is the most critical of the semantic web's data formats.

RDF's abstract syntax can be briefly described by saying that a set of assertions is modeled as a simple graph. Assertions are unordered, and RDF model theory ensures monotonicity; i.e., additional assertions cannot invalidate assertions that are already known. Monotonicity, in turn, is a key component in making RDF's open world model work: the underlying assumption is made that there might be more information out there about a particular resource than what is known at any given point of time.

RDF is therefore tailored for wide-scale automatic data aggregation that does not need to rely on detailed, application-specific knowledge of the meaning of any particular vocabulary.

While this semantic layer might seem thin, it holds promise when combined with opportunistic dereferencing of URIs.

However, the graph structure used to model data in RDF -- while a blessing for data integration -- is a curse for the trusted content problem at hand: any signature mechanism that is capable of signing abstract RDF data (as opposed to being tied to a specific transport representation, e.g., a specific RDF/XML serialization of a graph) requires the construction of a canonical serialization. It can be shown that this problem is, in the general case, equivalent to the NP-complete subgraph isomorphism problem [Subgraph Isomorphism Problem]. In Signing RDF Graphs [Signing RDF Graphs], Carroll et al. identify tractable subsets of RDF. While this research is promising, it has not yet led to standardized approaches.

The same fundamental problem applies to approaches that limit signing to specific representations of a given piece of RDF data, but then aim to identify the pieces of a larger aggregation of data that can be trusted.

An additional issue concerning semantic web trust concerns the issue of identifying and addressing data: While, once again, research on named graphs [Named Graphs] has been carried out, no standardized framework exists at this time that would enable URI-based addressing for individual RDF triples that are part of some graph. Addressability of RDF triples is, today, limited to resources on the Web that hold a specific serialization of a given graph. More fine-grained addressing is not easily possible.

Tying RDF to other formats: RDFa and GRDDL

While RDF (despite today's challenges concerning its trust framework) is a likely useful component in solving large-scale data aggregation for trusted content, the ability to combine primarily machine-targeted content (in RDF) with primarily human-targeted content (in, e.g., HTML) brings additional power. GRDDL and RDFa are two standard (in the case of RDFa, relatively far along on the way toward being standardized) technologies that serve to enable automatic extraction of RDF data from other data formats.

GRDDL [GRDDL] is, at its root, a tagging technique: HTML documents (or, in fact, arbitrary XML documents) are annotated with a small piece of information that enables a processor to identify a transform (e.g., written in XSLT) that will produce RDF/XML data from the document in question. That information is then parsed, and is assumed to be an RDF/XML representation of the information that is contained in the original document. GRDDL is, among other use cases, helpful in connecting microformats (see below) with the Semantic Web.

RDFa [RDFa] is an extension to XHTML that enables embedding of RDF into XHTML documents; RDFa's most prominent use case is the embedding of information about Creative Commons licenses in documents (see ccREL [ccREL]); a use case that is indeed closely related to the trusted content problem at hand.

In particular, RDFa is suitable for annotating XHTML documents (and parts of such documents) with information about metadata that applies to that information. This pattern could be used to link content to both signatures and trust meta-information for the purposes of the PrimeLife trusted content use case.

RDF as a Framework for Metainformation

More generally, while RDF poses a number of interesting research challenges when used as the basis for integration and aggregation of trusted data, it is readily available as a framework that can be used to model the relationship between different resources on the Web and meta-information that relates to these resources. As described above, RDFa will be a useful tool to express such linkages in the case of XHTML; similar patterns might be applicable in other contexts.

We expect to further investigate the use of RDF for these purposes.

3.4.3 HTML

The HyperText Markup Language is, together with assorted APIs (in particular the Document Object Model, DOM), CSS, and the JavaScript programming language, the framework for modern Web Applications. It is this overall environment that we consider here.

When used as an environment for aggregation of content from different sources, key requirements that HTML (or, more generally, the HTML+CSS+JavaScript environment) needs to fulfill include:

- the ability to isolate individual pieces of the overall mash-up (including the surrounding document) from each other's interference
- the ability to sign and canonicalize pieces of HTML documents
- the ability to verify signatures on pieces of HTML documents that are embedded
- the ability to create a user interaction that cannot be overridden by dynamic HTML in the content.

In this puzzle, the signature problem seems most tractable: Canonicalization of HTML content (while not easy) can be achieved by using heuristics like ***`tidy`***, possibly combined with XML canonicalization. The ability to put "id" attributes on arbitrary HTML elements enables addressing of elements; the ability to use the object tag enables embedding of almost arbitrary data within HTML content. Microformats, RDFa and GRDDL can be used to add machine-readable semantics to documents, and to make links to metadata machine-extractable while staying within the framework of HTML itself.

Issues arise mostly around the isolation of control: When different pieces of HTML are mixed and mashed into a single document, then script and style data can be embedded in a number of ways. Script data embedded with a single piece of HTML code can cause arbitrary change to the entire document. The ***`iframe`*** element (in conjunction with some recent experiments around enabling advanced "sandboxing"), but requires the individual pieces of a mash-up to be available as separate resources. Otherwise, filtering of mashed-up content to remove script and style information will be required.

Additional challenges arise when a user interface for trustworthy content is considered: In this case, the user must be able to trust the user interface's assertions about the trustworthiness of individual pieces. This problem is relatively easy to tackle when the content surrounding the mash-up can be trusted, as - in this case - it is appropriate for that service to control the overall mash-up experience. However, if that is not the case, additional thought needs to be given to the ways in which HTML (and scripting) features can subvert the communication of trustworthy mash-ups to the user.

3.5 Policy Mechanisms

As discussed in previous sections, we envision policies to steer almost any part where components interact. In practice, about the only case where policies are really used is for access control. However, there are a couple of cases where concepts related to trusted content are used today in practice and where policies occur implicitly, i.e., they are *hard coded* into the application. That is, here the policies are not dynamic, often not stated or only in human readable form, and then hard-coded into the application. The most notable examples are code signing and enforcing protocols such as https in Firefox. Besides that, there is a body of academic work that considered policies that allow one to reason about the trustworthiness.

3.5.1 Access-Control Policies

These are discussed extensively in PrimeLife Deliverable D.3.4.1 (Open Source and Standardization)

3.5.2 Hard-wired Policies

A couple of applications in practice implicitly encode trust-policies. Examples are code-signing and enforcing https in Firefox 3.

Code signing

Code signing is used to ensure that updates to the operating system, or applets run in a browser, can be trusted. The idea here is that some authority determines who is trusted to produce and code (be it a windows kernel module, a linux package, or a java applet). It is then ensured via cryptography that the system will only accept code from such sources or at least warn the user that the code does not stem from a *trusted* source. Typically, this is done by issuing a certificate on a signature public key of the code producer by this authority. Thus the code producer can ship the code together with a signature on the code attesting to the fact that the code can be trusted. The authority's signing public key is distributed to the target system of the code (e.g., embedded in the browser or the operating system when it is first shipped). Thus, when the target system receives code together with a signature on its hash value and the certificate of the code producer, it will verify the signature on the code as well as the certificate on the code producer's signing public key with respect to the authority's signing public key it has on file and it will only install/use the code if this process is successful.

The trust policy that is implicitly defined here depends on the application. The policy comprises the following.

- The tests or procedures a code producer must undergo in order to obtain the certificate by the authority.
- The tests the code producers performs on the code before signing and releasing it.
- The liability the different parties take if something fails.

For instance, getting a certificate on a signing key for the linux distribution debian only requires meeting another member of the debian community who as already obtained a key https://nm.debian.org/gpg.php [GPG].

In Windows, code signing (including certificates) enables trust decisions based on integrity of software and publisher's identity. Code signing policies let users, administrators and the operating system make consistent trust decisions.

Windows supports different code signing technologies. For instance, "authenticode" relies on CAs, certificates, and signatures to ensure integrity of executables and to verify the identity of their publisher. "Strong name signatures" are exclusively used to provide unique identities to .NET assemblies. They are not linked to certificates and need independent key exchange mechanisms.

Code signing requires an infrastructure to protect keys, the acquisition of certificates, and processes for code-signing submission and approval. However, publishers are strongly encouraged to sign code since Windows warns users when running or installing unsigned code.

Other code checking policy mechanisms

In addition to code signing, some systems have put in place an automated code checking mechanism. This mechanism automatically scans and analyzes files, particularly executables, for vulnerabilities. Packages containing detected vulnerabilities, are not installed. Examples include the portage system of the Gentoo distribution of Linux. (james provide reference).

Enforcing updates via https

Another hardwired mechanism to ensure security is a requirement that unsigned updates are delivered over a secure channel (to ensure that they are not altered in transit). An example of this is that Firefox version 3.0 has introduced the requirement that update sites for unsigned extensions (and plug-ins) must be accessed via https rather than http. An extension which does not satisfy this requirement is automatically disabled.

3.5.3 Academic work

Blaze, Feigenbaum et al. (e.g., BlFL_96 [BlFL_96]) concentrated on technical trust management in a distributed network environment.

Following their principles, a trust-management system has five basic components:

1. A language for describing `actions', which are operations with security consequences that are to be controlled by the system.
2. A mechanism for identifying `principals', which are entities that can be authorized to perform actions.
3. A language for specifying application `policies', which govern the actions that principals are authorized to perform.
4. A language for specifying `credentials', which allow principals to delegate authorization to other principals.
5. A `compliance checker', which provides a service to applications for determining how an action requested by principals should be handled, given a policy and a set of credentials.

In 1995 they developed their first formal trust management system PolicyMaker followed by its successor KeyNote in 1997 that now covers a variety of applications. Its stable public

release is described in RFC 2704 [RFC2704] and implements the components above as follows.

1. In KeyNote actions are specified as a collection of name-value pairs.
2. KeyNote uses any convenient string to represent a principal. This string can directly represent cryptographic public keys.
3. In KeyNote the same language is used for both policies and credentials. The language is human readable and writable, and compatible with a variety of storage and transmission media.
4. Trust relationships and digitally-signed credentials can be formulated in the KeyNote language. Thereby credentials in which entities (which can be identified by ????) can be granted limited authorization to perform specific kinds of trusted actions.
5. When an application using KeyNote gets an action request it submits a description of the action along with a copy of its local security policy to the KeyNote interpreter. KeyNote then "approves" or "rejects" the action according to the rules given in the application's security policy.

3.6 Reputation

3.6.1 Ebay

The members of the eBay [ebay] marketplace are allowed to sell and buy arbitrary items within the community. After an item has been sold, the seller and buyer have to exchange the item purchased and the payment in a fair way. Most items are low-price physical goods that are exchanged directly between the users. After every transaction, users may give comments and/or scores to each other; these scores are added to the users' reputations, which can then be consulted during other transactions, giving the involved peers a way to assess the risk they are taking and, overall, increasing confidence in the marketplace.

Nature of reputation object

The interaction partners in an eBay sale are dynamic reputation objects.

Designation of reputation

Reputation is built explicitly in the form of reputation profiles constructed by others' ratings.

Architecture of reputation system

eBay runs an integrated reputation system primarily used by and applied to the eBay marketplace. The reputation is stored at the central eBay server. It is global.

Requirements

- Availability of reputation depends on the availability of the eBay reputation server and the willingness of the interaction partners to rate each other.
- For integrity of web content the buyers have to trust the sellers and eBay as a marketplace. For integrity of ratings the users have to trust in each other.

- Accountability of authors is given by the way eBay verifies the registration of its members. Accountability of raters is not considered. The only way to stop unfair bad ratings as an author is to ask eBay for help.
- For completeness of reputation users have to trust eBay that their system considers all ratings given. Additionally the willingness of the interaction partners to rate each other must be given.
- Pseudonyms of raters and authors are given, but for partners in any transaction the pseudonyms become linked to a real name and address.
- Non-linkability of ratings and transactions is only possible if the auction is not listed with the rating given for it.
- Users can evaluate reputation anonymously.
- Confidentiality of single ratings is not possible, but a user can make his whole reputation profile private.

The requirements for the reputation and rating algorithm are not fulfilled.

3.6.2 OASIS Federated Reputation Management

Currently no usable public documents exist but are expected end of the year at the earliest. Either someone with more insider knowledge might outline this here already or we have to omit this section now. More information is available from the ORMS Webpage [ORMS].

3.7 Publishing Styles

This section discusses the restriction on establishing trusted content imposed by the different publishing styles.

3.7.1 Webpages

Traditional webpages are probably the easiest case of the publishing styles we consider in this section. That is, as webpages are published by a defined entity (hosting the webserver) and by virtue of the access control on the webserver, the webmaster is responsible for the content of the webpages. Thus, employing SSL (and SSL certificates), it can be assured that a given webpage indeed stems from the organisation to which the SSL certificate was issued, provided the certification authority is trustworthy. Moreover, it seems that any other way to establish the trustworthiness of the content of a webpages can be applied by embedding all this information into the webpage.

3.7.2 Blogs

As discussed in the previous section (see chapter 2), a blog is a set of immutable, individual addressable entries. Normally, there is a small set of authors who publish on a certain Website. The trustworthiness of an article depends on the author who wrote it. In addition to publishing articles, blogs can be used to leave comments on any published article. Some blogs even allow comments on comments.

Requirements

- Authentication of the author of an article.

- Possibility to write an article under a pseudonym, without losing trustworthiness, e.g. by showing a credential about trustworthiness.
- Non-traceability of an author.
- Possibility to comment on an article while preserving one's privacy.
- Possibility to comment on an article to increase trust in it, especially if there is further information about the author of the comment (e.g. his profession etc.).
- The user interface should try to increase the privacy awareness of the user to lower accidental release of personal information about oneself and others.
- Availability of articles depends mostly on the blog service provider.

3.7.3 Wikis

Like blogs (see Blogs (see chapter 3)), a wiki can be seen as a set of individually addressable entries. If one addresses a wiki page including its revision, it can be seen as immutable as well. In contrast to blogs, the trustworthiness of an article depends on more than one author though. For the set of authors of a wiki entry, the requirements stated in the blogs section apply as well.

In some wikis (e.g., mediawiki [MediaWiki], MoinMoin [MoinMoin], etc.), there is a special userpage, where a user can leave personal information, which on one hand may be a privacy problem but may also increase the trust in the user as author. However, these userpages are mostly not under any special access control, which has a serious impact on their trustworthiness. Additionally, a user can write what he wants and without having to prove anything.

3.7.4 Publishing styles for service composition

Mash-ups were briefly introduced in the section Web 2.0 (see chapter 2). For the evaluation we distinguish different types of mash-ups with respect to publishing.

- Presentation/Consumer mash-ups
- Data mash-ups
- Logic/business mash-ups (more or less equivalent to workflow)

We compare the three types under the criteria described in chapter Publication Criteria (see chapter 3). The evaluation criteria can be split in two sets. On one hand there are criteria that yield to the same results for all three types of mash-ups. On the other hand there are mash-up specific criteria that may result in different evaluations for the three types.

First we will look at the common criteria from chapter Publication Criteria (see chapter 3) and apply it on mash-ups as such. The following section shows the results of this common evaluation. In the remaining sections we will look at each of the three types of mash-ups individually and evaluate them against the mash-up specific criteria.

Common evaluation

- **Dynamic contents/static contents**: The content generated by service compositions is always dynamic. In fact the dynamic aspect is the core idea of service compositions. Usually the data items come from one or more databases and incorporate real time values, e.g. of business processes.

- **Reference vs. Contents**: Service compositions generate typically content, esp. if the used technology involves the presentation layer. However, a prominent example of reference as a result is a search engine that is offered as public service (e.g. Google, Windows Live).
- **Reliability of publisher**: The reliability of the publisher is typically high. Most services are operated by companies, since there is no market (yet) for personal services. Company services organize (parts of) their business process - hence the data is very trustworthy, since the consumer is the company itself or a business partner. Externally offered services are often catalogs, weather information, search functionality. The reliability is again high, since the publisher offers this data for business purposes. However, there are exceptions since not all companies are trustworthy. In addition there are lots of individuals posing as bogus companies who are obviously not trustworthy. This demands an authentication mechanism, which is difficult to achieve since different services may use different authentication and authorization mechanisms (see next point).
- **Authentication and Access control enforcement**: Authentication and access control are generally not addressed in mash-up technologies because they either combine publicly available services or run within a single trust domain. When authentication is supported, it is based on proprietary mechanisms and it can be difficult to mash-up two services featuring different authentication mechanisms.
- **Granularity of Documents**: Results of single services are usually very fine-grained. They retrieve and deliver information specific to the request. As services get mashed up granularity gets more and more coarse-grained, since data from various services is aggregated. However, the result is typically still fairly detailed compared to other publishing techniques.

Presentation/Consumer mash-ups

Presentation/Consumer mash-ups bring information from more than one source into a common UI, such as web portals (e.g., Live.com [Live.com], iGoogle [iGoogle], My Yahoo [My Yahoo]) displaying the information side by side. Little real integration is involved. Mechanisms such as drag and drop of pre-built widgets or RSS feeds are used.

- **Data flow**: Services communicate across separate "sandboxes"
- **Execution side**: client-side integration of data. Usually in the user's web browser
- **Support for authentication**: Authentication support is weakest of all three types of mash-ups.
- **Trust domain**: data is aggregated typically from a single provider (e.g., the host of a web portal), but composition from multi trust-domains is also possible.

Data mash-ups

Data mash-ups extract data from multiple sources and combine it. They enable cross referencing and comparison of data, e.g. mix of geographical data with WiFi hotspot locations, house prices or crime stats. Extraction of these data might be hard and require programming. Visual development environments exist to build data mash-ups (Yahoo! pipes, PopFly, Dapper, OpenKapow, Serena, etc…).

- **Data flow**: Inter-service communication
- **Execution side**: client- or server-side service composition

- **Support for authentication**: (public, protected, or private access) depends very much on the application domain and involved services. Since the combination takes considerable programming efforts they are restricted.
- **Trust domain**: aggregation happens usually across multiple trust-domains

Logic/business mash-ups

Logic/business mash-ups combine data and processes. They connect two or more applications and automate certain tasks. They always involve programming. Within the enterprise they overlap with traditional workflow applications and with composite applications but they should enable rapid customization and adaption (Serena Software). They present more overlap with SOA than the other two categories as client-side and server-side mash-ups compete with server-side orchestration technologies such as BPEL.

- **Data flow**: Inter-service communication
- **Execution side**: server-side service composition
- **Support for authentication**: private access, can support complex authentication and authorization schemes.
- **Trust domain**: aggregation of data from services in a single trust-domain

3.8 Network Layer Privacy

There are different possibilities to attain privacy on the network layer which range from simple proxies to more elaborate schemes such as onion routing or mixnets. The Onion Router (TOR [TOR]) as well as the Invisible Internet Project (I2P [I2P]) have been discussed in the PrimeLife deliverable D3.3.1/D3.4.1 [OS Projects]. The following descriptions do concentrate on the influence of those technologies on the trust relationship between two communication partners.

3.8.1 Mixnets

A mix provides several functions which enable anonymous communication. If a user makes use of one mix, the anonymity set corresponds to the number of users of the mix which is the same situation as when using a simple proxy. Apart from anonymity set, the user of a mix has the advantages that a mix usually changes packets to harden traffic analysis. For example, the sequence of the packets is changed, the size of the packets is altered and dummy traffic is inserted into the network.

3.8.2 DC-Nets

The dining cryptographers problem introduced by Chaum resulted in the proposition of DC-Nets [Cha88] which allow for information theoretical sender anonymity. In particular, DC-Nets do not require trusted third parties but to achieve receiver anonymity through message broadcast. Hence, the sender does not need to know the addressee as it broadcasts the message to be sent. This behavior is not only bandwidth consuming but also reveals that a message is sent, thus, revealing information.

3.8.3 TOR

TOR makes use of the onion routing technique where a data packet is sent through different nodes from its source to its destination. The original packet is being encrypted with the public key of all nodes in the path beginning with the so called exit node. Consequently, each node only learns its neighbors in the communication path. Hence, TOR does introduce anonymity at the network layer regardless of any privacy mechanism at the application layer.

With respect to trust establishment, TOR does not hinder authentication. In a sensible scenario, however, the authentication must be carried out in a privacy preserving way which is possible using techniques such as Identity Mixer [idemix]. In particular, Identity Mixer provides anonymity at the application layer which is not addressed by TOR. Hence, the combination leads to an effective implementation of anonymous/pseudonymous communication. Concerns with respect to revocation of anonymity and non-repudiation requirements are dealt with solely at the application layer.

3.8.4 Conclusion

For the purpose of our goal in this document, we may assume that either communication is anonymous, for both or at least for one of the two communication parties. Indeed, in order to assess the trustworthiness of content it seems preferable to assume that the communication channel is anonymous and not to rely on any identity related information that could be derived from the channel.

Chapter 4

Conclusion

Although the Internet is evolving towards Web 2.0 where content is highly dynamic and more and more contributed by users, the existing Web protocols are essentially Web 1.0. Regarding security, the main concern of Web 1.0 was to protect the channels rather than the content. Indeed, the protection of content in Web 1.0 is not achieved by Web protocols but by organizational processes by the provider of the content. Since these providers are mostly organizations such as corporations and universities, the protection of the provided content is typically the realm of the Web masters of these organizations. However, even when it comes to securing channels, the existing Web protocols suffer many limitations, ranging from technical issues to user interfaces.

As we have argued in this document, in the context of Web 2.0 it is not only required that the Web protocols allow for the protection of channels but also for the protection of the content itself. Some of the existing Web protocols and standards such as signatures and communication protocols can indeed be employed to protect content. Still, significant new work is required to enable the evaluation of trustworthiness in Web 2.0. This work ranges from developing methods to refer to content and actors, to methods of providing additional information about them (meta-information), to automated mechanisms such as trust policies to support the users in assessing the trustworthiness of content, as well as to intuitive user interfaces.

References

[AH00]
Supporting trust in virtual communities. Alfarez Abdul-Rahman and Stephen Hailes. In HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences, volume 6, page 6007, Washington, DC, USA, 2000. IEEE Computer Society.
http://portal.acm.org/citation.cfm?coll=GUIDE&dl=GUIDE&id=820322

[BBCNews]
Fake professor in Wikipedia storm, BBC News. Accessed August 22, 2008.
http://news.bbc.co.uk/2/hi/americas/6423659.stm

[BlFL 96]
Decentralized Trust Management. M. Blaze, J. Feigenbaum and J. Lacy. IEEE Conference on Security and Privacy, Oakland, CA. May 1996.
http://citeseer.ist.psu.edu/cache/papers/cs/874/
http:zSzzSzwww.cis.upenn.eduzSz~bcpiercezSzcourseszSz629zSzpaperszSzBlaze-policymaker.pdf/blaze96decentralized.pdf

[CNet]
Bounty offered for stock tipster, CNet News. Accessed August 22, 2008.
http://news.cnet.com/2100-1023-202979.html

[Cha88]
The dining cryptographers problem: unconditional sender and recipient untraceability, Chaum, 1988.
http://portal.acm.org/citation.cfm?id=54239

[Dey01]
Anind K. Dey: Understanding and Using Context. Personal Ubiquitous Comput. 5, 1 (Jan. 2001), 4-7.
http://dx.doi.org/10.1007/s007790170019

[ENISA 07]
ENISA, Position paper, Reputation-based systems: A security analysis, 2007.
http://www.enisa.europa.eu/doc/pdf/deliverables/enisa_pp_reputation_based_system.pdf

[Facebook]
Facebook Social Networking site, accessed 30 July 2008.
http://www.facebook.com/

[GPG]
Debian GPG Signing Coordination, accessed 29 July 2008.
https://nm.debian.org/gpg.php

[GRDDL]
Gleaning Resource Descriptions from Dialects of Languages, W3C Recommendation, 11 September 2007.
http://www.w3.org/TR/grddl/

[Google]
Google Search Machine, accessed 29 July 2008.
http://www.google.com/

[Hyves]
Hyves Social Networking site, accessed 30 July 2008.
http://www.hyves.nl/

[Hyves Agreement]
Hyves Terms of Use, accessed 30 July 2008.
http://www.hyves.nl/useragreement/

[I2P]
I2P Anonymous Network, accessed 29 July 2008.
http://www.i2p2.de/

[JIB07]
A survey of trust and reputation systems for online service provision. Audun Josang, Roslan Ismail, and Colin Boyd. Decision Support Systems, 43(2):618–644, 2007.
http://www.oasis-open.org/committees/download.php/28303/JIB2007-DSS-Survey.pdf

[Linkedin]
LinkedIn Social Networking site, accessed 30 July 2008.
http://www.linkedin.com/

[Live.com]
Windows Live Search. Accessed August 22, 2008.
http://Live.com

[MediaWiki]
Mediawiki, accessed 29 July 2008.
http://www.mediawiki.org/

[MoinMoin]
The MoinMoin Wiki Engine, accessed 29 July 2008.
http://moinmo.in/

[My Yahoo]
Personalized Yahoo! information and search page. Accessed August 22, 2008.
http://my.yahoo.com/

[MySpace]
MySpace music portal, accessed 30 July 2008.
http://www.myspace.com/

[NIST]
Cryptographic Hash Project, NIST. Accessed August 22, 2008.
http://csrc.nist.gov/groups/ST/hash/index.html

[Named Graphs]
Named Graphs, accessed 29 July 2008.
http://www.w3.org/2004/03/trix/

[ORMS]
OASIS Open Reputation Management Systems, accessed 29 July 2008.
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=orms

[OS Projects]
Description of Open Source Projects in the Identity Managements Space, accessed 28 July 2008.
https://trac.ercim.org/primelife/wiki/D3.4.1/OpenSourceProjects

[OWL]
OWL Web Ontology Language, W3C Recommendation, 10 February 2004.
http://www.w3.org/TR/owl-features/

[RDF]
RDF Primer, W3C Recommendation, 10 February 2004.
http://www.w3.org/TR/rdf-primer/

[RDFa]
RDFa in XHTML: Syntax and Processing, W3C Candidate Recommendation, 20 June 2008.
http://www.w3.org/TR/2008/CR-rdfa-syntax-20080620/

[RFC2704]
The KeyNote Trust-Management System Version 2, M. Blaze, J. Feigenbaum, J. Ioannidis, A. Keromytis. September 1999.
http://www.apps.ietf.org/rfc/rfc2704.html

[SPARQL]
SPARQL Query Language for RDF, W3C Recommendation, 15 January 2008.
http://www.w3.org/TR/rdf-sparql-query/

[Signing RDF Graphs]
Signing RDF Graphs, Jeremy J. Carroll, 2003.
http://www.hpl.hp.com/techreports/2003/HPL-2003-142.pdf

[Subgraph Isomorphism Problem]
Subgraph Isomorphism Problem, accessed 29 July 2008.
http://en.wikipedia.org/wiki/Subgraph_isomorphism_problem

[TLS]
The TLS Protocol (RFC 2246), Version 1.0, T. Dierks, C. Allen. January 1999.
http://www.ietf.org/rfc/rfc2246.txt

[TOR]
The Onion Router, accessed 25 July 2008.
http://www.torproject.org/

[Technorati]
Technorati, accessed 28 July 2008.
http://technorati.com/about/

[Trust]
Stanford Encyclopedia of Philosophy on Trust, first published 20 February 2006.
http://plato.stanford.edu/entries/trust/

[W3C Addressing]
W3C Naming and Addressing, accessed 30 July 2008.
http://www.w3.org/Addressing/

[WYY05]
Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu, Finding Collisions in the Full SHA-1, in Proceedings of CRYPTO 2005, pages 17-36 Lecture Notes in Computer Science, Springer Verlag.
http://www.springerlink.com/content/da906llfxwnlmlx2/

[X.509 Style Guide]
X.509 Style Guide, Peter Gutmann, October 2000.
http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt

[XMLsig]
XML Signature Syntax and Processing (Second Edition),W3C Recommendation, 10 June 2008.
http://www.w3.org/TR/xmldsig-core/

[ccREL]
Creative Commons Rights Expression Language, published 3 March 2008.
http://wiki.creativecommons.org/CcREL

[ebay]
Ebay, accessed 29 July 2008.
http://www.ebay.com/

[iGoogle]
Personalized Google information and search page. Accessed August 22, 2008.
http://www.google.com/ig

[idemix]
Identity Mixer - Pseudonymity for e-Transactions, accessed 29 July 2008.
http://www.zurich.ibm.com/security/idemix/

[phpBB]
PHP bulletin board system, accessed 30 July 2008.
http://www.phpbb.com/