Master Thesis
Computer Science
Thesis no: MCS-2005:18
August 2005

Dynamic identities for flexible access control

F. Andersson, S. Hagström

School of Engineering
Blekinge Institute of Technology
Box 520
SE - 372 25 Ronneby
Sweden

This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 2 x 20 weeks of full time studies.

Contact Information:
Author(s):
Fredrik Andersson
Address: Ronneby, Sweden
E-mail: fredrik@rsn.bth.se

Stefan Hagström
Address: Ronneby, Sweden
E-mail: stefan@iserv.se

University advisor(s):
Rune Gustavsson
School of Engineering, BTH

School of Engineering
Blekinge Institute of Technology
Box 520
SE - 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : +46 457 271 25

ABSTRACT

This thesis will analyse the pros and cons of a module-based approach versus the currently existing certificate schemes and the proposed requirements for a module-based certificate scheme to serve as a plausible identity verification system. We will present a possible model and evaluate it in respect to the existing solutions and our set of identified requirements.

Keywords: PKI, identity validation, certificate authority

Contents

Chapter 1

Introduction

1.1 Background

As long as the concept of ownership has existed, there has coexisted a need to verify the self-claimed owner. Over the years, the means of verification have evolved, but we argue that most of these means can be categorised into the fields of:

- Knowledge
- Social network
- Ownership
- Physical traits

Traditionally in electronic media, knowledge means a shared password. In social networks, the identity of unknown persons can be vouched for by already trusted. Ownership in the form of keys is still the preferred method of validation for some of our most valued possessions (i.e. keys to safe boxes, vaults, electronic secrets). The simplest form of authentication, physical recognition, is something that we tend to trust the most in our daily life despite the abundant evidence that our eyes often can be deceitful.

Today, most of these means are validated digitally, but the fields still remain. Knowledge in the form of username/ password is still the most commonly used mean of verification on the Internet. Social contacts are still used to traverse trust chains in PKI environments (i.e. PGP). Possession of tokens are also very common, as physical keys have been replaced by digital. In the last couple of years we have seen an introduction of biometrical products in the regular consumer market, allowing ordinary users to authenticate themselves to their home computers with physical traits such as fingerprints.

Although the authentication schemes have become more elaborate, identity theft is still a big international problem. Even though identity frauds have existed since the origin of the identity concept, the anonymity of the Internet has created a sense of a barrier between one's actions and the expected consequences. Basically, a person's "real" identity is separated from the virtual identity he/she assumes on the Internet. Cynically put, when the means of identification is lost, ethics and moral conscience has a tendency to fade as well. The need for better identification protocols is getting worse by the year, and predictions of a collapse of the Internet have been published[16].

Although there exist many "solutions" for validating identities, there are to our knowledge no existing solution[1] that would be suitable for a general digital identity. The focus tends to be on a cryptographic level, proving that there is none or low chance for actor C to intercept or intervene

[1] see ch.4

in the communication between actors A and B. Although essential for secure communication, they are hardly ever practical to use, and furthermore very limited as to the identity information they carry. To summarise, they solve the problem of secure communication between two parties, not the problem of identity trust.

One major problem with identities is the actual definition. What constitutes an identity? What parts of the information is sufficient for identification at each given time of query? Should the identity be mapped to a physical persona or should it be viewed from a psychological or sociological context? Even though these classic questions relate more to the areas of biology, psychology and sociology, they are still very valid questions when it comes to validating a digital identity.

More services requires automated identity checks today, and with that comes the demands for an accepted standard for digital identities. How would a standard that is sufficient for all (or at least enough) given contexts be built?

1.2 Challenges of authentication - Research question

There exist several problems with today's authentication schemes. It has been argued[13] that the most commonly used certification systems used today can be defined into three main categories:

- Directory based
- Referral based
- Collaborative based

Although the directory based X.509 standard is the one most supported by existing security solutions (PGP, openPGP, openSSL ...), we consider it to lack many of the features needed for a certificate solution for the vast majority.

The referral based PGP solution also support their own certificate standard, as defined in PKCS #12[17], but this approach is much more limited than X.509 and it still lacks flexibility.

Since these solutions are based on the concept that a chosen certificate authority (CA) signs your entire certificate, changes in your personal identity require rather much work. Even if you are your own authority (as in the case of PGP), any change requires you to create a new certificate, sign it, revoke your current key and upload the new one.

In the case of PGP, this might be acceptable, since one only tends to keep name and e-mail in the PGP certificate, but in order to have a more flexible information model, this procedure would not be feasible for everyday use. What the authors feel needed is a certificate model where different informational modules can be signed (and encrypted) by different authorities.

X.509 (and the PGP certificate model) is very statical in its nature, and even though revision 3 of X.509 enables a module based approach, it is only usable if the nature of the module based information is as statical as the rest of the certificate data.

As more services require different sorts of authentication, we are flooded with authentication tokens. What we propose is a model where the needs of the stakeholder can be met but where the authentication token(s) can be merged with your existing tokens. By letting different entities validate small, atomic parts of your identity, it would be possible to build a collection of attributes and information that could be presented as one digital persona, so that a combination of directory based and referral based authentication is used.

The magnetic card frenzy (with every store having their own card) of today is a good example of how no standard satisfy the needs of the different actors. Both in real life and on-line, the general tendency is to have a separate mean of identification for each resource one wants to access. Generally, stake holders want to keep as much information about the authorised users as possible. However, it can be argued that this kind of overly zealous collection of data is both unnecessarily

redundant and integrity invasive. Many of the stake holders require basically the same information. By letting the client present enough data from his or her digital persona, the stake holder should be able to grant the client access. With the emergence of portable cryptographic hardware tokens, such a solution would solve many of the authorisation problems one encounters in daily life as well (i.e. library card, grocery store card, door cards, etc.).

With more flexible certificates, one would also be able to encrypt sensible information in your certificate. Not all data in your identity is necessary to produce in every authentication situation, and there might be data that you want to store in your certificate that should only be readable by trusted entities. As an example, one could store information about blood type, known sicknesses and used drugs in the certificate in case of an emergency. This could be encrypted with the public key of some medical institute to ensure the integrity of this information.

We seek to define the requirements of such a system, design a feasible model and discuss its implications. What implications and/or drawbacks would such a model yield in comparison to the existing models?

In shorter terms, our research questions are:

- What is the concept of identity in our domain?
- What are the requirements for a system applying this concept?
- How could a suitable model incorporating these requirements be defined?
- How would this model work compared/contrasted to existing models/solutions?

1.3 Purpose of the thesis, Restrictions & Audience

The goal with this thesis is to acquire basic understanding of the concept of identity, and based on this design a suitable model in which one uses a minimal set of credentials for obtaining a resource at any given time.

The purpose of designing and evaluating a module based certificate standard is to facilitate in the authentication procedures that almost everyone is faced daily. Should such a system be usable, it would benefit most citizens in any modern society.

This paper will not discuss the topics of hardware solutions and cryptographic schemes further than to explain their roles in the overall system.

The targeted audience for this thesis are mainly, but not limited to:

- Digital security oriented businesses
- Security consultants
- Manufacturers of hardware security tokens
- Academical staff working the area of digital security.

1.4 Methodology

1.4.1 Preparatory work

In our preparatory work, we will mainly focus on researching existing solutions and written material regarding our topic. This will be done to build a solid base on which to acquire the system requirements.

1.4.2 Eliciting the system requirements and specifying system model design

System requirements will mainly be based on information gathered from the preparatory work. However, we assume that many requirements will be found and added during the design phase. Our main focus will be put on the validation phase, the chain of events required for the involved parties to trust each other, and the trust issues involved in a distributed certification model.

1.4.3 Evaluating the results

By comparing the proposed solution with the currently existing solutions, we will evaluate pros and cons of either alternative. We will also consider the practical implications of implementing our proposed solution as an accepted standard.

1.5 Own results

In accordance to the research questions, we will present:

- A persona based certificate model
- A Comparison matrix
- Evaluation scenarios

1.6 Guidance for the reader

In *Identities*[2], we will discuss the common problems of identities, try to define what constitutes an identity and what is required to prove one's identity. Requirements for a modular approach are identified in *Requirements*[3], and in *Existing solutions*[4], we will compare and analyse the solutions that are currently employed. The proposed solution is presented in *Proposed Model*[5], and in *Evaluation*[6] we evaluate the solution in comparison to existing solutions. Finally, we discuss the implications with the proposed model and it's possible solutions in *Discussion*[7].

[2] see ch. 2
[3] see ch. 3
[4] see ch. 4
[5] see ch. 5
[6] see ch. 6
[7] see ch. 8

Chapter 2

Identities

2.1 Introduction

To know how to validate an identity, we must first seek to define what an identity is. Since the word can be interpreted in various ways, for the purpose of this paper, we will narrow our definition a bit. In a given scenario of identity validation one may find one or several of the following actors:

- Client
- Validator
- Third party

In most cases, each party must in some parts of the sequence play both the role of the client and validator. For example, if citizen C wants to connect to his bank B, over all one can define C as the client. However, during some part of the communication, he must also verify that the party he is communicating with really is B. In this subsequence, the bank B is the client, whereas C is the validator.

2.2 Contextual identities

A major problem with identities is the actual definition. What constitutes an identity? What parts of the information are sufficient for identification at each given time of query? Should the identity be mapped to a physical persona or should it be viewed from a psychological or sociological context? Even though these classic questions relate more to the areas of biology, psychology and sociology, it is still a very valid question when it comes to validate a digital identity.

We identify four different categories of identification:

- Characteristics
  Skills, abilities, physical traits i.e biometric information, etc.
- Social context
  What social groups do I belong to? Occupation, family, religion, etc.
- Possession
  Do I possess a token that will grant me access? Keys, etc.
- Knowledge
  Do I have the knowledge to access the resource? Procedures, passwords, etc.

Characteristic-based identification has traditionally been based on recognition of physical attributes, such as photo based verification, but biometric solutions are emerging ever more rapidly. Hardware for user authentication via finger prints have hit the consumer market during the last years, and it would not be especially far fetched to believe that iris scanners are next. As DNA extraction is developed to be faster and more portable, it would not be unbelievable for future authentication via DNA. However, it is not only physical attributes that can be measured in order to identify a client. Behavioural identification methods are emerging as well. Recent papers[20],[24] have presented techniques for authenticating users based on their keystroke pattern on a keyboard or mouse movement pattern.

Certain social groups require an already trusted party to vouch for a new member, thus guaranteeing the authenticity of the new user. PGP[1] builds its' trust system on a social network authentication. By signing a public key of a user you trust, a party that trusts you may choose to trust the new public key.

Physical keys are perhaps the simplest way of anonymous authentication. Although perhaps easily associated with car or house keys, digital key-pairs in its purest form is a perfect example of a more modern approach.

Knowledge-based authentication is possibly one of the most common way to identify oneself in the digital world, where password-based authentication still is the most common way to manage users. However, knowledge-based authentication is not limited to pass phrases. In most systems, specific procedural knowledge is required for a user to validate himself.

One could argue that knowledge and possession are more or less of principle the same, but for the sake of clarity, we have chosen to separate them. It should be mentioned that these examples are only theoretical categories, meaning that in many cases, a combination of techniques are used.

Different validating entities have different requirements on identities and their coupling to physical persons. A government authority, for example, wants to connect the presented identity to a physical person, since society-resources is based on that connection. For the Swedish authorities, a person is identified by his or her civic registration number. All other identification tools (such as the passport or driver's licence) is connected to the person's civic registration number. Since this number is an artificial token based on date of birth plus a four digit control sequence making it unique, it is required for identification tokens to provide proof that this number is connected to the person presenting the token. In the case of driver's licences or passports, this is made by including a photo of the person in addition to the signature of the carrier. This signature is also required in cases where photo presentation is not possible. In later years, the ordinary signature has been complemented with a digital system. By creating a digital certificate at one's bank, one may perform citizen errands such as the tax declaration without supplying the traditional handwritten signature.

However, other parties may not require the identity to be bound to a real person. In fact, there may be cases where it is preferred for this connection to be untraceable. Unfortunately, a desire to withhold ones physical identity is often considered conspicuous. There are, however many cases in which one may legitimately want to act under a nome du plume. In these cases, other means of verification have to be used. Traditional means of validating ones rights without connection to a physical person have been possession of an artifact or possession of knowledge, i.e keys and passwords. With this kind of validation, no records can be kept in regards to whom have requested which resources, thus granting privacy as well as security. In addition to this, validation could also be made by trust bonds. In this case, an already fully trusted source vouches for a new entity, thus granting him access.

[1] see ch. 4.4

2.3 Digital identities

In digital contexts, the connection to a physical person is vague at the very best. Unfortunately, this has led to enormous problems of frauds and general bad behaviour on the Internet. In effect, more forums / resources require user registration and validation. Even though many consider it sufficient to connect an account to an email address, an unfortunate trend is to register more information about the physical person behind the account.

As discussed in the previous section, the categories of traditional authentication can be seen in digital environments as well. In personal computer systems, knowledge based pass phrase authentication is still the most common system. Public Key Infrastructure (PKI) is a system of public key encryption using certificates from Certificates Authorities to verify and authenticate the validity of each party. PKI is based on possession, although in most cases combined with a pass phrase for use of the private key.

Pretty Good Privacy (PGP)[2] implements all categories, based on a digital key, encrypted private key with a pass phrase, photographic evidence of the carriers appearance as well as the previously mentioned trust concept to accept an identity without having to verify it oneself.

2.4 Problems

Although characteristics provides passive means of authentication that are rather hard to loose, steal or fraud, there exists a certain error margin with every way of measuring analog characteristics.

Anything that requires possession risks the possibility of the token being stolen or copied. The same goes for knowledge based systems. With social groups vouching for each other, the trust is only as strong as the least trusted group member (assuming that the trust relationship is 1(trusted) to n(untrusted)).

There exist several problems with the means of validation. The most basic conflict is the balance between integrity and security. Although quite a few people seems eager to sacrifice their personal privacy/integrity in order to gain (some kind of) security. What we seek is an identification system that does not require the validating authority to store all information about a client.

2.5 Identity and Persona

In Jungian psychology, "persona" is defined as the personal facade one presents to the world[15]. The word stems from the Latin word for mask, personae. This is in essence what we seek: A possibility for the client to decide how much information to provide in any given situation. Even though the validating entity will require the client to present certain information to grant him or her access, the control of the information is always in the client's hand.

2.6 Our approach

In every validating scenario, the set of credentials presented should be just enough to validate, so that the integrity of the user is cared for. However, the security of the service provider must also be assured, and a balance between these interests must be establised. As described above, what

[2] see ch. 4.4

we seek is a way for the user to decide which credentials to provide in each situation. We will refer to this as the Persona approach.

Chapter 3

Requirements

3.1 Introduction

In this chapter, we seek to define what would be desirable in a persona based certificate system[1]. Further on, we will compare existing solutions in accordance to the sought requirements.

3.2 Flexibility of information

3.2.1 Container

Although the suggested solution should be as flexible as possible in its design, there are still some requirements that targets the entire container of data. Starting off, there must be a way to uniquely identify any presented container. In effect, some kind of authority is required to issue unique information for each created certificate. We identify a couple of problems that have to be solved for this to be a feasible solution.

- Non-exhaustible unique identifier
- Working in hostile environments
- Anonymous id retrieval

As for the identifier, it has to be a large enough sequence for anyone person and organisation in a foreseeable future to get at least one certificate. With IPv4 in mind, setting to small a scope for the identifier can lead to serious consequences. Since the Internet can be considered a rather hostile environment[16], the suggested solution should be if not protected against, at least built with threats like denial-of-service attacks and resource exhaustion in mind. No matter how large a scope the identifier has, an automated attack registering identifiers would soon exhaust the range. Since anonymity is an issue[2], this would have to be solved without integrity invasive measures. To grant this, we predict a solution with multiple certificate authorities to spread the load and possible risk. Multiple certificate authorities, however, bring questions of trust, administration and problems with revocation.

3.2.2 Data manipulation

As for the actual data, the main focus is of course flexibility. In effect, as much as possible should be granted the user and the creator of the data. However, we consider a variety of requirements to be general:

[1] see ch. 4.6.5, p. 20
[2] see ch. 3.3.2

- Optional signing
- Optional encryption
- Optional compression

Data integrity can be considered be one of the most important feature requirement of the system. For any stake holder to trust the presented module, it is required for it to be signed with a proper authority. Multiple signatures is also a valid requirement, as it is up to the verifying authority to accept one, or a combination of several signatures provided.

When it comes to sensitive data, encryption should be provided. Asymmetric encryption is more or less necessary, since public signatures is required. However, asymmetric encryption is hardly practical when it comes to streamed data, or large quantities of data. Would it be better for the proposed solution to use a combination where the actual data is encrypted symmetrically with a random generated password that is stored asymmetrically encrypted in the module?

Since size is of some importance in network enabled environments and embedded systems, optional compression of data would be a suitable requirement. Preferably, any compression should be possible to use since the data stored is unknown.

3.2.2.1 Data referencing

In order for the data to be structured, some kind of hierarchy should be allowed in order to facilitate a trust system. Data blocks should be able to reference both other containers, data blocks, or meta data. For this to work, a rule set should be defined for what actions are to be taken when the desired references can not be found or verified.

3.2.3 Revocation

Revocation is potentially one of the most costly problems with today's PKI solutions[21]. With a more flexible persona scheme, we consider flexible revocation to be essential. In addition to being able to revoke the entire persona / container, we seek a way to revoke parts of the persona that no longer holds true, or are not possible to verify. With this demand comes an even greater need for a reliable and efficient way to revoke the trust of data.

3.2.3.1 Recovery from revocation

Since the suggested solution should be of practical use, methods for recovering from revocations should exist. Should a certificate (or authority) be revoked, means of facilitating the required certificate administration is needed.

3.3 Non-functional requirements

3.3.1 Defined standards

Although our primary concern is flexibility, this kind of solution requires a well defined set of standards to make it usable at any grade. Predicting that many separate stake holders would prefer to have general information (i.e personal addresses and phone numbers) stored somehow, commonly defined modules for these types of information would be a sought requirement.

3.3.1.1 Personal information

For the personal information, we propose a level based system, where one can chose to provide enough information for a certain level. Each stake holder could then require that a certain general level of information is provided within the certificate for the module to be valid. The drawback of this would be if the intra-dependant reference would break, resulting in loss of access. Also, supplying personal information, encrypted or not is a controversial topic for many people. An unencrypted personal information module system would certainly raise questions regarding the anonymity and integrity issues described further below. Therefore, it might be preferred for an individual to have parts of the personal information encrypted into the module of every stake holder that requires it, accepting the change in size and administration this would require. As one example, one could foresee medical information being kept privately in each module.

3.3.1.2 Standards

The main problem with defining standards is the intra-dependency it brings. Neglecting standards would lead to a system that is unusable, whereas a to strictly defined standard system would seriously hinder the flexibility and development of the system. Upgrading a strictly defined system often results in mere patches that solves the addressed problem in a less desirable way. To define standards that help growth of use while not hampering the development nor the module design of each stake holder might be one of the biggest problem faced when defining this kind of a system.

3.3.2 Anonymity

Being anonymous when authenticating your rights have become increasingly important during the last years. As more surveillance is added in our everyday lives, many feel that their integrity and anonymity is being threatened. There is no need for every authentication session to require that the rights should be directly connected to a real identity, even though this is all to often assumed and accepted by most. On the other hand, any friend of personal integrity would advocate that the opposite should be true - that when there is no absolute need for the real identity to be validated, it should be enough to provide access credentials without personal information. What we seek in form of anonymity is a system that in no way requires the user to reveal their real life identity while authenticating his or her rights to a stake holder.

3.3.3 Simplicity

Since this kind of solution should be possible for anyone to use, simplicity is as important as functionality. Without any user support, even the most brilliant technical solution would be in vain. Traditionally, user interfaces for authentication / authorisation are designed to hide most of the underlying technology, in order to lighten the burden on the user. Even though modularity would solve many of the technical aspects for the system, how would one still manage to create a system as easy to use as for instance a drivers licence? Even though this aspect mainly regards the user interfaces of the administrative and practical applications, we feel it would be wise to at least have simplicity in mind while designing the system.

3.3.4 Resource efficiency

One of the drawbacks of letting stake holders store information in a persona would be the overall size of the certificate. Even though technology of today certainly provide large data storage in relatively small sizes, one should not ignore the data size to be stored and/or transmitted. In many cases, data size reflects directly on the cost. Since the proposed solution should be able to be used on portable and embedded systems, the certificate size should be considered while designing the system.

As well as the above mentioned size issue, computational power might be a problem when it comes to embedded or portable devices. The system should be lightweight enough to be used in any environment.

Chapter 4

Existing solutions

4.1 X.509

4.1.1 Introduction

X.509 is a standardised format for digital certificates developed by ITU (International Telecommunications Union). The whole X.509 standard consists of over a dozen RFCs specifying everything from certificate management to transport of PKI operations. The first version of X.509 was made public in 1988, version 2 was made public in 1993, the third and current version (3) was proposed in 1994 and made public in 1995. This version addressed some security and flexibility matters in versions 1 and 2.

X.509 was originally developed as a authentication service for the X.500 directory service and still has a tight coupling to X.500 in the X.509 certificate format standard. The legacy from X.500 manifests itself by the way issuers and subjects are specified.[10]

4.1.2 Areas of use

X.509 certificates are by far the most common certificate standard used today. It has gained widespread acceptance in the computer industry and can be found in various types of applications. Its uses range from authenticating parties involved in an SSL encrypted communication to authentication of users wishing to login to a Linux workstation through the use of PAM.[5]

4.1.3 X.509 Certificate Specification

X.509 Certificates consists of the following information:

- Version - Identifies which of the three versions of X.509 that applies to the certificate.
- Serial number - The party that created the certificate is required to assigning each certificate a unique serial number in order to distingush it from other certficates it has issued. This information is used in numerous ways. Probably the most important use is when a certificate has been revoked, in which case the serial number is placed in a CRL.
- Signature algorithm ID - Identifies which algorithm was used by the CA to sign the certicate.
- Issuer name - X.500 name of the party/organisation that signed the certificate. Using the certicate implies trust of the signing party. Most often this is the CA.
- Validity period - Every certificate is valid for a specified period of time. The validity period is specified by a start date and time and an end date and time.

- Subject (user) name - The name of the party that the public key belongs to. This field uses the X.500 naming convention.
- Subject public key - the public key that is certified to belong to the subject.
- Extensions - This information is only present in version 3 certificates and can specify extra information that should be bound to the certificate. Examples are email addresses and dns names.
- Signature - A signature of information specified in the above described fields.

4.2 Liberty Alliance

4.2.1 Introduction

Liberty alliance is working to define standards and guidelines for a federated identity management system.[3]

4.2.2 Background

Liberty Alliance was founded in 2001 by Sun Microsystems as an answer to Microsoft's Passport. Liberty alliance is today a global consortium comprising of several fortune 500 companies. The companies that has joined Liberty Alliance is coming from several different sectors, such as data processing (RSA, SAP, Verisign etc), telecommunications (Ericsson, France Telecom, Vodafone) and from the service branch (Royal Mail, Visa)

4.2.3 Liberty Alliance Roadmap

Liberty Alliance has defined a roadmap comprising of different versions of the specification towards a federated identity management system honouring user privacy, security and user control of personal information.

Phase 1

Phase 1 is the specification and implementation of Liberty Identity Federation Framework (ID-FF). This enables identity federation through identity/account linking, simplified sign-on and session management.

Phase 2

Phase 2 is the specification and implementation of Liberty Identity Web Services Framework (ID-WSF). This specification will provide a framework based on web service technology. The framework will be a base to provide interoperable identity services, identity service description and discovery, and permission based attribute sharing.

Phase 3

Phase 3 is the specification and implementation of Liberty Identity Services Interface Specifications (ID-SIS). Using the ID-WSF specification, phase 3 will provide services such as personal identity profile, alert, calendar, geo-location, contacts and so on.

4.2.4 Liberty alliance key concepts

This section aims to explain the key concepts of Liberty Alliance

Federation

In the context of Liberty Alliance, federation means to establish a relation between two entities. This relation can exist between any number of identities and service providers.

Principal

Principal is the same as a "user" whose identity can be authenticated.

IdP

Identity Provider, the service which authenticates and asserts a principal's (users) identity.

Single Sign-on

The ability for principals to authenticate with one identity provider and have that authentication honoured by service providers.

Circle of trust

A group of identity providers and service providers that have a business relationship based on Liberty architecture and operational agreements. The whole idea is to allow users to transact business in a secure and apparently seamless manner.

DS

The Discovery Service provides identity based discovery of web service providers.

IS

Identity service is the definition of a service invoked through or with a user's identity.

4.2.5 Usage scenario

The idea behind Liberty Alliance is to allow a principal to seamlessly authenticate itself with service providers that have established a relationship. For example: Imagine you are a customer of Delta Airlines. You have registered at Delta airlines website and have just purchased a ticket to London. When the purchase is finished, Delta's website presents several different links that might be of interest to you: A link to Hertz in case you need to rent a car, a link to hotels in case you wish to book a hotel room and so on. Say you need to rent a car, then you click on the link to Hertz. When clicking on the link, Delta's website generates a random pseudonym and redirects your web browser to Hertz site, passing the generated pseudonym as a token. When arriving at Hertz site, Hertz will initiate a SAML connection to Deltas identity server to verify that the passed pseudonym actually exists. If this is the first time you click to Hertz from Deltas site you will be asked to log-in with your already registered Hertz username/password combination, or if you have not previously registered a Hertz account you will be asked to do so. The pseudonym generated by Delta will then be associated with your registered information and allows Hertz to log you on automatically the next time you click to Hertz from Deltas website.

Note that the only shared information between Delta and Hertz is the generated pseudonym. Hertz does not know of any information you registered at Delta, likewise Delta has no idea of what information you registered at Hertz. An identity has been created that cannot be explicitly tracked beyond a single corporations knowledge about you.

4.3 SSL

4.3.1 Introduction

SSL, Secure Sockets Layer is a protocol developed by Netscape in 1994 as an effort to tackle growing security concerns. There are 3 versions of SSL, although the first version was only used internally at Netscape. Version 2, released in the fall of 1994 suffered from a number of security flaws, where the most serious one was the vulnerability to a "man in the middle" attack. Version 3, released in November 1995 was designed with public review and input from industry and was published as an internet draft document. It has since grown in popularity and has become the standard for securing HTTP traffic on the Internet.[28]

4.3.2 Certificate model

SSL exclusively uses x509 certificates to ensure that the corresponding party is whom it claims to be. The X509 certificates contain the public key of the server, which is used initially to establish a secure channel of communication between server and client. The initial channel is then used to transfer a symmetric key in a secure fashion. The symmetric key is used for further secure communication between server and client. The rationale behind this exchange of a symmetric key is that symmetric encryption is generally much faster.

4.3.3 Trust

A central participant in SSL trust issues is the CA (Certificate Authority). The CA is a trusted party, such as Verisign, Thawte etc. The certificate authority has a "master key pair". One public key and one private key that is used to sign the X509 certificates the CA issues. When a certificate is issued, the administrator installs it on the corresponding SSL enabled webserver. When a client request a secure channel to such a server, the server sends its public key with its certificate. The client then makes certain that the certificate was issued by a trusted party using the public key of the CA, that the certificate is related to the contacted server and that the certificate is still valid.

4.3.4 Areas of use

SSL is the most widespread mean of secure communication on the Internet today.Initially developed only to secure HTTP-traffic, it has since evolved and is today used to secure various types of communication, such as IMAP, POP3, SMTP etc over the internet.

4.4 PGP

4.4.1 Introduction

In 1991, Philip R Zimmerman released version 1 of Pretty Good Privacy (PGP[30]). Although facing opposing forces in both RSA and the US Customs office, PGP has risen to become the most popular signature and encryption standard used by the public today. Its main use lies in email signature and encryption, but tools like disk encryption have emerged.

4.4.2 Certificate model

PGP offers two different certificate models per default; PGP's own certificate model and X.509. As X.509 is discusssed in a separate section, we will focus on PGP's own certificate model here.

The PGP certificate is based on an identity scheme where a user id is the central information. Each user id must contain a name and an email address. There is also support for photos to

connect the certificate to the physical appearance of the holder. Every part (user id or photo) is self-signed by the private key. Each certificate can hold multiple user id:s and photos, making it rather modular in that sense.

4.4.3 Trust

Since there exists no demand for a third party CA in PGP, the means of securely distributing public keys have been solved rather differently. The idea is for the user to contact the key holder and validate that the fingerprint of the signature matches. Even though this might be considered a rather "secure" solution to the problem, this procedure is hardly practical when it comes to larger communities. For this purpose, PGP support a trust system where users can sign each others public keys. With this system, you can chose to trust a new key based on it being signed by a key you already trust. However, there is no way to revoke a signature on a key. Should a user whom you have signed to be trustworthy show negligence with his/ her key, it is not possible for you to revoke your trust. In effect, one has to be extremely restrictive when choosing who to trust, and even more so when it comes to signing. In close groups, the damages of a hijacked key are severe enough, but could be controlled via other means of communications such as telephone or face to face meetings. Used in a larger scale, the impact of breaches in the trust chains would be to big. One would not be able to trust keys based on chains of signatures. Because of this, PGP supports trust to be establish if two or more people that are already fully trusted have signed the new identity. The risk of all the signing parties to be fouled by a malicious user is decreased dramatically, thus allowing for a higher probability of a valid identity.

4.4.4 Areas of use

PGP was created to give the public a way to gain privacy by encryption in their everyday lives, not to create a global identification system. In that sense, it does not quite fit in this section of "Existing solutions". However, since it is one of the solutions currently most used to identify people over the Internet, we feel that it is worth to analyse. We believe PGP is primarily used in email based communication, even though new uses have appeared (such as the disk/file conventional encryption utility). Even though its focus may not have been directed that way, the lack of systems using PGP public keys for general authentication shows that it might not be suitable for such a job. The main use for PGP has been for authentication between few people in a rather small community.

4.4.5 Conclusion

4.4.5.1 Pros

Even though not built for a general authentication scheme, what we can identify as pro:s with PGP in this context are:

- No need for a CA
- Simplicity in key management
- Strong cryptographic solution
- Flexibility (to some extent) in user information

4.4.5.2 Cons

However, what we consider all solutions lack is simple module management. For it to be suitable for a more general authentication standard we consider the following ussues to have a negative impact:

- Trust issues when used in a larger context
- Not able to revoke signatures
- Lack of modularity/flexibility in certificate information
- Simplicity in key management

4.5 .Net Passport

Microsoft released its passport service in 2001 as a single sign in service. Their goal was that a single sign in service should entice surfers and websites to use it for all functionality that requires signing in. Although initially widely supported by on-line commercial sites, the popularity and support have since faded as both security vulnerabilities and privacy issues have arisen. Microsoft no longer actively support the use of Passport for third party enterprises.

Although not directly comparable in its architecture/model to the other solutions, the Microsoft .Net passport saga has in our opinion several valuable lessons to learn.

4.5.1 .Net passport technical specification

Passport allowed users to create accounts by providing personal information such as name, email and postal address, country etc. When users visited a passport enabled web site they did not have to sign in again with another user name and password, instead the user's identity was verified by the passport service. The user information was centrally stored in Microsofts Passport servers.[6]

Hence the Passport service has been withdrawn for non Microsoft sites the exact details of the authentication process has been scraped from Microsoft website. Therefore we have only been able to use third party sources for the technical specification and may therefore be somewhat incomplete. It should be noticed that the following information is not taken from, or affiliated with Microsoft.

To be able to authenticate to Passport enabled websites the user has to log-in into the Passport account providing a user name and passport. During this process, the user is redirected to a server in the .passport.com domain. When the username and password have been validated, the Passport service sets a variety of cookies in the .passport.com domain. When the user logs in to a site that is not itself passport.com the user is redirected back to that site in a manner that sets various cookies for the domain the user wished to login to. Theese cookies are then used by the Passport enabled site to authenticate the user. This description is a bit oversimplified, for example the cookies is encrypted using 3DES and there are possibilities for a Passport enabled site to specify variables like requirement that the last manual sign in isn't longer ago than a specified time. However the architecture and implementation of Passport is not our main focus here, instead we want to point out how fragile the acceptance for a digital identity system can be.

4.5.2 Why it failed

4.5.2.1 Security breaches

The passport service has suffered several security breaches[6]. Our belief is that the cumulative effect of these security breaches lead to that the giants of the web such as eBay, Monster.com, La Times among others abandoned the passport service.

4.5.2.2 Centralized system

After the security breaches in the Passport service was discovered, the "consumer confidence" for the service dropped as predicted by Gartner.[12]

The security breaches in combination with the centralized nature of Passport made the Passport enabled websites worried that a breach of the Passport servers could mean that personal information of a web site's whole user base could potentially be compromised. As described by A.D. Rubin[27],

> "Dictating (or even strongly recommending) the use of technologies that are not felt to be trustworthy in a system whose purpose is to establish trust can undermine significantly the perceived value of that system".

It is therefore crucial to the acceptance for such a system that it is built around technologies that has the public's confidence and in case of a security breaches, the damage can be controlled.

4.6 Kerberos

4.6.1 Introduction

The work Kerberos originates from the legend of Cerberus in Greek mythology. Cerberus was a vicious creature and is described as a three headed dog that guarded the underworld ruled by Hades. Cerberus was the gatekeeper of the underworld and authenticated thoose who wished to enter. Like Cerberus, Kerberos authenticates thoose who wish to access network resources.The modern Kerberos is a network protocol designed for strong client/server authentication using traditional symmetric key cryptography.[28]

4.6.2 Background

The Kerberos protocol was created by Massachusetts Institute of Technology as a research project in the early 1980s. The resarch project was sprung out of the recognition by MIT that huge increase of widely availble inexpensive computers would change the computer industry. MIT also offers a free implementation.[4] Since the source code is publicly released anyone who wishes can download and assure themselves that the code is trustworthy. Several different Kerberos implementations are also available from commercial vendors.

4.6.3 Goals

The Kerberos authentication protocol was designed with the following four requirements in mind

Security

The protocol must not allow a network eavesdropper to obtain enough information to impersonate a valid user. Kerberos never transmits a password in cleartext over the network. By using time limited tickets which is used to grant access to services it also protects against replay attacks and prevents that an eavsdropper can decrypt the service granting ticket in reasonable time.

Reliability

All services that uses Kerberos to authenticate users relies on the availability of Kerberos in order to offer availability themselves. Therefore the Kerberos service must be highly reliable and offer failover services where one system is able to back up another.

Transparency

The authentication process should be as invisible to the user as possible.

Scalability

The system must be able to support a large number of clients and servers.

In order to fulfil these requirements, the general architecture of Kerberos is to use a trusted third party service for granting clients access to services. The trusted third party is called Authentication Server, and acts as the central point in a Kerberos enabled network.

4.6.4 Kerberos usage scenario

This section aims to explain a typical usage scenario of a Kerberos authentication session. In order to meanfully explain the usage scenario, we first have to define some artifacts involved in the scenario.

4.6.4.1 Scenario artifacts

C: The Client requesting access to a service.
AS: Authentication Server, granting or denying the client access to the requested service.
S: Server.
Cid: Identifier of user on C.
Sid: Identifier of the server S
Cpass: Password of user on C
Cad: Network address of C
Ksas: Secret key shared by AS and S.

4.6.4.2 Scenario

A user logs on to his or her workstation and request access to a service on Server S. The client module in C prompts the user for his or her password and then sends a message to AS comprised of the user id, the server id and the user's password. AS checks that the supplied user id and password combination is correct and that the specified user has sufficient permissions to access the service hosted at S. If both tests are passed the AS creates a ticket comprising of the user id, the network address of C (Cad) and the server id. The created ticket is encrypted with the secret key Ksas. The encrypted ticket is then sent back to C. C can then use the ticket to apply for service at S. Since the ticket is encrypted it cannot be changed by the client or a hostile opponent.

When C wants to access resources at S, it sends a message to S containing the id of C and the encrypted ticket. S then decrypts the ticket and verifies that the user id in the message is the same as the id in the ticket. If the IDs match, the server considers the user authenticated and grants the client access to the requested service.

4.6.5 Feature matrix

From the analysis of the existing solutions we have formed a "feature matrix" of the in our opinion most important requirements.

	PGP	SSL	.NET	X.509	Kerberos	Liberty Alliance
Flexibility in user information	-	-	-	-	-	x
Simplicity for user	x	-	x	-	x	x
User "control"	x	-	-	-	-	x
Simple Key Management	-	x	-	-	-	-
Accepted Industry Standard	x	x	-	x	x	x
Scalability	-	x	x	x	x	x

Figure 4.1: Feature matrix of evaluted solutions

4.6.5.1 Flexibility in user information

Flexibilty in user information is how flexible the solution is in regard to required information needed to identify the user and the possibility to specify additional more or less rudimentary information.

4.6.5.2 Simplicity for user

Simplicity for user is unfortunatley a subjective requirement. One cannot directly compare for example PGP with SSL since it target different areas of use. Although, simplicity for the end users is required for almost every technical solution to be widely accepted and embraced.

4.6.5.3 User control

The user control requirement aims to specify whether or not the user can control the information being presented to the validating party.

4.6.5.4 Simple key management

Simple key management takes into account how keys and key exchanges/negotiations are handled. For example if the solutions requires a lot of administrative work.

4.6.5.5 Accepted industry standard

If the solution is embraced by a standardisation organisation like IETF/IEEE/ISO etc or embraced by an industry consortium.

4.6.5.6 Scalabilty

This measure takes into account the scalability of the evaluated solution in its context of use. For example, single sign-on can be scalable enough to be used inside a corporation and its different branch offices but not scalable in the same manner as solutions intended to be used on the Internet like Liberty Alliance or PGP.

Chapter 5

Proposed model

5.1 Introduction

In this chapter, we will discuss the proposed solutions to the identified requirements. Focus will be put on the validation procedure and the implications a decentralised system will bring.

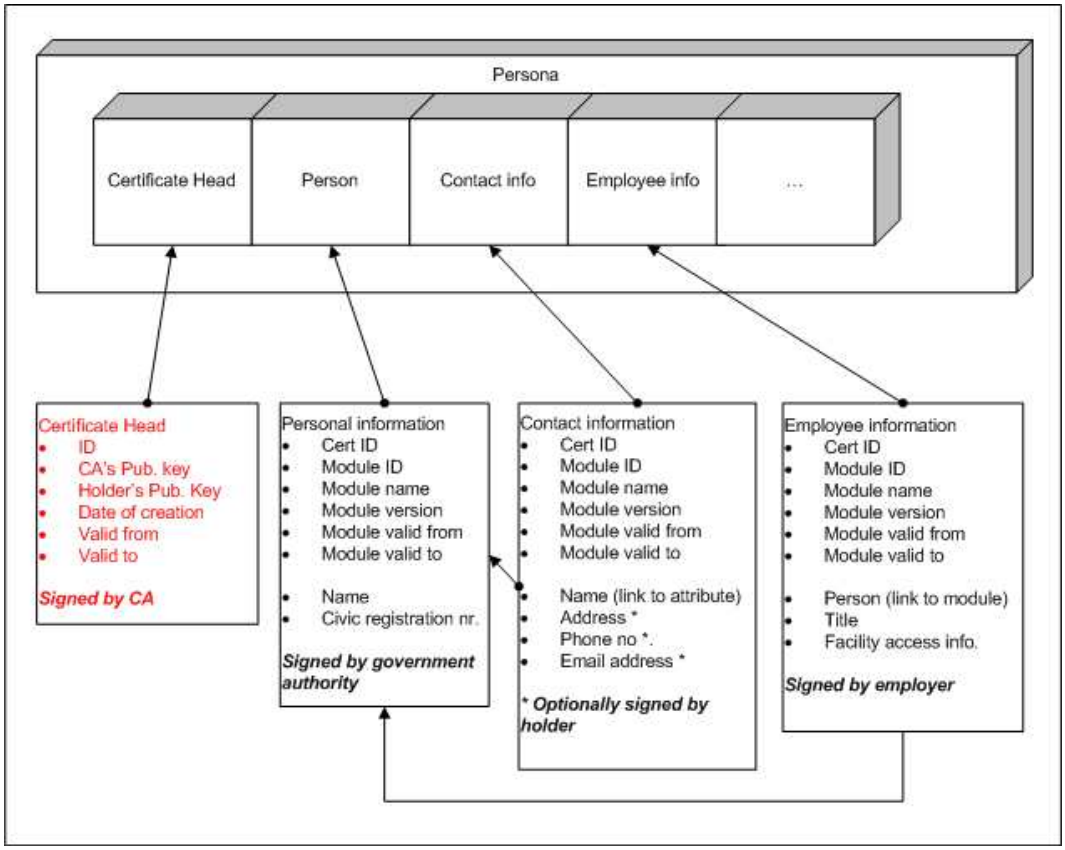

Figure 5.1: Example of Persona certificate

5.2 Protocol & specifications

5.2.1 Certificate envelope

5.2.1.1 Certificate authority

In order to create a certificate, you need to create the base on which to tie your modules. This container, further referenced to as "Certificate envelope", is issued by a traditional Certificate Authority. The Certificate envelope is basically just like any other module, but since it is required for all other modules to function it was named certificate envelope.

The required artifacts in each and every certificate envelope are:

- Certificate id

  The certificate id is the artifact that uniquely identifies a certificate envelope and is required to map modules to a specific certificate envelope. Separating the identifier from the users public key creates a more loosely coupled structure. By doing so, we gain traceability of a certificate, so that a validating source easily can identify and contact the certificate / revocation authority. In addition to this, it would be technically possible to reassign a certificate's key pair. Even though this would facilitate recovery from revocation, it would further complicate the already delicate problems of revocation distribution and checking.

- Certificate authority public key

  The public key of the certificate authority that issued the certificate envelope. This is the corresponding public key to the key used for signing the certificate envelope in order to ensure the integrity of the envelope.

- Client public key

  In order to tie the certificate to a person, a standard PKI keypair is used. In order for validating sources to verify that the party presenting the the certificate is the owner, a challenge is made using this public key as the cryptographic key.

- Signature of the envelope

  The certificate envelope is signed with the CA:s private key. The signature is used in conjunction with the CA:s public key in order to ensure the integrity of the certificate envelope.

- Date of creation

  The date when the certificate was created.

- Valid from

  The date the certificate envelope is valid from.

- Valid to

  The date the certificate envelope is valid to.

- Certificate revocation information

  Optional, alternate way to look up possible certificate revocation. As an example, a third-party revocation authority could be suggested here in case the primary authority is unreachable.

Figure 5.2: Structure of a certificate envelope.

5.2.1.2 Flow of information

The client initiates a creation of certificate envelope by sending a certificate envelope request message to a certificate authority. The only required data in this message is the public key of the client. Other information such as name, social security number etc may be required by the certificate authority in order to approve the certificate envelope request message and create a certificate envelope. When the certificate envelope request message has been validated, a unique certificate envelope id is generated. The certificate authority then uses its private key and the client's public key to create a signature of the created certificate envelope.

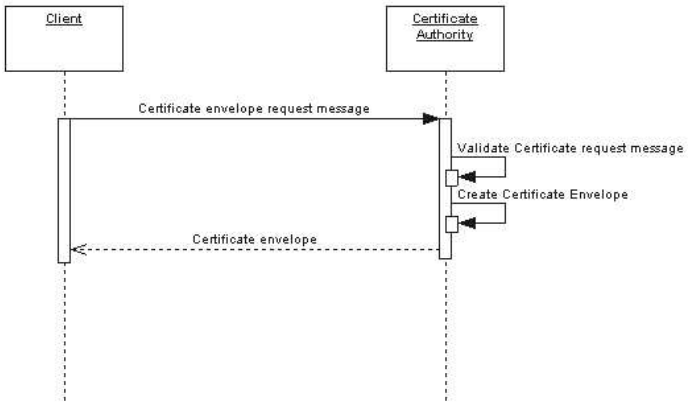

Figure 5.3: Client -> CA interaction diagram.

5.2.2 Module Authority

5.2.3 Flexibility

The primary guideline here is flexibility. Therefore, as few items as possible should be required. Our proposal is to keep as much of the validation logic for the validating authority to decide. For third parties to use a module, the creator should provide an API-like specification in cases where one wants to provide the possibility for module dependency. One could even imagine portable code being distributed in the module as to automate the validation procedures.

5.2.4 Creation of Modules

A module can only be created by a certificate holder, so that any module can be traced to its creator. Every module is also absolutely linked to the certificate it was created for. Essentially, the certificate id is the only thing that connects a collection of modules when presented to a validating source. Each module has also an id, so that any module in any certificate can be uniquely identified based on creator id and module id. For third party interfacing, each module also contains a name and a version. Any third party relying on the current module would specify this by creator, name and version. In addition to these variables, a module also contains timestamps for the modules validity in terms of "valid from" and "valid to". This information is placed in the first node of the module root and called "HEAD".

5.2.4.1 Signatures

The entire module should be signed by the creator in order for it to carry any level of trust, but this is not required. Personal information of lower trust value that might be subject for frequent changes could be unsigned by the certificate holder in order for it to be changed with more ease. This mix of signed and unsigned information could be solved by a coming feature of BTL[1] that we intended to incorporate in the suggested solution. In BTL, signatures are categorised as monitors, and BTL will supposedly support mid-stream pausing of monitors. This solution would enable the suggested solution to have partly signed modules, enabling the modules to be even more flexible. This will be more thoroughly discussed in the section BTL[2] In order for multiple sources to approve of a module, multiple layers of signatures shall be allowed in the specifications. With partial signing of modules available (partial should not be limited to a subnode recursively, but any number of chosen subnodes in the module hierarchy), one would allow for third parties to sign chosen parts of the module.

5.2.4.2 Encryption

Since a module is rather likely to have some kind of information that is preferred to keep private, encryption shall be available. Although a PKI based system, asymmetric encryption is considered less suitable for larger and streamed data due to speed issues. The encryption model suggested is for the actual data to be encrypted symmetrically with any chosen encryption and a randomised key, adding encryption information and key asymmetrically encrypted as attributes to the data. The attribute should be available encrypted with the public key of any party that should have read access to the data.

5.2.5 Validation of Modules

As previously mentioned, the validating authority will determine validation criteria from its own rule sets, even though "recommendations" and/or guidelines can be included in the module. Ba-

[1] see ch. 5.3.2
[2] see ch. 5.3.2

sically, a validating server will require one or several modules when approached. When given the requested modules, signature verification and module validity (if needed) should be checked. As mentioned above[3], one possibility would be for the module creator to include a rule set for the validating source to run for the module to evaluate as valid under the current circumstances. However, we foresee the main issue when validating modules to be revocation controls. Any validating source should be able to verify that the presented signatures, modules and certificate are still valid. In the event that any of these revocation checks return information about a revoked entity, fail-over rules should be implemented. Systems for revocation checking will be discussed further below[4].

5.2.5.1 Information flow

We suggest the following two ways of data flow between Client C and Stake Holder SH:

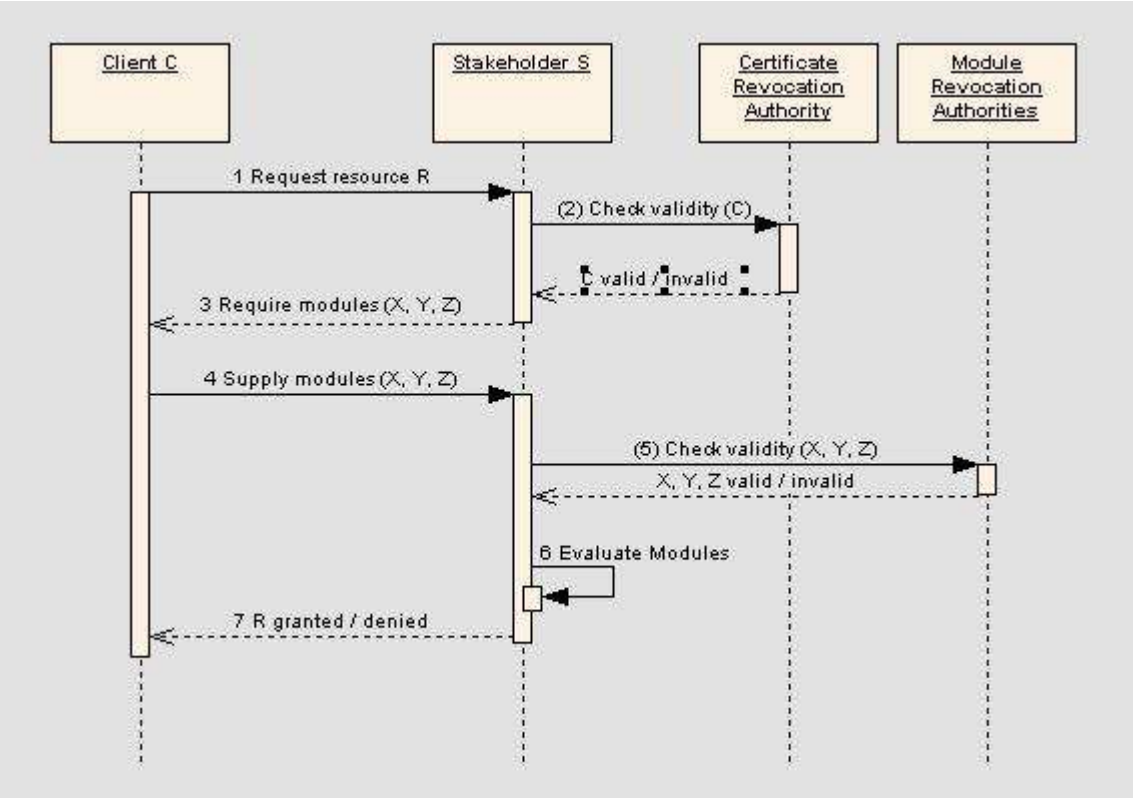

Figure 5.4: Scenario A

Scenario A

1. Client C requests access to a resource held by Stake Holder SH.
2. SH optionally checks for revocation of C's certificate to decide upon further checking, and aborts if not valid.
3. SH requests needed modules for the requested resource.
4. C supplies SH with the requested modules.

[3] see ch. 5.2.3
[4] see ch. 5.2.8

5. For each module requested (and supplied), SH may chose to check for revocation.

6. Based on module validation result, SH evaluates whether or not to grant C access to the requested resource.

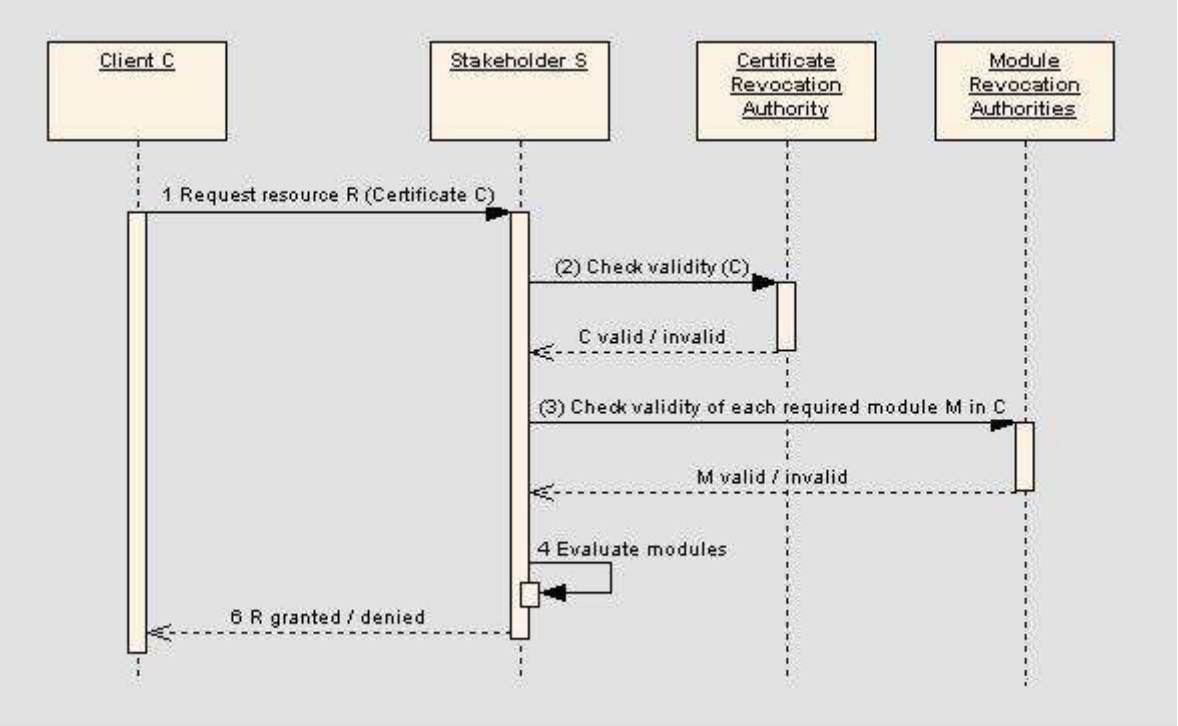

Figure 5.5: Scenario B

Scenario B

1. Client C requests access to a resource held by Stake Holder SH. C sends its entire certificate.

2. SH optionally checks for revocation of C's certificate along with any modules required against the appropriate Revocation Authorities.

3. Based on certificate and module validation result, SH evaluates whether or not to grant C access to the requested resource.

5.2.6 Module linking

5.2.6.1 Introduction

In order to make a modular system efficient, avoiding redundant data is essential. Even though it should be possible for a module to be entirely atomic, it is perhaps more likely that it in someway is dependent on other data. As an example, a module containing access rights could require personal data (i.e name, address) to be bound to the existing information. Since one can assume many similar modules to require the same information, instead of letting each module store this information, one would let each of these module reference a separate module containing the sought information. This would be one of the strongest arguments for using a modular certificate system. Should basic information, such as personal addresses, change, it would only reflect on the corresponding module. No relying module would have to be changed.

On the downside, even though avoiding redundant information, linking to other modules brings other questions - what should be done if a link is missing? Is it always preferable to rely on data that can not be guaranteed? What authority can define a module that third-party module authorities shall rely on?

5.2.6.2 Defined standards

To allow for different parties to share information in modules, it requires clearly defined standards for how these modules should be designed. It would be possible to import already existing data schemes (i.e the vCard standard[9]). The standards should be enforced by a (to the greatest extent) impartial and trusted source, proposedly the IETF. Protecting the standards are as important as defining them. If not actively maintained and updated, they would soon fail to serve their purpose.

5.2.7 Specifications

5.2.7.1 Certificate envelope

For the Certificate ID, we propose a 128bits integer. Even though a 64 bits integer would have been more than enough to ensure that we would not run out of address space, using 128 bit allows for better hierarchy support. The idea is for the certificate Authority chain to be present in the certificate Id, so that revocation and certificate lookups can be facilitated. The receiver's and creator's public keys should be ASCII-encoded byte arrays, with their size stored as attributes of the nodes. The date of creation along with valid from and to should be int64 representations of seconds since 1970-01-01 (epoch time)

5.2.7.2 Modules

A module is identified by two variables combined. The issuers certificate ID, and the serial number of the module created by the issuer. A 32 bits integer would only allow for 4,3 billion modules per issuer, which may call for the serial number to be an int64, granting a range of over $1,8 * 10^{19}$ modules.

5.2.8 Revocation

5.2.8.1 Introduction

As mentioned[5], one of the major issues of PKI has been the revocational models. The stake holder has to be sure that the public key presented is coupled with a private key that has not been compromised. Traditionally, a revocation is reported to the Revocation Authority (which may or may not be the Certificate Authority) which then serves querying clients.

Although the distribution of revocational information traditionally has been considered[21] one of the most resource demanding issues with PKI, the problem of revocation is much greater in our suggested model, since there are multiple sources to check for revocation. When validating a module, the validating entity should optimally be sure that every key of every signature of every module it chose to check, in addition to the certificate key itself, is valid. Given that the modules may be intra-dependant in a tree structure, the number of elements to check can be rather large. However, in reality, the validating entity should chose to check for revocation only to the extent needed in the current context. Therefore, module revocation should be completely up to the validating source. However, since the module certifier may have a better insight in how the module preferably should be validated, revocation checking information could be included in the module. In this scenario, multiple ways of checking for revocation (by whatever mean specified)

[5] see ch. 3.2.3

could be defined, and even prioritised. One could also use some kind of rule set where a decision is taken based on multiple checking systems, where each is valued after a specified scheme.

5.2.8.2 Existing models

Direct querying / OCSP The simplest, and in small systems perhaps often the best, way to check for revocation would be a query directly to the issuing Certificate Authority or the Revocation Authority specified by the CA. This way, one can always be sure to have the most updated view of a certificate at all times. No cache or TTL problems are faced, no lists have to be parsed. Obviously, the drawbacks of this are primarily the load this would put on the Revocation Authority, the requirement of guaranteed communication with the RA, and the speed of revocation checking. IETF[2] has proposed a standard for direct querying of certificate status, called "On-line Certificate Status Protocol" or OCSP[19] for short.

CRL Certificate Revocation Lists, included in the X.509 standard, is perhaps the most used standard for certificate revocation. It has been criticised of being resource costly and inefficient [26], but also extended and modified with a variety of different optimisations. In its pure form, CRL is a time stamped, signed data structure of revoked certificates. A CRL is periodically issued from the RA, even if no new revocation has been made, in order for a client to ensure it has an updated revocation list.

Overissued and segmented CRL In 1999, David A. Cooper presented two new ideas[7] of CRL optimisations, overissued and segmented CRLs. In the case of overissued CRLs, the idea is for the RA to issue new CRLs while older and valid CRLs still exist. With this technique, the clients cache would not expire at the same time, thus distributing the requests evenly over time, reducing the problem of server peak loads. With segmented CRLs, the idea is to create smaller CRLs, thus making each transaction smaller, allowing the server to handle a larger number of requests simultaneously.

Delta CRLs As the name implies, when using delta CRLs, any issued delta CRL only contains changes since last base CRL was issued. This allows for the load to lighten between base CRL releases. However, there have been propositions to optimise delta CRLs as well. Using sliding windows would give a similar effect to the one mentioned in the previous paragraph. Issuing base CRLs with a high frequency but long life time, along with matching delta CRLs will even out traffic load on revocation authorities.

DNS piggyback A couple of ideas[23],[8] have been presented, for PKI to use DNS (Domain Name System), or DNS-like systems as distribution system. Since CRLs have the same fundamental structure, CRLs could be distributed in the same manner. This would enable a revocation system with fast responses and high availability. However, since every node would keep some kind of cache, one of the negative aspects would be the speed of data propagation in this system. Reducing the load of the authoritative server to the cost of a greater number of clients dependant on local server cache.

5.2.8.3 Suggested revocation model

As for CRLs and its optimisations, there are a number of cases where these techniques are not especially useful. In our case, if a validating entity faced periods of time without the possibility of network operation, segmentation would not be of any use. In order for the validating entity to make sure that the cache was updated, the entire list would have to be cached. On the other hand, CRLs in general and overissuing in particular will create a time granularity the size of CRL

expiration time, thus increasing the risk of accepting an invalid certificate, module or signature. However, what one looses in cache expiration time with CRLs, one gains in speed and availability.

When it comes to direct querying, this method would only be preferred if network operation could be guaranteed at all times, or the security implications were low enough for revocation checks to be skipped. On the other hand, for all validating entities with high security demands and guaranteed (free / low cost) communication towards the revocation authority, direct querying would almost always be preferable.

5.2.8.4 Certificate validation

Certificates should not (in theory) be revoked especially often (the changes in a certificate should rather affect modules). Revocation of a certificate should only be made in the cases of private key compromise, loss of private key, or CA compromise. Based on this, two conclusions are drawn:

- Even though the number of revocation checks that would be made can be expected to be rather quantitative, the traffic should be reasonably low due to the low frequency of certificate revocations.
- When a certificate is revoked, it should in most cases be rather urgent to propagate this information.

When it comes to revocation of the certificate, we suggest a model similar to the DNS-piggyback system described above. Since every certificate shall be traceable through its chain of authorities, the revocation authority could be directly accessed based on a certificate ID. Should your local CRL/DNS server's cache be expired (or lack the certificate information all together) a look-up from the authoritative server would be made and cached. The cache would mean a slower revocation propagation, but for stake holders of resources of a more critical nature, a look-up directly against the authoritative server could be made.

5.2.8.5 Module validation

When it comes to the revocation checking of modules, it is entirely up to the module creator to provide the means of controlling revocation. However, as a recommendation, the same system as for certificate validation could be used. Modules should, in theory, be revoked more often than a certificate, since they should reflect the status of an entities access rights and/or attributes. Being of a slightly less critical nature than certificate or CA compromise, one could use larger TTL:s to lower the stress of revocation servers.

5.2.8.6 Signature validation

Unfortunately, we consider the requirement of signature validation less likely to be solved. Adding revocation checks for each signature in each module would require enormous amounts of traffic and administration. Even if the revocation systems and the validating clients were to withstand the traffic and computational power, the problem of signature administration for the users would still be a major problem. Every signing entity would have to keep a public record of revoked signatures. In effect, for every entity, there would have to be a signature repository. Requiring that every user constantly monitors the subject of a signature would not be feasible, and would make the system unusable for a major part of the intended users. On the other hand, supporting a signature revocation system that only a minority uses would instead lead to a false sense of security. While the trust system of PGP may work in small, well-known groups, we believe it would not work in a global identity system.

5.3 Data format

5.3.1 Introduction

After identifying the most basic requirements, we had to choose which file format to use. Since we wanted a node structure and a format that would not be platform dependent, we started to examine various serialisation formats that supported this. XML is widely supported but hardly resource efficient. Binary XML is emerging, but not supported in a greater scale. We sought a binary format with higher performance, preferably with support for real time data modification and supervision.

5.3.2 BTL

A project that seemed to have similar aims was BTL, Binary Tree Language. The specification(Appendix A) for BTL was written by Ulrik Mikaelsson in 2004. While the reference implementation was not quite finished when we adopted the idea, we felt BTL to be a solid ground to build on. It provided a number of requirements we had identified.

The main problems BTL solves in our model are:

- Streamable data handling

  BTL is built for use in (file or network) streams. Since the primary use of a certificate involves exchange over some kind of network, data streaming would enhance the performance.

- Filters

  BTL incorporates the notion of filters to modify data. Basically, one can switch on and off filter in the streams, and since filters are stackable, multiple modifications can be done simultaneously. In our case it provided the possibility for encryption/decryption as well as compression of data.

- Monitors

  The purpose of a monitor in BTL is to, based on input data, calculate an output without modifying the input data stream. Monitors are designed similar to filters in that way that they can be applied anywhere in the stream, as well as stacked independently of each other. In addition to this, it should also be possible to pause monitors within a stream. This piece of functionality would provide us the possibility to sign chosen parts of a module, so that data within a module could be modified freely without breaking the original signature. As an example, say that a module contained personal information. i.e. name, social security number, address, phone numbers and email addresses. A trusted third party, such as a government authority, could sign the information about your name, address and social security number, while the data that is more likely to change, such as the phone numbers and email addresses could be freely modified by you without breaking the signed information.

- Efficient data storage

  Since BTL is, as the name suggests, not plain text, much is gained in the storage efficiency. In addition to this, the possibility for further compress the data exists.

- Referencing

  Although not yet defined in the specifications, we have received information from the author that node referencing will be available in BTL. This provides back-end support for the items discussed under *Module linking*[6].

[6] see ch. 5.2.6

- Tree structure

  Modularity is reached in BTL by using tree structures for the data storage. A tree is a set of nodes, where each node can contain any number of subnodes or data elements. A node also contains a set of attributes.

- Typed data

  Using typed data allows for greater performance, and eliminates the need for extra parsing.

- Explicit encoding

  By using explicit encoding, it is always clear which character set to use to decode the data, thus getting rid of the need for a client to "guess" the encoding.

- Performance

  Although yet to be proven, BTL is designed to be optimised performance-wise.

5.4 Validation

5.4.1 Introduction

The often complex, and perhaps most interesting part of an identity scheme is the procedure of authentication and authorisation. Even though authentication is a non-trivial task, if one aims for some kind of soft security scheme when authorising a user, the evaluation evolves to a complex decision tree. The process of authenticating an approaching entity and separating authorised clients from unauthorised ones involves security aspects at a number of levels. We will discuss these levels in falling order from more abstract to more concrete.

5.4.1.1 Authentication and Authorisation

It is important to differ between the terms *Authentication* and *Authorisation*. Authentication is the process wherein one verifies that an entity really is what it is said to be. Authorisation, on the other hand, is the process of deciding which resources an entity is entitled to.

5.4.1.2 Hard contra soft security

When it comes to the policies of authorisation, they can be grouped by hard and soft security. In hard security, an entity is either given full access to a resource, or none at all. Packet filtering in a firewall can be considered to be hard security, since a packet is either let through, or denied routing. However, it most cases, security has to be weighed against use, and a more flexible security has to be used. Soft security[25] requires more "intelligence" in form of misuse detection heuristics.

5.4.2 Authorisation issues

As previously mentioned[7], data links between modules will require the validating entity to prioritise between data, using fall-back data when needed.

[7] see ch. 5.2.6

5.4.2.1 Rule sets

In order to have hierarchies of nested module links, we suggest the use of simple rule sets. As an example, one could use the directives "requires" and "includes" based on module name, provider and version (i.e Requires <provider>, <module name>, >= v.2). Another suggestion would be for links to have fall back options, so that if the primary link fails, another can be used. In rare cases, basic fall-back data could be stored redundantly in the module. In the case of personal information, primary link could be to a module with a complete vCard, while fall-back data stored in the module could contain name and address from when the module was issued.

The interesting part of this, however, is to process the presented data in order to apply the correct policy, relating to an ACL or several levels of ACLs.

We propose the use of a policy language (similar to, or based on for example XACML[14] or CARDEA[18]) defining required combinations of data together with required revocation and validity checks for each part of the data, resulting in a decision to grant or deny access to the requested resource.

5.4.2.2 XACML

XACML (eXtensible Access Control Language) is a general purpose access policy language based on XML, that defines the syntax for a policy language and the semantics for processing the policies. Conceptually, XACML differs between the Policy Decision Point (PDP), which evaluates the request and determines the authorisation level and the Policy Enforcement Point (PEP), which then enforces the authorisation decision. Even though we have mainly used the term validating entity as the PDP in this paper, one could separate between the deciding entity and the enforcing one if that logic applies better in one's system.

5.4.3 Trust

5.4.3.1 What is trust?

Although discussed in numerous papers, the definition of trust is still rather unclear and debated. In Diego Gambetta's words, trust is:

> "a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action"[11].

But what is it that you have to trust? The agent's ambitions? That he does not intend to act in a way that would be harmful for you? Would that not limit the actions to what the agent *consider* to be harmful for you? In that case, trust would only be a combination of benevolence and poor knowledge. On the other hand, trusting an agent to always analyse how his actions would affect you would not only be foolish, it would also be insufficient, since there always could be chains of events of which the agent could not forsee, or even act against. In this context, we define trust as the level of certainty that we can afford to put into the response we get from another party.

Centralised and decentralised trust models In contrast to the ad-hoc and centralised trust models discussed previously in this paper, we suggest a hierarchy based but decentralised trust model. Even though the traditional CA topology exist for the certificates, the validating entity have to rely on third party module authorities and their revocation authorities. For every validator, there exists a certain set of possible module authorities to verify against, that the validator is completely dependant on.

5.4.3.2 Obtaining trust

Actors In our suggested model, a validating entity evaluates its trust versus the client based on its trust it has regarding the client's certifier (CCA), the module creator's (MA) and the revocation authorities (CRA, MRA) of the CCA and MA. In effect, the bonds between these entities and the validating entity must be strong enough for the entity to initiate any kind of relationship with the client. Both MA and CRA are relatively easy to enforce trust bonds with, since these are specified by the validating entity in accordance to the context. The CCA, however, is with high probability, relatively unknown to the validating entity. This bond is created by trusting the chain from which the CCA is certified.

5.4.3.3 Maintaining trust

Unfortunately, even more essential than creating trust bonds is updating them. As mentioned previously[8], revocation within traditional PKI is done by the Certification Authority appointing a Revocation Authority which keeps track of revoked certificates. Our suggestion is for trust to be maintained via a network of revocation distribution points[9].

5.4.4 Responsibility

An interesting question raised with a decentralised authorisation system is that of responsibility. When there is no single authority that certifies everything, who is to take responsibility if / when a breach in the security occurs? For such a system to be operational, publicly defined areas of responsibility must be accepted by all participants. Obviously, the validator must be responsible for verifying that the presented certificate and modules are valid, and for the evaluation of the level of authorisation. The RA is responsible for presenting the latest status of a certificate or module, but who is to assume responsibility when for example the communications are down, resulting in denial of service?

5.4.5 Session security

In a validation scenario, many different entities could be involved[10]. In this decentralised system, how would it be possible to keep every session atomically secure, so that session integrity could be enforced? In order to acquire this, every communication must be secured. Communication between the validating entity and the trusted sources is made over a non-trusted network. If any part of the validation (communication with trusted parties) should be compromised, the session integrity falls. Even though possibly resource inefficient, each validating entity must go to great lengths to ensure that the kept keys of the trusted parties are updated, so that information communicated over the network cannot be forged.

5.4.6 Software security

On a lower level, one has to ensure that the software validating an approaching client is able to withstand maliciously crafted certificates, intended to illegitimately gain access to the system. No protocol or system is secure if the software implementing it is vulnerable to attacks. Even though no software can ever be guaranteed to be free of bugs or vulnerabilities, one can implement different countermeasures for the different forms of exploits. As always, extensive black box and white box testing, security audits are essential for ensuring a higher level of software security.

[8] see ch. 5.2.8
[9] see ch. 5.2.8.3
[10] see ch. 5.2.5

Flooding A problem that is getting increasingly more common is denial of service attacks by flooding the victim with data or connections, exhausting the available resources until the target service is no longer usable. According to CERT/CC[1], 44% of the registered attacks in 2003 were denial of service attacks. Although more or less impossible to stop at the end-point of the attack, the software should be written in such a manner that flooding attacks result in graceful degradation of the system.

Buffer overflows One of the largest problems in recent software security has been the exploitation of unprotected memory buffers - i.e. writing beyond the allocated size of the buffer. Although newer languages and environments eliminate these flaws, one can seldom afford writing software for smaller and embedded systems in these environments.

Logical Errors The classic kind of bug where flawed logic compromise security can also be the most expensive to solve. When dealing with parsers and evaluating algorithms, these must be sufficiently mathematically proven.

Chapter 6

Evaluation of model

6.1 Introduction

Although proposing a solution to many of the suggested requirements we set up in the beginning of this paper, many new problems and aspects were found during the continuation of the research. In this section we will summarise the issues with the suggested model.

6.2 Decentralisation

The largest difference between our suggested model and the currently used standards is our shift in location of control and logic. As opposed to most PKI solutions, this decentralisation allows for the user (client) to have more control over the data stored and presented to the different validating entities. As discussed under *Existing solutions*[1], the closest to a decentralised system examined is PGP, where communication and distribution is handled in an ad-hoc approach. Our suggestion keep the notion of authorities for certificates and modules, while the validation is done by a combination of modules with separate validation nodes.
Even though with a certain trade-off in complexity, decentralisation brings about a number of general improvements. Though many of these are of a theoretical nature, we identify a couple of benefits decentralisation brings our suggested solution.

6.2.1 Flexibility

Distributing the information control over more actors certainly brings about more administrative tasks for more actors, but nevertheless allows for more flexibility in the standards of data storage. More actors participating in the discussion and ongoing change would hopefully generate a more sane development of module standards.

6.2.2 Reliability

Although heavily implementation dependent, decentralisation will create the possibility for an improvement in fault tolerance. If validation is based on uncertain (or in some cases optional) information from third parties, this shift in logic will in effect lead to a more tolerant system.

6.2.3 Anonymity

User control over the information gives the option of separating your digital identity from your personal counterpart, allowing a user to separate his or her "logical" identities (i.e. work from

[1] see ch. 4

private, etc.) with more ease. There might be many legitimate reasons for one to want to make this separation, but unfortunately, most identity solutions of today require the connection between real life identity and the presented identity to exist.

6.3 Administration

Specified as a requirement[2], we identify the administrative issues of the proposed system as a major drawback. With more logic in a greater number of nodes, the configuration and maintenance of the system is dramatically increased. While this might be acceptable for a certificate, module or revocation authority, this is a much greater problem for the validating authority, not to mention the end user. With more control over the data comes the requirement of supervision. Even though much of the tasks can be facilitated by the client's software, thus letting information control be parallel to user knowledge, we still consider this to be a severe obstacle for the system to be widely accepted. One can only assume that a user that feel a single sign-on system to be to complicated has no interest in managing a more complex digital persona.

6.3.1 Recovery & renewal

Disregarding normal use of the persona, even more administrative tasks are required when a certificate (or module) has to be renewed. This is further more complicated if a certificate or module has to be revoked. This is, as previously mentioned, a major problem in traditional PKI solutions, but to an even farther extent problematic in our suggested model, in which more components can expire or be revoked by several entities.

6.4 Resource demanding

Also specified as a non-functional requirement[3], resource efficiency is important when it comes to embedded systems. Asymmetric encryption is in itself much more resource demanding than its symmetric counterpart. Although the suggested solution should be using a combination of the two to minimise the computational requirements, we consider both the validation and revocation checking to be rather resource intensive.

6.4.1 Validation

Decentralisation of logic requires even more computational power in each node. For a validating entity to evaluate the appropriate authorisation policy, both a stable, fast network and a sufficient computational power is required.

6.4.2 Revocation

Revocation and revocation checking is one of the major bottle necks in the proposed model in regards to resource efficiency. For each validation sequence, in addition to the certificate itself, every module required (directly or indirectly) ought to be checked for revocation. For this to be of ample speed, not only local computational power of the revocation checking entity is required. For every Revocation Authority RA, it is required that the link between the checking client and RA is stable and with low latency, in addition to the RA by itself is able to respond within a short period of time.

[2] see ch. 3.3.3

[3] see ch. 3

6.5 Comparison to existing solutions

	PGP	SSL	.NET	X.509	Kerberos	Liberty Alliance	Proposed model
Flexibility in user information	-	-	-	-	-	x	x
Simplicity for user	x	-	x	-	x	x	-
User "control"	x	-	-	-	-	x	x
Simple Key Management	-	x	-	-	-	-	x
Accepted Industry Standard	x	x	-	x	x	x	-
Scalability	-	x	x	x	x	x	x

Figure 6.1: Feature matrix of evaluted solutions

Relapsing to our feature matrix[4], we have now included the proposed model. Of the most important issues presented, two issues remain largely unresolved - Accepted industry standard and simplicity for the user. Even though industry standard is vital for such a scheme to survive, it is something that has to emerge with time. User simplicity, however, is slightly beyond our scope, and is an area of constant research. What can be said, however, is that the proposed system puts the user in control - something that will require a higher level of computer experience for the said user.

[4] see ch. 4.6.5

Chapter 7

Conclusion

This chapter serves as to relapse to the introduction, in which we presented four questions to be answered in this thesis.

1. What is the concept of identity in our domain?
2. What are the requirements for a system applying this concept?
3. How could a suitable model incorporating these requirements be defined?
4. How would this model work compared/contrasted to existing models/solutions?

What we sought to define when we started out was the basic concept of identity. In chapter *Identities*[1], we focused on the basic issues with identities and the parallels between traditional means of solving these problems and the means of doing so in the area of computer science. Answering the first question we stated that we intended to create a model for validating a user based on minimal information provided by the user in every scenario (Persona approach).

The requirements for a persona approach were evaluated and discussed in chapter *Requirements*[2], and in the following chapter[3], we compared existing solutions in accordance to the previously stated approach. These two chapters led to the presented feature matrix[4] of what we considered the solutions to offer in respect to the identified requirements, thus answering question 2.

In *Proposed model*[5], we presented the persona approach, a loosely coupled modular certificate model incorporating the forementioned requirements and discussed the implications of such a model, answering question 3. Considering it to be the most interesting part of the model, we chose to put our main focus on the validation scenarios in the presented model.

The persona approach was further evaluated in *Evaluation*[6], which was summarised by updating our feature matrix, connecting to the set requirements and analysis of other approaches. This chapter served to answer the forth and last question.

[1] see ch. 2
[2] see ch. 3
[3] see ch. 4
[4] see ch. 4.6.5
[5] see ch. 5
[6] see ch. 6

Chapter 8

Discussion

8.1 Introduction

Initially, we had great plans for how a modular approach to presenting a digital persona could be achieved. It was also quite easy to identify problems with existing solutions. The conceptual charm of user control over personal information, and the possibility to create a system where security and authorisation procedures do not necessarily interfere with personal integrity or even anonymity, felt both intriguing and feasible.

However, as we started to analyse the practical implications of such a system, several of the sought requirements fell, and many new-found issues appeared to contradict such an approach.

8.2 Theory into practise

8.2.1 Introduction

Putting this system in use would require a great amount of resources. The solutions that were examined in this paper have all evolved under many years (or in the case of SSL, even a decade).

8.2.2 Defined standard

In the digital identity/computer software area, there have been a couple of ways to establish a recognised standard. Traditionally, larger actors have developed their own systems, using their share of the market to force the new-developed standard onto it, competing with existing technologies. Techniques such as Microsoft's Passport and Netscape's SSL have been introduced in such a manner. In the case of SSL, the competition was small or non-existant, and SSL became one of the most widely accepted standards for securing network transactions. The IETF-supported TLS[29](In some contexts referred to as SSL v3.1) was created in 1999, based on SSL, to create an open industry standard five years after the first version of SSL was created.
Microsoft's Passport, on the other hand, was introduced in a time where competition was much higher, and as several security vulnerabilities were discovered and attacked, third party companies using Passport one by one abandoned the project. After the much noticed drop out of Ebay, Microsoft no longer actively support Passport usage in third party companies.

Newer enterprises, such as the Liberty Alliance, aim to establish standards by cooperation between the major actors, thus creating standards that can be accepted quicker and eliminate possible competitors. Backed up by industry giants such as IBM, Intel and Sun, such endeavours

have both the resources and the support needed for creating lasting standards.

This type of industry-wide support is what we consider essential for the suggested solution to be implemented and put into use. Without complete support by the major actors (governments and larger corporations) we consider it more or less impossible for this type of solution to ever reach a broad consumer base.

8.2.3 Implementation

In addition to the software solutions that have to be created, suitable hardware solutions should be analysed as well. Handheld devices capable of managing the certificates and modules, allowing the user to select appropriate means of authenticating in each and every scenario together with hardware entities for the validation front-end should also be investigated.

8.2.4 Testing & Maintenance

As always with security-related products, testing tends to become the focus of attention. One serious security flaw in the system could mean loosing the trust of the entire market. As mentioned[1], extensive tests and audits are crucial. Additionally, one should go to great lengths to assure the continuation of a reliable system.

8.2.5 Control contra ease of use

As previously stated[2], one major drawback with the proposed system is its demands for user administration. Unfortunately, we do not see any simple solution to the conflict between "user friendliness" (in the sense that most tasks are done without user work) and user control. What can be done is to give each user the option of control. However, realising that many of the users would not be in control of the very identification system they rely on would inevitably decrease the general trust of the system. As in so many other cases, the users become the weakest link in the system, once again creating a demand for a better system solving these problems. Could it be that PGP is the most elegant solution to authentication, with a trust system based on close, personal connections? Perhaps there is no way to really trust the "mass". Perhaps the zealously controlling, big brother tendencies growing in the U.S. and E.U. today are the only (or at least most efficient) way to really control the ways of a down-spiralling society. Orwell's telescreens[22] does not seem as far fetched after inspecting the European union's motions for internet service providers to log all traffic flowing through their net. The public does not seem to object to the new ways of government either, affected by the fear propaganda, eager to gain any "security" improvements even when forced to trade off privacy or integrity. Even though history ought to have taught us differently, we still seek to trade our freedom for security, our control for simplicity.

In the words of Benjamin Franklin:

> "Those who would give up essential Liberty, to purchase a little temporary Safety, deserve neither Liberty nor Safety"

To relapse, what one can offer is the option of user control. We believe that the proposed modular, decentralised approach can offer this.

We do not, however, consider it feasible that anyone would invest the enormous amount of resources required to develop this system, seeing as the largest pro with this approach is aimed for

[1] see ch. 5.4.6
[2] see ch. 6

the end-user, who, most probably, do not want it to the extent that he would use his consumer power to have it created. To summarise, even though certificate modularisation and decentralised validation may have a conceptual elegance that possibly charms the privacy eager among us, putting it into practise would probably be extremely costly and most likely futile.

8.3 Gained experiences

When we first set out to study the requirements and implications with a module based identity/persona scheme, we had planned to implement a reference solution. After putting many weeks of work into coding this proof of concept, we realised how little we could ever implement within the given time frame. Lots of work was put into coding framework, stub servers and GUI components that were required before even beginning to build the actual proof of concept system. Although not providing us with the sought concept proof, we grasped what a tremendously huge project it would be to put these plans into work, or even test them. To even think that one in 20 weeks could do something more than a mere pre-study seems rather foolish with this experience in mind.

8.4 Further work

To follow up this idea, we propose more work to be put into several fields:

8.4.1 Module languages for validation sequence

Perhaps one of the most interesting parts of the system is the evaluation of the presented data from an approaching client. We suggest the use of a language similar to, or built on XACML for defining rule sets on which a validating entity should base his authorisation procedure. A thorough study of suitable languages or how one should implement this support would be of great use to the development of the system.

8.4.2 Referential implementation

Obviously, for it to work or even evolve, some kind of proof of concept should be made. Although this would cost a lot of resources, it is vital for the development of the system that a referential implementation is created. This model should also be able to provide new topics of discussion, revealing new issues with the system. The parts that are most crucial to be implemented are:

- Certificate authority server
- Certificate handler (client)
- Revocation authority server
- Data storage (further development of a suitable data format, i.e. BTL)

8.4.3 Revocation schemes

Being one of the largest drawbacks of the proposed system, optimising revocation schemes would greatly improve the system. Although this field has been explored in numerous papers and research projects, one could examine what methods would suit this type of multiple sources revocation checking better.

8.4.4 Hardware solutions

For the system to be successful, it has to be as portable and easy to use as a traditional key. Several hardware tokens in the format of an USB stick or flash card that allow data storage combined with a small CPU are appearing on the consumer market. Investigating how one could solve PKI cryptography on these, in combination with the control that is required for the modular system would help to clarify how user interaction could be facilitated.

Chapter 9

Acknowledgments

The authors would like to thank the following individuals for their efforts to help in the writing of this thesis:

For his role as supervisor, his highly valued opinions, support and patience,
Professor **Rune Gustavsson** at the Blekinge Institute of Technology

For inspiration and ideas for technical solutions,
Mr **Ulrik Mikaelsson**

For their much appreciated opinions and comments on the thesis,
Mr **Adam Andersson**
Mr **Marcus Andersson**

Chapter 10

Glossary

- X.500 - Directory service specified by ITU
- Authentication - Validating that the presented identity is true
- Authenticate - Determine the identity
- Authorisation - Evaluating the access rights of the approaching identity
- CA - Certificate Authority
- CCA - Client's Certificate Authority
- CRA - Client's Revocation Authority
- CRL - Certificate Revocation List
- ITU - International Telecommunications Union
- MA - Module Authority
- MRA - Module Revocation Authority
- PAM - Pluggable Authentication Module
- PDP - Policy Decision Point
- PEP - Policy Enforcement Point
- Persona - The identity presented towards others
- PKI - Public Key Infrastructure
- RFC - Request For Comments
- Stakeholder - Service provider
- XACML -eXtensible Access Control Language

Appendix A

BTL Specification

Draft as of 2005-05-13 Author: Ulrik Mikaelsson

definition: BTL Document A BTL Document is built of a set of direct commands for the BTL parser. A command consists of 1 byte of instruction code, and a variable amount of parameters. A difference from XML is that a BTL stream (file or network-stream) may contain several root-nodes. The only commands allowed outside a root node are those of stage 0 (set encoding, amongst others)

definition: Parameters see Data.

definition: Data Data normally consists of two parts. One byte type-identifier, and a payload. There are however one exception, boolean, where the value can be read directly from the type-identifier. Optionally, when a certain type of data is expected in the format, the type-identifier may be absent, as specified in this format specification.

definition: Type-identifier A type identifier are simply one byte, prepending the actual parameter data, telling the type of the data. The list of types are arranged to be quick and simple to break down and parse. The list of valid types are:

	HEX	BINARY
boolean null	0x00	0000 0000
boolean false	0x01	0000 0001
boolean true	0x02	0000 0010
1 byte unsigned integer	0x10	0001 0000
2 byte unsigned integer	0x11	0001 0001
4 byte unsigned integer	0x12	0001 0010
8 byte unsigned integer	0x13	0001 0011 (optional)
16 byte unsigned integer	0x14	0001 0100 (optional)
32 byte unsigned integer	0x15	0001 0101 (optional)
varlength unsigned integer	0x17	0001 0111 (yet undefined and unsupported)
1 byte signed integer	0x18	0001 1000
2 byte signed integer	0x19	0001 1001
4 byte signed integer	0x1A	0001 1010
8 byte signed integer	0x1B	0001 1011 (optional)

16 byte signed integer	0x1C	0001 1100 (optional)
32 byte signed integer	0x1D	0001 1101 (optional)
varlength signed integer	0x1F	0001 1111 (yet undefined and unsupported)
binary data, max 2^ 8 length	0x20	0010 0000
binary data, max 2^ 16 length	0x21	0010 0001
binary data, max 2^ 32 length	0x22	0010 0010 (optional)
binary data, max 2^ 64 length	0x23	0010 0011 (optional)
binary varlength	0x27	0010 0111 (yet undefined and unsupported)
text, max 2^ 8 length	0x28	0010 1000
text, max 2^ 16 length	0x29	0010 1001
text, max 2^ 32 length	0x2A	0010 1010 (optional)
text, max 2^ 64 length	0x2B	0010 1011 (optional)
text, varlength	0x2F	0010 1111 (yet undefined and unsupported)
list, 1 byte length	0x30	0011 0000
list, 2 byte length	0x31	0011 0001
list, 4 byte length	0x32	0011 0010
list, 8 byte length	0x33	0011 0011 (optional)
list, 16 byte length	0x34	0011 0100 (optional)
list, 32 byte length	0x35	0011 0101 (optional)
list, varlength	0x37	0011 0111 (yet undefined and unsupported)
homogenous list, 1 byte length	0x40	0100 0000 (optional)
homogenous list, 2 byte length	0x41	0100 0001 (optional)
homogenous list, 4 byte length	0x42	0100 0010 (optional)
homogenous list, 8 byte length	0x43	0100 0011 (optional)
homogenous list, 16 byte length	0x44	0100 0100 (optional)
homogenous list, 32 byte length	0x45	0100 0101 (optional)
homogenous list, varlength	0x47	0100 0111 (yet undefined and unsupported)

definition: Integer There are various lengths of integers and each length may come signed or unsigned. One thing is however consistent for all integers. They are all given in network byte-order. The type-identifier will tell you the length of the integer, just parse it. The last bit of the type-id will be a 1 for all signed integers, and a zero for all unsigned integers.

definition: String and Binary A string is simply a binary chunk of data, that MUST be parsed using the current set encoding. Both strings and binary data begins with a one, two or four-byte long unsigned integer, specifying the length of the string or data. The number of of bytes telling the length of the string is given in the type-id of the parameter.

definition: Float (Floats are still undefined and should not be used.)

definition: List A list is an ordered collection of values. There is a difference between a list and a homogenous list. In a regular list, each item can be of a different type. In a homogenous list, every items is of the same type. In a homogenous list, the type-indicator of the list appears just after the list-length, otherwise, each item is type-indicator + value. A list is type-indicator + list-length + list_type(if homogenous list) + items.

definition: Encoding Encodings are given with a special command that instructs the parser to parse coming text segments using a new encoding. The new encoding are specified as a text-type parameter using the ASCII-encoding for the encoding abbreviated name. If NO encoding

is specified, the parser MUST assume UTF-8-encoding. Encodings are limited to the current environment. That means a close_node-command for this node resumes the old encoding. (List of valid abbreviations to come)

definition: Filters A filter is a routine for transforming the data-stream in some way. Filter can be stacked on top of each other, and the behaviour is always stack-like. (Push-Pop) Example uses of filters are encryption and compression.

Note that the use_filter command is not within the filtered-stream, but the stop_filter is. After the stop_filter command, the filter is closed, and the previous stream should be used for data. Filters are implicitly closed if the node in which they were introduced is closed. Filters MUST ALWAYS be closed in the same context as they were created. Thus, a filter may not be closed in a child node to the node where it was created.

definition: Monitors A monitor is a mechanism used for creating various verification routines of the data. Typical uses are checksumming, and cryptographic signatures. An active monitor is implicitly closed if the node in which it were introduced is closed, but there will be no way to verify the monitor, and the verification will fail if the verification needed some kind of data.

definition: Commands A command is given by a command-code, and optional parameters. Different commands have different accepted parameters, and may have a duration, which normally ends with the end of the node.

Every node has various stages after creation. For each stage, a different set of commands are allowed. Each command is encoded into two parts, the four three significant bits are stage-indicator, and the 5 remaining are command-code. There is ONE rule that must be followed, stage indicator may never be lower for the current command, except in two cases. Decreased stage inidicator IS allowed when the current stage-indicator is zero, OR when the previous command was close_node.

set_encoding
Command-code: 00 000001 (0x01)
Parameters:
encoding abbreviation: string (max 2^ 8), ASCII-encoding is ALWAYS used.
Occurrence: Optionally once.

use_filter
Command-code: 00 000010 (0x02)
Parameters:
filter-name: string (max 2^ 8), ASCII-encoding is ALWAYS used.
parameters: list, 1 byte length

stop_filter
Command-code: 00 000011 (0x03)
parameters: None

add_monitor
Command-code: 00 000100 (0x04)
Parameters:
monitor-name: string (max 2^ 8), ASCII-encoding is ALWAYS used.
monitor-id: string or integer
parameters: list, 1 byte length

pause_monitor
Command-code: 00 000101 (0x05)
Parameters:
monitor-id: string or integer

resume_monitor
Command-code: 00 000110 (0x06)
Parameters:
monitor-id: string or integer

close_monitor
Command-code: 00 000111 (0x07)
Parameters:
monitor-id: string or integer
verification-data: list, 1 byte length

set_name
Command-code: 01 000001 (0x41)
Parameters:
value: string (max 2^ 8) (without the type-descriptor-byte)
Occurrence: Required once per node.

set_id
Command-code: 01 000010 (0x42)
Parameters:
value: string or integer
Occurrence: Optinal once per node.

define_attribute
Command-code: 01 000010 (0x43)
Parameters:
name: type string (max 2^ 8) (without the type-descriptor-byte)
value: any type
Occurrence: Optional, unlimited times.

create_child
Command-code: 10 000001 (0x81)
Parameters: None
Occurrence: Optional, unlimited times.

add_data
Command-code: 10 000010 (0x82)
Parameters:
value: any type

close_node
Command-code: 11 111111 (0xFF)
Parameters: None
Occurrence: Once for each create_child

References

Books

[10] Jalal Feghhi. *Digital Certificates: Applied Internet Security*. Addison-Wesley, 1998. ISBN 0201309807.

[27] Aviel D. Rubin. *White-Hat Security Arsenal: Tackling the Threats*.

[28] William Stallings. *Network Security Essentials*. Prentice Hall, 2003. ISBN 0-13-035128-8.

[30] Philip R. Zimmermann. *The official PGP user's guide*. MIT Press, Cambridge, MA, USA, 1995. ISBN 0-262-74017-6.

Papers

[7] David A. Cooper. A model of certificate revocation. In *ACSAC*, pages 256–, 1999. URL citeseer.ist.psu.edu/cooper99model.html.

[11] Diego Gambetta. *Can We Trust Trust?*, chapter 13, pages 213–237. Basil Blackwell, 1988. URL citeseer.ist.psu.edu/gambetta88can.html. Reprinted in electronic edition from Department of Sociology, University of Oxford, chapter 13, pp. 213-237.

[13] E. Gerck. Overview of certification systems: X.509, ca, pgp and skip, 1998. URL citeseer.ist.psu.edu/252354.html.

[16] Hannu H. Kari. Internet is deteriorating and close to collapse - what we can do to survive?, 05 2004.

[20] Fabian Monrose and Aviel Rubin. Authentication via keystroke dynamics. In *CCS '97: Proceedings of the 4th ACM conference on Computer and communications security*, pages 48–56, New York, NY, USA, 1997. ACM Press. ISBN 0-89791-912-2. doi: http://doi.acm.org/10.1145/266420.266434.

[21] National Institute of Standards and Technology. Public key infrastructure study, final report, April 1994.

[24] Maja Pusara and Carla E. Brodley. User re-authentication via mouse movements. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 1–8, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-974-8. doi: http://doi.acm.org/10.1145/1029208.1029210.

[25] A. Rasmusson and S. Janson. Personal security assistance for secure internet commerce, 1996. URL citeseer.ist.psu.edu/rasmusson96personal.html.

[26] Ronald Rivest. Certificate-based authorization policy. *ACM Transactions on Information and System Security*, 6(4):566–588, November 2003.

Resources

[1] The cert coordination center. URL http://www.cert.org as of 2005-05-20.

[2] The internet engineering task force. URL http://www.ietf.org as of 2005-05-20.

[3] The liberty alliance project. URL http://www.projectliberty.org as of 2005-05-20.

[4] Kerberos: The network authentication protocol. URL http://web.mit.edu/kerberos/www as of 2005-05-20.

[5] Pam login tools. URL http://n.ethz.ch/student/mariost/pkcs11_login as of 2005-05-20.

[6] Microsoft passport to trouble. URL http://alive.znep.com/~marcs/passport/index.html as of 2005-05-20.

[8] O. Gudmundsson D. Eastlake. Rfc 2538 - storing certificates in the domain name system (dns), 03 1999. URL http://www.faqs.org/rfcs/rfc2538.html as of 2005-05-20.

[9] T. Howes F. Dawson. Rfc 2426 - vcard mime directory profile, 09 1998.
URL http://www.faqs.org/rfcs/rfc2426.html as of 2005-05-20.

[12] Gartner. Security flaw shows microsoft passport identities can’t be trusted.
URL http://www4.gartner.com/DisplayDocument?doc_cd=114948 as of 2005-05-20.

[14] Simon Godik and Tim Moses. extensible access control markup language (xacml). In *OASIS Standard*, February 2003. URL www.oasis-open.org/committees/xacml/
as of 2005-05-20.

[15] Carl G. Jung. *The Relations between the Ego and the Unconscious*. 1928. CW 7: Two Essays on Analytical Psychology.

[17] RSA Laboratories. Pkcs 12 v1.0: Personal information exchange syntax, 1999. URL http://mirror.switch.ch/ftp/doc/standard/pkcs/pkcs-12/pkcs-12v1.pdf as of 2005-05-20.

[18] Rebekah Lepro. Cardea: Dynamic access control in distributed systems. In *NAS Technical Report NAS-03-020*, November 2003.

[19] R. Ankney et al. M. Myers. X.509 internet public key infrastructure online certificate status protocol - ocsp, 06 1999. URL http://www.faqs.org/rfcs/rfc2560.html as of 2005-05-20.

[22] George Orwell. *1984*. 1948.

[23] George Ou. Proposed redesign of pki an affordable and scalable pki model, 09 2003.
URL http://www.lanarchitect.net/Articles/Cryptography/Proposed-PKI/ as of 2005-05-20.

[29] C. Allen T. Dierks. Rfc 2246 - the tls protocol version 1.0, 01 1999.
URL http://www.faqs.org/rfcs/rfc2246.html as of 2005-05-20.