Trust in open grid systems

Jesper Krogh

Kongens Lyngby 2005
IMM-M.Sc.-2005-89

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

Summary

The existing security infrastructures enables secure collaboration in systems where it is acceptable to have a single or few central authorities. In a distributed system with many independent collaborators like the internet this approach is not feasible.

This M.Sc. Thesis has focus on modelling a security infrastructure with features similar to the human notion of trust within a distributed computing network. A system that handles the evidence and recommendation evaluation on top of a public key infrastructure that enables an open and collaborative system with the subject in focus.

This report descibes the design and implementation of such system.

Keywords : Security, Encryption, Certificate, Trust, Evidence, Public Key Infrastructure.

Resumé

Eksisterende sikkerhed infrastrukturere tillader samarbejde i systemer hvor det er acceptablet at have en eller få centrale autoriteter. I et distribueret system med mange uafhængige deltagere, såsom internettet, er dette ikke længere muligt.

Dette eksamensprojekt fokuserer på at modellere en sikkerheds infrastruktur der har egenskaber lignede dem man kender fra det menneskelige tillidsbegreb i et distribueret computernetværk. Systemet håndterer evidens og anbefalinger ovenpå en kryptografisk infrastruktur der tillader et åbent system med individet i centrum.

Nøgleord : Sikkerhed, Kryptering, Certifikater, Tillid, Evidens

Preface

This thesis was prepared at Informatics Mathematical Modelling, the Technical University of Denmark in fulfillment of the requirements for acquiring the Master of Science in Engineering.

This project was carried out from september 1, 2004 to july 30, 2005 and was supervised by associate professor Christian D. Jensen, to whom I wish to thanks for patience, feedback, comments and support.

Lyngby, July 2005

Jesper Krogh

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Collaboration on the Internet is usually done in an anonymous manner, or when security is required, an security infrastructure like X.509 is deployed in an application specific manner.

> A grid system is often defined by the statement "a Grid is a system that coordinates resources, that are not subject to centralised control, using standard, open, general purpose protocols and interfaces to deliver a nontrivial quality of services."[14].

Grid and grid-like systems has been around for several years and are divided in two segments. The first segment is the distributed-computation networks like Distributed.net and Seti@home, which collaborates on very specific tasks with an enormous amount of computation required. The other segment is the scientific grid that likewise has a big amount of computation required. This segment has several other properties like; a significant amount of data to transfer in order to distribute the computations and needs to assure a higher level of reliability on the individual computing systems.

The distributed-computation networks connects a vast amount of resources. The infrastructures used lays several limitation on the types of resources that can be shared in the system. The end users cannot request resources on the grid and the computations has to be done several times in order to prove the correctness of the result. These systems accomplish collaborations between impressive amounts of entities in order to solve a specific task. The demand for security/confidentiality is very low or non-existing.

Scientific grid-systems usually deploy a tight security infrastructure for example the Scandinavia deployed Globus based grid - Nordugrid, where the security infrastructure is based on an X.509 public key infrastructure with Access Control Lists implemented on top of. The scientific grids are typically built from clusters of workstations specifically bought and allocated for the purpose. These grid-systems are typically limited to academic environments where trust in the end users is high and the demand for confidentiality is low.

1.1 Problem

The existing scientific grid infrastructures enables resource discovery and job-submission in various ways, but fails to deliver a security infrastructure that is flexible enough to allow collaboration between entities that are not within the same administrative domain or application framework.

Home users disk-systems approach the terabyte limit; processing power rises enormously and the bandwidth rise as well. The computational power required for large scale simulations could benefit from an infrastructure that would enable collaboration in a secure ad-hoc manner between the vast amount of different computer-systems connected to the Internet, ranging from Printers, PDAs, web cameras to supercomputers.

The existing grid infrastructure are not open for the collaboration and resource sharing in an optimal way. Home users can easily benefit from the grid-vision by allowing entities to share resources on demand in a secure and flexible manner.

Among systems and users that would benefit from a secure and flexible security infrastructure are:

- Home users - Could share disk-space, fulfilling the need for an offsite system backup.
- Home users - Could share processing-power to various organisations of their choice.
- Home users - Could once the applications are modified to take advantage of the system distribute heavy computations to several other computers.

- Academia - Some tasks are ”embarrassingly parallel” making them suitable for this distributed environment.

- Advanced resource sharing e.g. grant friends and friends friends(logical groups) access to my Home Theatre PC collection.

Building a flexible security infrastructure that can scale to the different use scenarios that eventually will appear in an open grid system would benefit almost anyone. The need for privacy needs to be thought into the system.

1.2 Using trust

The situations where collaboration is needed seems to be similar to the situations where we're facing collaboration in real life. Some resources has a strict access control list in order to collaborate, other types of collaborations takes place without any prior interaction of the parties collaborating.

The idea is to model a security system that has similar properties as the one used in the society around us, thereby being modelled on a “human notion of trust”.

In the real world is trust a natural thing when cooperation needs to take place. People generally evaluate lots of different things in order to make a trusting decision.

This report evaluates several certificate/trust models and suggests a solution that closer to the model that seem to work in the real world than any of the existing systems does. A security infrastructure that can facilitate distributed computing in a security infrastructure flexible and generic enough to satisfy any applications scenario is suggested.

1.3 Report

The next chapter summarises the state of the are within grid systems covering the security properties of the different types of grid systems. The general encryption and public key infrastructures are described. The state of art within trust and trust management systems are likewise described.

The second chapter gives an analysis of the technologies described and discusses the within the scope of the problem.

The next chapter describes the complete feature set that the system implement followed by the actual implementation and test.

At the end the actual results are summarised and a ideas for future work are described.

Chapter 2

State of the art

This chapter describes the current state of security infrastructures, public key infrastructure and usage of trust in distributed computing system like grid-systems.

The first section describes the state within grid and grid-like systems, the second section describes the general encryption mechanisms and public key infrastructures available and the last section describes the state of trust management systems.

2.1 Grid Systems

> The term ”Grid Computing” is often defined by the statement: ”a Grid is a system that coordinates resources, that are not subject to centralised control, using standard, open, general purpose protocols and interfaces to deliver a nontrivial quality of service”[14].

During the past decade several grid or grid-like systems have been developed. These systems can generally be split into different types of grid systems, the “first generation grid systems”, where application specific distributed computing infrastructure has been developed to solve one particular problem and the "scientific grid systems" that takes a more generic approach to the problem facilitating an application neutral infrastructure. Although the former will be discussed, focus will be on the latter. The scientific grid systems are services in the ordinary Internet system, which allows coordination of resources. This gives rise to some new problems, such as multiple administrative domains, new ways to collaborate and a vast uncertainty in the performance and reliability of the resources.

This chapter focuses primarily on administrative and security aspects of grid systems, secondarily on the resource selection problem.

2.1.1 First generation grid systems

Among the first generation grid system are Seti@home[1]. Seti@home utilises the raw computing power of home users by harvesting CPU-cycles to compute on the search for extraterrestrial intelligence. Distributed.net[2] tries to compute a cryptographic key of various bit-sizes. Folding@home[3] tries to use distributes computing to understand protein folding and related diseases.

Common for these first generation grid systems is that they are application specific and closed by nature, allowing only the centralised authority to consume resources on the system. The security problem is quite simple. Since all resources in the system are equal in terms of trust and the jobs are of low risk. There is no risk to security nor privacy. In order to confirm the results of computations the specific tasks are usually computed several times by different computers and results are compared.

The participants in these systems generally have no interest in participation, apart from a web-based high-score list or a reward for solving the challenge or the sake of the "good purpose". The entities behind the systems are generally large Universities, which enables them to leverage the real world trust in governmental initiatives. The coordinator generally do not express any trust in the participants.

These grid-like systems are only of historic interest, though they are still interesting due to the fact that they accomplish participation of ten- to hundred-thousand systems at the same time, which still today outnumbers the more generic grid systems.

[1] http://setiathome.ssl.berkeley.edu/
[2] http://www.distributed.net
[3] http://folding.stanford.edu/

2.1.2 Scientific grid systems

Scientific grid systems such as the Nordugrid[4] are fully functional. Nordugrid are deployed at many sites with currently over 5.000 CPUs totally connected. Grid systems like the Nordugrid system are typically based upon the Globus toolkit[5]. More specifically the Nordugrid's Advanced Resource Connector is based upon the Globus toolkit version 2.

2.1.2.1 Globus Toolkit

With regards to the security infrastructure, the Globus Toolkit 2 and 4 (the most deployed) looks similar, so only the former is described. The Globus Toolkit 4 is based on web-services instead, but the mechanisms are the same. For a fresh introduction to the Globus Toolkit 4 look at this article from IBM Developerworks.[6]

The Globus Toolkit is the most widely deployed grid toolkit. It consists of a set of protocols, libraries and applications that build upon the standard Internet protocols to facilitate sharing of resources.

The Globus Toolkit consists of several components, the Grid Index Information Service, GIIS, that is a central resource discovery service. The GIIS[5],[17, p.542] is materialised in a LDAP server, that aggregates the global information about resources together in multiple levels. When a client requests a resource, it queries the GIIS for available resources that matches the specifics of the job, then connects directly to the resource delivering a job-description RSL (xRSL, in Nordugrid context). RSL is the Resource Specification Language used within the Globus Framework.

The resource uses the Grid Security Infrastructure (GSI)[17, p.382], to resolve the identity of the person submitting the job. GSI takes care of evaluating the access control mechanisms in order to grant or deny access to the resource. If accepted the job is computed and while the job is running its status is reflected in the local Grid Resource Information Service. This enables the client to query the job status and thereby discover when a job has finished. There are several notification mechanisms available in the xRSL specification, but no direct call-back from the resource.

[4] http://www.nordugrid.org - Grid Research and Development collaboration
[5] http://www.globus.org - Globus - The Globus Alliance
[6] http://www-128.ibm.com/developerworks/grid/library/gr-gsi4intro/

Grid Security Infrastructure - GSI GSI forms the basis of all security mechanisms in the Globus Grid system. This system is based on a X.509 public key infrastructure[13]. One of the essential extensions to the X.509 PKI is the X.509 proxy certificate that recently has been accepted as an IETF standard[28]. The proxy certificate allows single sign-on and delegation by generating a new "proxy certificate". This certificate is signed using the original certificate and contains information stating what the proxy is allowed to do. It is then up to the certificate recipient to honour the proxy to the extend that the certified information allows. The proxy certificate needs the original certificate to be validated. The typical proxy certificate usage is that a grid job submitted to a resource needs access to a storage element containing the data needed for the computation. The storage elements restricts access to the data to the owner but given the proxy certificate is it possible for the client to transfer access to the data to the particular system where job has been assigned to. Single sign-on is done by delegating the full proxy roles to the to the local grid-client framework, thereby enabling the client to sign-on transparently to the user.

Virtual organisations The concept of Virtual Organisations (VO) have been developed to facilitate cross organisational collaboration within the grid. By definition, the participants in the Virtual Organisation do not need to use the same authentication framework, but they needs to provide a mechanism to translate authentication requests to and from the framework in order to get a common authentication mechanism within the hole Virtual Organisation.[27, p.20].

2.1.3 Open Grid Services Architecture - OGSA

OGSA has undergone a large standardisation work in the Grid forum[7] and seems to be where the current grid focus is.

In the web-service oriented world, every component consists of a service, thereby in OGSA the security is split out as services within the framework described by of[12][17, p.370]: An Authentication service, that is responsible of verifying an identity. e.g. user-name/password validation. The second service is the identity mapping service that maps the identities cross-organisational domains e.g. for X.509 using the complete triple consisting of Subject Distinguished name, Issuer distinguished name and serial number. (Note: The authentication service needs to have access to all certificates to verify the signatures). The third service is the Authorisation service, that takes care of validation the identity against a particular authorisation policy, a VO Policy service, which takes care of policies

[7] http://www.gridforum.org

regarding virtual organisations, a credential conversion service, that takes care of converting between different credentials, an audit service to secure that a given policy is enforced, a profile service, that can provide additional attributes to a given identity and a privacy service that can enforce a privacy policy.

An extension to OGSA, called OGSI, Open Grid Services Infrastructure aims to provide state-full web-services instead of the usual stateless nature of the web-service world.

2.2 Encryption and Public key infrastructures

This section describes encryption and public key infrastructures.

2.2.1 Encryption

Encryption is divided into two separate mechanisms, symmetric encryption also known as secret key encryption and asymmetric encryption also known as Public key encryption. Encrypted text is called cipher-text (denoted by C) and not encrypted text is called plain-text (denoted by P).

2.2.1.1 Symmetric Encryption

Symmetric encryption algorithms like DES, 3-DES and AES allows the transmission of data securely over insecure networks like the Internet. The transmitting and receiving entity need to share the same key.

The encryption is called symmetric due to the principle:

$$E(k, P) = C \tag{2.1}$$

$$D(k, C) = P \tag{2.2}$$

Where k is the shared key, P is the plain-text and C is the cipher-text. E is the encryption algorithm and D is the decryption algorithm.

Symmetric encryption is generally fast, but the problem of distributing the shared secret key makes it difficult to use the symmetric encryption in an application alone.

2.2.1.2 Asymmetric Encryption

Asymmetric encryption algorithms are algorithm where two keys are used to enable encryption and signing. The most common used algorithm is RSA, which was the first that could be used to both things, where the original DSA algorithm could only be used for signing.

Public key cryptography simplifies the generic key distribution problem to a public key distribution problem. Where n hosts communicating securely with each other using only symmetric encryption would require: $\frac{n^2-n}{2}$ keys, public key encryption reduces this to n keys, that can be used to exchange an application/situation specific symmetric key. Usually the public/private key pair are not used directly since the operations are computationally expensive compared to the symmetric encryption. Thus the normal process is to generate a symmetric key, distribute it using the public key and use the symmetric key for further communication.

Asymmetric encryption is formally described like this[22, p.73]

$$E(k_{pub}, P) = C \tag{2.3}$$
$$D(k_{priv}, C) = P \tag{2.4}$$

Thereby

$$P = D(k_{priv}, E(k_{pub}, P)) \tag{2.5}$$

Where E is the encryption function, D is the decryption function, E_{priv} and E_{pub} is the corresponding private/public key-pair.

Public key encryption, allows distribution of the public part of the key, without compromising the security of the system. Thereby is it possible to securely encrypt data to anyone who has a publicly known key available.

The digital signature functions works this way:

$$E(k_{priv}, P) = C \tag{2.6}$$
$$D(k_{pub}, C) = P \tag{2.7}$$

Compared to the encryption, the only difference is that the private/public keys are switched. This enables a key owner to produce a "signature" C of a data block P and produce a piece of data, that can only be extracted using the corresponding public key. Thereby it is verified that the data was produced by the entity that have access to the corresponding private key.

Since the operations are computationally expensive, they are usually implemented where the plain-text is a cryptographical hash of the plain text, thus the signed material are of a much smaller size.

2.2.1.3 Cryptographic hash functions

Cryptographic hash functions are a vital component to many public key infrastructures. Cryptographic hash functions are functions that take input of arbitrary length and output a single digest value based, with the properties: A "small" change in input guarantees a change in the digest. It is hard/difficult to find different input that computes the same digest-value.

Widely known cryptographic hash functions are: MD5, SHA-1[30], SHA-256, where there recently has been found weaknesses in the first two mentioned using cryptanalysis. This doesn't render them broken, but generally makes it easier to compute a hash-collision on hash-values computed based on them. Examples of the usage of hash function are in the next section describing digital certificates.

$$H(data) = digest \tag{2.8}$$

2.2.1.4 Digital Certificates

Digital certificate is a vital component of a Public Key Infrastructure. Using the asymmetric encryption/digital signature algorithm, the original key distribution problem has been reduced to a n-keys for n-collaborators key distribution problem. The problem is to address confidence in the public keys distributed. Certificates are typically used to bind properties to public keys; thereby making others (issuer from now on) assert that certain properties are bound to the public key.

The cryptographic operations used in this assertion are alike across the different public key infrastructure. The data is coupled with the corresponding public key and the digital signature generated by the issuer, thereby given the hash function H, the subject public key. $PK_{subject}$ the issuer private key PRK_{issuer} and the data D. A certificate is typically composed of the following fields:

$$\text{Certificate} \quad = \quad E(PRK_{issuer}, H(Data + PK_{subject})) + Data + PK_{subject}$$

A hash of the certificate attributes and the public key of the subject, signed with the issuers private key. The certificate is often distributed with the public key and the certificate data. The certificate can then be verifies using the public key of the issuer using equation 2.7. After the signature has been verified the hash is recomputed form the data contained in the certificate and compared to the signed data.

The data usually contains meta-data about the certificate, like expiry-date, issuing date, certificate revocation list, issuer public key or hash of the same, but these things change slightly from PKI to PKI and depend on what the issuer actually tries to assert in the certificate.

2.2.2 Public Key Infrastructures

This section describes various Public Key Infrastructures (PKI), with focus in the assertion and assumptions done in order to use the in a functional environment. The basic cryptographic functions are described in the earlier chapters. Public Key Infrastructure are specific ways to issue certificates in a manner that gives the certificates properties related to the system they're used in.

2.2.2.1 PGP - Web of trust

Pretty Good Privacy was one of the first encryption tools available to the public on the Internet. PGP was originally designed as a human rights tool. PGP focuses on solving the public key distribution problem for using encryption in email[9] but has a few other application scenarios. PGP is one of the most widely adopted public key infrastructures.

The PGP system tries to address the identity problem on the Internet, by setting up a web of trust that collectively make it harder to fake identity. The certification path shown in figure 2.1 where the arrow illustrates that A signed the public key of B, C and D, who all have signed the public key of E. A signature in the PGP system states that the person who issued the signature has verified that the identity information in the certificate corresponds to a real-world identity with the same properties. Every entity has connected a set of

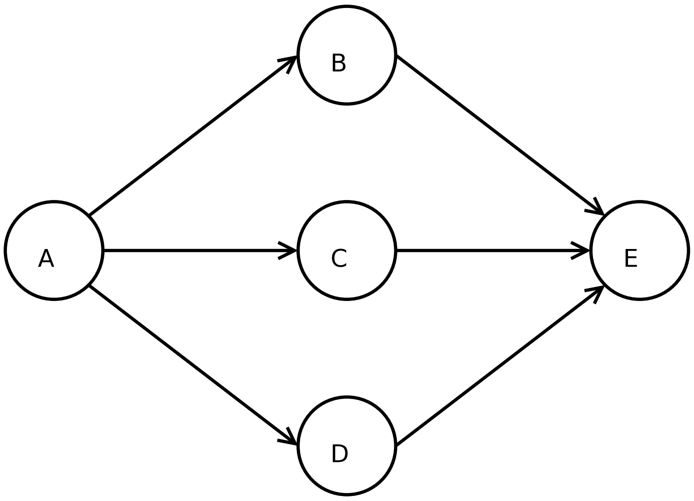

Figure 2.1: Certification path in PGP with 5 entities.

email-addresses to the certificate and the signature issued are a combination of the email address, the issuers public-key, the certificate public key and the “administrative bits”, such as issue and expire date.

Every entity tells the local certificate evaluation system (PGP-keyring), how much trust there should be set in an identity (Owner Trust). This weight is used to compute how far the web-of-trust reaches. Only a single or few signatures in this system makes it possible to get a valid certification path to anyone in the PGP-universe. Though this doesn’t mean that any trust should be addressed in the certificate at the end of the path, since it is an open system, that by concept allows anyone to generate certificates and issue signatures to anyone. Even without their knowledge.

The certification paths can become arbitrarily long, so key servers where everyone store their public keys are a central part of the PGP system. The process of certification is typically:

```
1: Generate key and upload it to a key server (local PC)
2: Meet the person(B) to verify the identity within the
   certificate and its fingerprint.(Somewhere else)
3: Fetch B's public-key from the key server.
4: Verify the fingerprint with the one exchanged at 2.
5: Sign the key and upload it to the key server.
```

As can be seen in the process the only thing stated in a signature is that the identity was assumed correct when issuing a signature. The signature reflects trust in that the certificate belongs to a particular person(email-address), bound by the name stated in the certificate.

When the certificate is first uploaded to the key server, it is self-signed certificate. Self-signed means that the only signature attached to the certificate is one that can be validated using the certificate itself. When the person uploads the new signature to the key server, the key server will make a lookup and find the original copy of the key and merge the signature into that certificate. This enables the other persons that fetches the certificates to see which others have signed the certificate.

The certification path illustrated in figure 2.1 show the situation where A has full trust in E, even though it has never personally verified the identity of E. This situation assumes that A has not altered the default trust-level of B, C and D. The figure shows the evaluation where 3 different signature-paths together gives a fully trusted identity. (Note: This can be altered in various ways that are optional to the identity doing the trust-evaluation).

2.2.2.2 X.509 - PKI

The X.509 Public Key Certificates were originally invented to assign public keys to X.500 Distinguished Names. The X.500 Distinguished Names would form a global directory of all computers, entities, etc. The X.509 public key infrastructure is a strict hierarchical certificate path from the Certificate Authority (CA) at the top level. The certificate of this “root” needs to be accepted by everyone who should verify certificates in the system. The certificate authority has a certificating policy that describes which demands the certificate owners have fulfilled in order to get an certificate. This CA could distribute the certification work to lower level CA’s(RA) to make a complete security infrastructure, where everyone with one or few root-certificates could verify all others certificates.

In order for this to be functional, all certificate authorities in the system must be failure proof and have the same level of assuring that the certificate policy is satisfied. Thereby is it possible but very unlikely that a X.509 PKI has several Certificate Authorities.

The IETF has put much work in the standardisation of the X.509 PKI for use within the Internet. Usage within the most widely used protocols like S/MIME (email) and HTTPS (web) is very common. The X.509 PKI is an identity based PKI where the goal is to bind identities to public keys. The components of a

X.509 PKI are[16, p.7]:

- Certification Authorities (CA)
- Registration Authorities (RA)
- End users - Subject or Certificate Holder.
- Certificate and CRL repository
- CRL issuer.

The Certificate Authority holds and controls the root-certificate, thereby has access to the corresponding private key. This enables the CA to issue certificates to Registration Authorities that can issue certificates to end-users. It is up to the Certificate Authority to ensure that the Registration Authorities full fill a given Certification Policy guaranteed by the Certificate Authority. The Certificate Authority issues the Certificate Revocation List, which makes it possible to retract certificates. The location of the certificate revocation list written directly in the root-certificate that everyone needs access to in order to verify certificates. This is a feasible approach as long as the CA's is few and the root certificate can be safely distributed or fingerprints of the certificates printed in public places.

The root-certificate is then resolved using the issuers distinguished names or certificate fingerprint written in the certificate in the certificate.

Since the X.500 naming structure doesn't exist, the naming of certificate entities is thereby let entirely to the Certificate Authorities. In the ideal world this PKI wouldn't give rise to any issues, it would be easy to verify the identity of anyone. But there is no central authority and it seems unlikely there should come one that everyone could agree to trust. Even if there could be an agreement it would lead to a central point of failure.

Trust in the X.509 PKI is given by definition, there is no way to express different trust levels in different entities within the system. The belief is that the Certificate Authority wont fail and thereby is all issued certificates fully valid regarding to identification of entities. This gives a natural high barrier in order to get a certificate issued by a certificate authority, which then again gives rise to many different certificate authorities with different certification policies.

2.2.2.3 SDSI/SPKI

Simple Distributed Security Infrastructure/Simple Public Key Infrastructure (SPKI) is described in [10][11]. This infrastructure addresses the complexity of the X.509 security infrastructure. The motivation was to get away from the central repository of X.500 and the central certificate authorities that forms the basis of X.509. The "Walton's Mountain assumption" describes the problem:

> Walton's Mountain Assumption - that if you know a name for someone, you know their identity (all their defining characteristics) - and that if you know someone's identity, then you know whether they're authorised to do or have something they request.
>
> Quoted from the SPKI homepage[8].

The SPKI system addresses this by a public key infrastructure that assumes no global knowledge, but uses local knowledge to address the key-exchange problem. The SPKI system defines an entity by it's public key, which is globally unique. The entity is the person who has access to use the corresponding private key. SPKI defines several different certificate types. An example of a SPKI full certificate would look like this[9];

```
(sequence
 (public-key
  (rsa-pkcs1-md5
   (e #11#)
   (n
    |ALNdAXftavTBG2zHV7BEV59gntNlxtJYqfWIi2kTcFIgIPSjKlHleyi9s
    5dDcQbVNMzjRjF+z8TrICEn9Msy0vXBO0WYRtw/7aH2WAZx+x8erOWR+yn
    1CTRLS/68IWB6Wc1x8hiPycMbiICAbSYjHC/ghq2mwCZO7VQXJENzYr45|
    )))
 (do hash md5)
 (cert
  (issuer (hash md5 |+gbUgUltGysNgewRwu/3hQ==|))
  (subject
   (keyholder (hash md5 |+gbUgUltGysNgewRwu/3hQ==|)))
  (tag
   (* set
    (name "Carl M. Ellison")
```

8 http://world.std.com/~cme/html/spki.html - SPKI homepage

9 http://theworld.com/~cme/examples.txt - SPKI Examples

```
      (street "207 Grindall St.")
      (city "Baltimore MD")
      (zip "21230-4103")))
    (not-after "1998-04-15_00:00:00"))
   (signature
    (hash md5 |54LeOBILOUpskE5xRTSmmA==|)
    (hash md5 |+gbUgUltGysNgewRwu/3hQ==|)
    |HU6ptoaEd7v4rTKBiRrpJBqDKWX9fBfLY/MeHyJRryS8iA34+nixf+8Yh/
    buBin9xgcu1lIZ3Gu9UPLnu5bSbiJGDXwKlOuhTRG+lolZWHaAd5YnqmV9h
    Khws7UM4KoenAhfouKshc8Wgb3RmMepi6t80Arcc6vIuAF4PCP+zxc=|)
   )
  )
```

This SPKI certificaet defines a full S-expression sequence. It instructs that the name Carl Ellison.. etc. is certified by the key identified by the md5 hash given by the issuer. In this example is it the same as the key holder and as can seen in the second value of the signature, the same key that has been used to generate the signature of the certificate.

The SPKI is optimised for small systems, which is why the (do hash md5) instruction is in the certificate. This instructs the system that reads the sequence, that the block read just before should be hashed for use later on when parsing the sequence.

Every public key defines its own name-space, allowing local names to be assigned to keys:

```
(name (hash sha1 |TLCgPLFlGTzgUbcaYLW8kGTEnUk=|) jim therese)
```

This entity refers to jim, defined by the public-key given by the sha1-hash and in the name-space of this key addressing therese. The chaining in the SPKI makes the system extremely flexible, and thus allows SPKI to adress many use cases.

Signature verification in SPKI can be a hard issue given that the certificates not necessarily carry enough information to be verified. An analysis of the verification process of SPKI is described in "Certificate Chain Discovery in SPKI/SDSI"[4], stating that this problem actually can be solved in reasonable time, which seems to be the largest issue raised about SPKI.

SPKI is a simple, yet powerful PKI, that has not been adopted widely yet. It aims at being able to map any kind of certificate in the real world to a digital

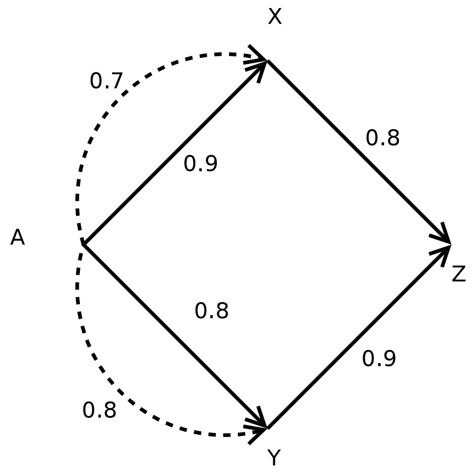

Figure 2.2: Probabilistic PKI with 4 entities. The entity A is expressing indirectly trust in Z

certificate[10]: health insurance card, prescription drug card, driver's licence (for permission to drive), driver's licence (for permission to buy alcohol),...

The SPKI seems to be the most generic PKI available, although adoption seems rare or even non-existing.

2.2.2.4 Probablistik Public Key Infrastructure

Where X.509 and SPKI addresses certifications with a single certification path, Ueli Maurer[21] suggests a public key infrastructure with multiple certification paths, inspired by the PGP system. The idea is that if A certifies the public keys of X and Y, X and Y both certifies B, given the information A can to some degree directly express trust in B.

In figure 2.2 the dotted lines reflect the amount of "confidence parameters" that A has assigned to the Trust in X and Y to certify certificates, where the full lines reflect the "authenticity parameters" A has in X and Y respectively and the "authenticity parameters Y and X has in Z. These values are then computed to a complete trust of 0.80 as the confidentiality of "authenticity" A has in Z to 0.8076 [21, Example 4].

Calculation of the confidence values are done using the formula;

$$
\begin{aligned}
conf(Auth_{A,Z}) \;=\; & \sum_{i=1}^{k} P(V_i \subseteq View_A) && (2.9)\\
& - \sum_{1 \leq i_1 \leq i_2 k} P(V_{i_1} \cup V_{i_2}) \subseteq View_A) && (2.10)\\
& + \sum_{1 \leq i_1 \leq i_2 \leq i_3 \leq k} P(V_{i_1} \cup V_{i_2} \cup V_{i_3}) \subseteq View_A) && (2.11)\\
& - \ldots && (2.12)
\end{aligned}
$$

So given the graph in the figure 2.2 the overall confidential value is calculated to

$$
\begin{aligned}
conf(Auth_{A,Z}) &= P((V_1 \subseteq View_A)) \vee (V_2 \subseteq View_A)) && (2.13)\\
&= P(V_1 \subseteq View_A) + P(V_2 \subseteq View_A) && (2.14)\\
&\quad -P((V_1 \subseteq View_A) \cup (V_2 \subseteq View_A)) && (2.15)\\
&= 0.9 \cdot 0.8 \cdot 0.7 + 0.8 \cdot 0.85 \cdot 0.9 && (2.16)\\
&\quad -(0.9 \cdot 0.8 \cdot 0.7) \cdot (0.8 \cdot 0.85 \cdot 0.9) && (2.17)\\
&= 0.8075520 && (2.18)
\end{aligned}
$$

$View_A$ it the set of all disjoint paths from A to Z that are aggregated together using the inclusion exclusion principle in order to avoid double calculation of information in the structure.

Ueli Mauers approach to aggregate trust and confidence values together take no notion of context into the perspective and assumes that every entity in the system has public specified the “confidence parameters” in the certificate. The assumption is that all entities are able to interpret the parameters in the same way.

2.3 Trust management

Trust management is defined by Blaze, Feigenbaum and Lacy as:

> "Trust management .. is a unified approach to specifying and interpreting security policies, credentials, and relationships; it allows direct authorisation of security-critical actions. A trust-management system provides standard, general-purpose mechanisms for specifying application security policies and credentials."[1]

2.3.1 Definitions of trust

Getting a notion of trust as computational concept is nessesary in order to make use of it within computing. The following sections forms some theory about how trust can be understood. This makes it possible to measure "trust" and "risk".

2.3.1.1 Morton Deutcsch

Morton Deutsch [6] formed this definition of trust back in 1962, which states:

> (a) the individual is confronted with an ambiguous path, a path that can lead to an event perceived to be beneficial (Va+) or to an event perceived to be harmful(Va-);
>
> (b) he perceives that the occurrence of Va+ or Va- is contingent on the behaviour of another person; and
>
> (c) he perceives the strength of Va- to be greater than the strength of Va+
>
> If he chooses to take an ambiguous path with such properties, I shall say that he makes a trusting choice; if he chooses not to take the path, he makes a distrustful choice.

This seems accepted as a definition of trust and is widely referenced in literature. It makes it clear that trust is by all means a subjective thing, reasoning about the perceived behaviour of another entity. Va- is the Negative impact and Va+ is the positive impact of an action.

Stating that if we wish to be considered trusted, then we need to act trustworthy, thereby producing evidence that we're trustworthy.

The information of Morton Deutsch's work is from [20, p. 25].

Description	Representation	Value Range
Situations	α,β,...	
Agents	a,b,c,...	
Set of agents	$\mathcal{A}$	
Societies of agents	$S_1,\ S_2,\ldots,\ S_x \in A$	
Knowledge (e.g., x knows y at time t)	$K_x(y)^t$	True/False
Importance (e.g., of α to x at t)	$I_x(\alpha)^t$	$[0,+1]$
Utility (e.g. of α to x at t)	$U_x(\alpha)^t$	$[-1,+1]$
Basic trust (e.g. of x at t	T_x^t	$[-1,+1]$
General Trust (e.g., of x in y at t)	$T_x(y)^t$	$[-1,+1]$
Situational Trust (e.g., of x in y for α at t)	$T_x(y,\alpha)^t$	$[-1,+1]$

Table 2.1: Notation from Formalising Trust as computation concept [20, p. 60]

2.3.1.2 Formalising trust as a computational concept

Stephen Paul Marsh impressive work on trust "Formalising trust as a computational concept"[20] identifies 3 different aspects of trust. Basic trust, general trust and situational-trust.

The notation used by Marsh is defined in table 2.1, it furthermore used the following assumptions [20, p.61]

- That x has a choice - he can say no if he wishes.
- That there is someone (e.g., y) to cooperate with in this situation.
- That x has a knowledge of y ($K_x(y)$ is true)
- That x is not "in debt" to y, This rules out any consideration that x helps to reciprocate for something y has done in the past.
- That x has a knowledge of the situation, in other words, that x perceives similarities between this situation and others he has experienced.

The three different types of trust are:

- **Basic trust** is defined just as a simple trust-value, that specifies the general trusting behaviour of the agent. It values from -1 to 1 where 1 would represent that the agent generally trust everyone fully whereas -1 would represent that the agent generally distrusts everyone.

- **General trust** is trust in a given agent, thus; given two agents x, y *in* $\mathcal{A}$ the trust that x has in y is general trust. General trust is situation independent.
- **Situational trust** is the last trust aspect, with takes the situation into the formula.

Furthermore a notion of Utility is defined $U_x(\alpha)$ to take into account how much utility x gains from interacting with α and a notion of knowledge is defined.

Marsh is aware that the the choice of the range from -1 to 1 leads to problems when they should be interpreted. Likewise the total distrust has the same issues when interpreted, given that you don't have evidence of all agents in the universe, saying that an entity has -1 gives that no-one can be distrusted more than this entity.[20, p. 57]

Marsh defines the situational trust by this equation:

$$T_x(y,\alpha) \quad = \quad U_x(\alpha) \times I_x(\alpha) \times \widehat{T_x(y)} \tag{2.19}$$

Where $\widehat{T_x(y)}$ represents the trust that x has in y given all evidence is taken into account. The multiplication operator gives several problems in this equation, since the utility and importance both could be -1 giving a sum of 1. Likewise the value 0 in any of the values would yield 0 as the overall trust value no matter what the other trust values are.

The corporation will then be possible when this equation is satisfied:

$$T_x(y,\alpha) \quad > \quad Coop\_Threshold_x(\alpha) \tag{2.20}$$
$$Coop\_Threshold_x(\alpha) \quad = \quad \frac{PerceivedRisk_x(\alpha)}{PerceivedCompetence_x(y,\alpha)} \times I_x(\alpha) \tag{2.21}$$

Which states that the cooperation will take place if the situational trust in is higher than the Risk multiplied with the importance divided by the competence. The perceived competence would arise from other types of evidence. It could be that the agent has been certified by some entity. Or acts under legal structures that can assure that distrustful behaviour can be tracked down.

Marsh furthermore describes principles of trust[20, Chap. 6.] where the most important is that:

- Trust it self-reinforcing. That when a certain trust-level has been reached and cooperation will take place, this cooperation will generate more evidence that further raises the trust.

- Trust is not transitive but trust can be compared it the agent is rational.

2.3.2 Policymaker

Policymaker is the first system to deal with trust management. Policymaker tries to encompass the full trust management problem. The principles in policymaker are(quoted from [2]):

- Unified mechanism, Policies, credentials an trust relationships are expressed as programs.

- Flexibility: The system is expressive rich enough to support the complex trust relationship that can occur in very large-scale network applications currently being developed.

- Locality of control: Each party in the network can decide in each circumstance whether to accept the credentials presented by a second party, on which third party it should rely for the appropriate “certificate”.

- Separation of mechanism from policy: The mechanism for verifying credentials does not depend on the credentials themselves or the semantics of the application that use them.

Policymaker gives a strong language to express policies and assertion in. The Using these principles a language for specifying policy assertions is proposed. Queries are of the form:

$$key_1, key_2, \cdots, key_n \textbf{ REQUESTS } \textit{ActionString}$$

Which represents a request to the policymaker system the set of keys is allowed perform a particular action here denoted by ActionString.

The other component is assertions, that are instructions to the policymaker system that processes the actual request.

$$\textit{Source} \textbf{ ASSERTS } \textit{AuthorityStruct} \textbf{ WHERE } \textit{filter}$$

An email application example is given where the assertion could look like this:

> policy **ASSERTS** pgp:“0x123456789abcdefa0b1” **WHERE** PREDICATE=regexp:“Organization: Bob Labs;

Which instructs the policy engine to assert that only mail signed with the PGP-keyid described by the fingerprint would be allowed to send mail where the organisation is “Bob Labs”.

The corresponding query could then look like:

> pgp:“0x123456789abcdefa0b1” REQUESTS “Organization: Bob Labs”;

The assertion is linked directly to a public key but unlike the X.509 PKI system, several levels of indirection can be addressed..

It is up to the application to construct the query from the actual mail message and verify that the digital signature on the message is valid. The policymaker engine only validates the language against queries.

2.3.3 SECURE

SECURE[10] (Secure Environment for Collaboration among Ubiquitous Roaming Entities) is a research project that has developed a model for trust. The foundation is reasoning about the human notion of trust that leads to development of a security model that allows entities to reason about risk when entering collaboration.

The SECURE framework focused on managing the trust life-cycle, evidence gathering, decision making and entity recognition, thereby being a complete framework for trust-management.

2.3.3.1 Entity recognition

Entity recognition in SECURE based on Pluggable Recognition Modules, which effectively puts the entity recognition issue outside the SECURE framework.

[10] http://secure.dsg.cs.tcd.ie/

Authentication Process	Entity Recognition
A.1: Enrolment: generally involves an administrator or human intervention.	
A.2: Triggering: e.g. someone clicks on a Web link to a resource that requires authentication to be downloaded	E.1 Triggering (passive and active sense): mainly triggering (as in A.2), with the idea that the recognising entity can trigger itself
A.3: Detective work: the main task is to verify that the principals identity is the peers	E.2 Detective work: to recognise the entity to-be recognised using the negotiated and available recognition scheme(s)
	E.3: Retention (optional): "preservation of the after effects of experience and learning that makes recall or recognition possible"
A.4: Action: The identification is subsequently used in some ways. Actually the claim of the identity may be done in steps 2 or 3 depending on the authentication solution (loop to A.2)	E.4 Action (optional): the outcome of the recognition is subsequently used in some ways (loop to E.1)

Table 2.2: Authentication and entity recognition

This entity recognition process should supply a level of confidence in the recognised entity to the SECURE framework. In "SECURE - Entity Recognition Scheme"[25] the entity recognition process is suggested. Shown in table: 2.2 (from [25, p.7])

This entity recognition scheme makes the normal (A.1) enrolment process superfluous thereby making it possible to dynamically enrol entities into the systems. In this context a peer recognition scheme APER[25, p.16] has been suggested. The recognition scheme is bases on broadcast of claims in ubiquitous environments. The scheme ends by describing 3 levels of recognition namely. These are listed in table 2.3

Claims are digitally signed broadcast on the network, that are collected by every peer that would wish collaboration. The "last-n-hashes" mechanism. The claimant provides a hash of the last seen claims, with its own claim. This would reflect a recent state of the network and thereby giving the claim a "freshness" notion. The last level is a direct challenge.

Level	Description
Level 1	Claimants signature verified on recently seen claims.
Level 2	Level 1 and claimants recent claims are "fresh", based on the "last-n-hashes" mechanism.
Level 3	Level 2 and a successful challenge.

Table 2.3: APER levels

2.3.3.2 Trust model

The trust model is complex and consists of several components. The fact that every interaction has an outcome this outcome is modelled by an event structure. This is done based on the fact that an action can have several outcomes at the same time. This enables the system to model the information a principal can have about an interaction.[3]. The example used is is an e-purse, where a purchase has the outcomes: good service g, response r, and correct c as outcome of an e-purse purchase. The outcome has the opposite possibilities too. Some outcomes are conflicting like no-response and correct, no-response and good-service. The event structure effectively models all possible outcomes of an action.

Every outcome is modelled by a triple of integers, which denotes contradicting, uncertain and evidence in favour from an outcome. Bases on all event-structures the trust-value that evaluations a complete interaction-history (set of event-structures from earlier events) into a trust value.

This model explicitly represents uncertainty and takes a generic approach to modelling trust information.

2.3.3.3 Overall view

The overall view of the SECURE frameworks decision-making process can be seen in figure 2.3. The Trust calculator is based on information monitored from earlier interactions. This contains the event-structures mentioned in the former section. The Trust calculator takes the principal recognised as input and probably some other parameter. The risk evaluator contains a set of probability density functions for each outcome, these informations get evaluated into a combined probability density function, for further evaluation by the request analyser. This component weights the possible outcomes and likelihoods against each other and make a decision based on that.

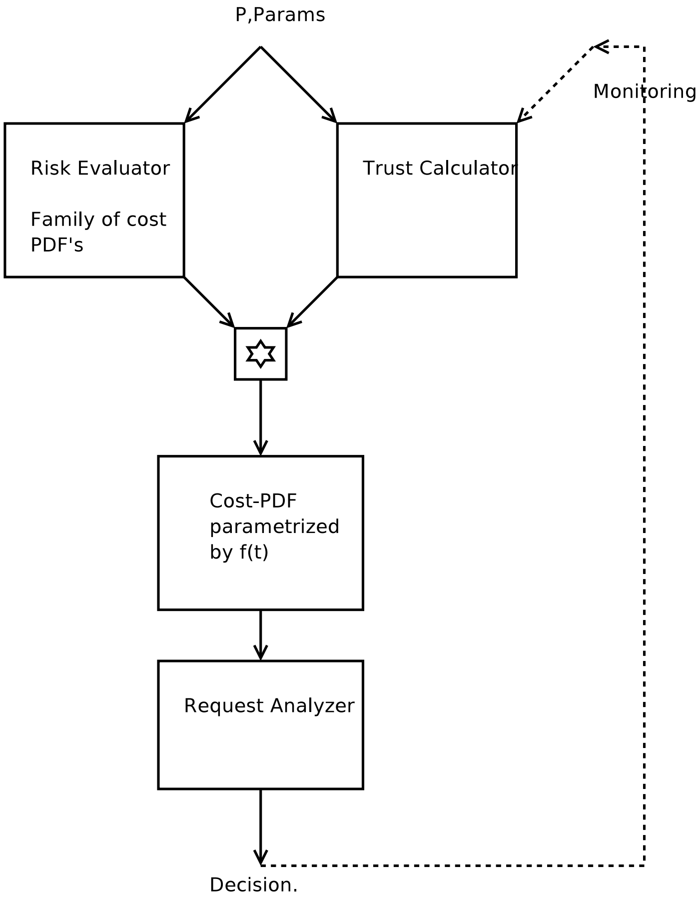

Figure 2.3: Overview of the decision-making process in SECURE

2.3.4 Automated Trust Negotiation

The reasoning behind Automated Trust Negotiation origins form the trust establishment procedures in the real world. Automated trust negotiation is a more advanced approach to the trust establishment system. Opposite of to the ordinary PKI's described earlier, the trust-negotiation focuses on an attribute bases access control(ABAC)[32] where usual structures focuses on the identity. Quoted from[32]:

> The goal of . . . automated trust negotiation (ATN) is to enable re-

source requesters and access to mediators to establish trust in one another through cautious, iterative, bilateral disclosure of credentials, in addition to the granting of system resources.

2.3.4.1 A real-world Trust Negotiation example

An example interaction described in [31] it the real-world purchase of a shirt, where the person interacts with the reseller, disclosing her credit card to the reseller. The trust mechanisms in this interaction are various, going from.

- The costumer trust the store to be decent, fair pricing and handle reclamations well.
- The costumer trust the store to handle the credit card details securely.
- The store trusts the costumer to, not try to pay with a stolen card.
- The store trusts the credit-card issuer, that the money paid actually exists and will be transferred.

There are several certificates in action here. The costumer wouldn't have chosen the store in the first place if it didn't look "decent". The costumer have probably seen the sticker on the entrance door to the store, that states that this store accepts credit-cards from the particular credit card issuer. The store wouldn't let the costumer leave with the goods before the payment validated at the credit card issuer.

Certificates could be something like:

- Being recommended by a friend.
- The shop having residence at the shopping centre.
- Many costumers in the shop.
- The shop is looking cleanly and decent
- The sticker at the front door, stating which credit cards the store accepts.

Seen from the costumers viewpoint the credit card it the sensitive credential in this scenario, it discloses information about how to draw money on the owners account and would thereby only be disclosed to "trusted" entities, that assure that only the agreed amount of cash is being draw and the information is not disclosed to any 3'rd parties.

The diverse nature of these gives an ordering of these certificates based on sensitivity. The protocol to disclose the certificates is called trust negotiation. Many of the "assertions" about interactions is not currently possible on the Internet. Some of them can be assumed when both the costumer and the store is placed within the same jurisdiction (country) and abuse can be tracked down by the legal system in the real world.

2.3.4.2 Trust negotiation on the Internet

In order to make the same assertions as described above on the net, the ordinary structure of certificate disclosure it not sufficient. The PKI needs to focus on attributes instead of identity. An essential component is a Policy language that describes the actual disclosure and/or access policies of the resources and certificates.

The $\mathcal{RT}$ policy language The $\mathcal{RT}$ policy language[18] has explicitly support for Attribute based access control. The work on the $\mathcal{RT}$ policy language covers more than described here, only the part essential to ABAC is described. The $\mathcal{RT}$ policy language can be used to describe access control based on roles modelled within the attributes of the certificates. In the following examples entities are described by $A, B, \ldots$, A role is an entity followed a role-name separated by a dot. $A.R_1$, which describes that A has authority to define members of R_1. Roles can then be chained. A chained role could look like; $A.R \leftarrow A.R_1$, which delegates the role R of A to the role R_1 of A.

The notation of $\mathcal{RT}$ is written in table 2.4.

If the entity of the right hand side of the role definition is different from the left hand side, the rule represents an delegation. Thereby $A.R \leftarrow B.R$ represent that authorisation of the role R is delegated from A to B. This makes it possible to define cross organisational authorisation delegations, like. $A.R_1 \leftarrow B.R_2$ which describes that the rule R_1 of entity A is delegated to entity B's rule R_2.

The logical extension is linked rules where the rules are written in combination as: $A.R \leftarrow A.R_1.R_2$ which is interpreted as $A.R$ is delegated to $B.R_2$ if $A.R_1$ contains B.

The language allows intersections in the role-definitions, representing that sev-

Name	Rule	Description
Type 1	$A.r \leftarrow D$	D has attribute(role) r defined by A
Type 2	$A.r \leftarrow B.r_2$	A asserts that an entity has attribute r if B asserts that the entity has r_2
Type 3	$A.r \leftarrow A.r_1.r_2$	A asserts that an entity D has r if there is an entity B that has r_1 and asserts that D has r_2
Type 4	$A.r \leftarrow A_1.r_1 \cap A_2.r_2 \cap \ldots$	Is an intersection, where all the rules needs to be satisfied. i.e. A asserts and entity has r if A_1 asserts that the entity has r_1 and ...

Table 2.4: $\mathcal{RT}$ policy rules in the basic language

1	EPub.disct	$\leftarrow$	EPUb.preferred $\cap$ EPub.student
2	EPub.preferred	$\leftarrow$	EOrg.preferred
3	EOrg.preferred	$\leftarrow$	IEEE.member
4	EPub.student	$\leftarrow$	EPub.university.stuID
5	EPub.university	$\leftarrow$	ABU.accredited
6	ABU.accredited	$\leftarrow$	StateU
7	StateU.stuID	$\leftarrow$	Alice
8	IEEE.member	$\leftarrow$	Alice

Table 2.5: $\mathcal{RT}$ usage example

eral roles are needed to satisfy the rule. $\mathcal{A}.\mathcal{R} \leftarrow \mathcal{B}.\mathcal{R}_1 \cap \mathcal{B}.\mathcal{R}_2$.

An $\mathcal{RT}$ usage example The example in table 2.5 from [19] describes that Alice is eligible for discount at the publishing service EPub, which is given to "preferred" costumers that furthermore are students. EPub defines preferred costumers as the preferred costumers of its "mother"-Organization EOrg which defines preferred costumers as members of IEEE. EPub defines students as any student at an university certified by ABU. The ABU is a certificate authority that issues certificates stating that the certificate holder is a university. Alice holds a StuID certificate issued by StateU(number 7)-university and a IEEE membership certificate(number 8).

In the actual system, every rule is represented by a digital certificate, thereby making it possible to verify the certificate chain in the system. The certificate discovery process is important to the system. In order for the described authorisation to take place, the EPub needs to have access to all relevant certificates in the actual certificate chain. But since the authorisation is delegate to different authorities and global access to all certificates cannot be assumed, this is an important issue.

In this particular example is it not unreasonable to believe that Alice and EPub actually would hold the complete certificate chain in there combined certificate.

The naming system in $\mathcal{RT}$ is important, since the usage of names like "EPub.university.stuID" will make sense only if the stuID attribute has the same meaning to EPub as to the University that actually has the attribute in its issued certificates and that the stuID attribute is used for the same purpose at all Universities. The proposal in $\mathcal{RT}$ it to use XML-name-spaces for the purpose as that will make sure that name-space collisions cannot be made by coincidence.

$\mathcal{RT}$ has a feature called parameterised roles. These looks like $A.stuID(year > 1999)$, which could represent a stuID attribute with a content-field named year that needs to be above 1999. This allows to further subdivide the roles based on attribute content. $\mathcal{RT}$ defines more advanced rules as threshold rules, thereby making it possible to state that the rule is satisfied by X or more, different certificates.

Sensible attributes and limited disclosure Another part of the work in automated trust negotiation covers the protection of sensible attributes and limiting disclosure. There are several cases where information about the existence of a certificate can be considered confidential, this problem is partially solved by the trust negotiation itself, and partly by introducing Ack-policies[32, p.12] that is a local flag stating, that the existence of a rule may not be revealed until it is satisfied. The negotiation strategies are not covered here but are carefully covered in [23],[29],[24] and [33].

As seen in the example the $\mathcal{RT}$-policy language has a very simple and flexible language to describe cross organisational access control decisions. The example spans over the EOrg, IEEE, ABU and StateU organisations, making it possible for Alice and EOrg to take advantage of the existing certificates issued by the university and IEEE.

2.3.5 Summary

Morton Deutsch definition of trusting behaviour is commonly accepted and widely referenced in other trust literature. Stephen Paul Marsh makes a great contribution to the trust literature by formally defining three types of trust and a relations between them. The different trust types is well-argued although the relation among them gives rise to some problems.

It seems as insufficient or inappropriate to use the multiplications operator to aggregate the trust-values, when they are defined in the interval between $[-1, 1]$.

The term trust management covers a lot of different approaches typically surrounding public key infrastructures that all tries to address the trust management problem. Some public key infrastructures carries trust information as a part of the certification process, but the authorisation process is always left on to the application itself.

Policymaker takes the first step towards a generic policy-language where it it possible make assertions and query the policy. The SECURE project adds another dimension to the trust management problem by addressing that the human notion of trust is based on more than just certificates and strict access control, but evidence and recommendations is a vital part that can be used to ease the overall trust management problem and even addressing scenarios which is not reachable within a more strict certificate based trust scenario.

Automated Trust Negotiation addresses the issue that certificates can be sensitive by themselves and negotiation strategies are needed to disclose the correct certificates in the different situations. The origin of this notion is a view on the real world where certificate are sensitive (credit cards, etc.) and negation strategies happens all the time. A more complete policy language RT has been adopted for automatic trust negotiation. Trust negotiation makes is possible to leverage the combined strength of certificates, not limiting the disclosure process to a single certificate.

2.4 Summary

The most commonly deployed grid systems are based on the Globus Toolkit, which uses “GSI” as the security infrastructure. GSI is an X.509 based security infrastructure with a few additions. This structure fits relative closed collaboration groups where the number of producers are few and the consumers are many.

In an open grid system, where the producer/consumer ratio is more even, it seems like a more flexible security infrastructure should be adopted. Infrastructure like PGP and SPKI are “subject”-oriented, more focused on a peer-like application scenario.

Trust management systems are still a very young concept, thus not being deployed yet. Stephen Paul Marsh provides good reasoning about forming trust as a computational concept and contributions from the SECURE project gives a more clear notion of outcomes and events.

Trust negotiation gives the flexibility of limited disclosure of certificates combined with the expressive $\mathcal{RT}$-policy language, that makes it very easy to define cross-organisational access-control-rules. This approach seems very close to the human approach to the decision making process.

Chapter 3

Analysis

Grid systems are currently used most in the academic environment, but the general distributed computing challenges, that grid systems try to solve are faced in many other environments. With respect to the security infrastructures and the issues that grid systems faces in this area, the demands are not particular different from the security issues faced in other areas of distributed computing or just the Internet in general.

In the real world collaborations, the focus on identity is often not the primary goal when making decisions. The human notion of trust based on recommendations and evidence is an important factor when humans are making decisions. We argue that being able to resolve the identity behind a public key is neither sufficient nor required in a grid environment.

Existing approaches like X.509 or PGP are mostly attempts to address the real world identity behind a particular public key. This enables the administrator of the system to associate the direct entity with the access control lists. This approach has been disputed already back in 1996 in “Decentralised Trust Management” in [2]:

> Existing systems that support security in networked applications, including X.509 and PGP, address only narrow subsets of the overall

> trust management problem and often do so in a manner that is only appropriate to only one application.

When discussing trust in grid systems, there is generally two problems that have to be considered.

- The access-control problem, e.g. should a principal be granted access to the resource.
- The resource selection problem, e.g. given a set of resources offering to provide the service, which will provide the best service.

These problems are related in a certain manner. Morgon Deutsch and Stephen Paul Marsh's work both focuses on the situation where the evaluating entity has a choice. In the former problem, the choice is to let the entity in or not. In the latter the choice is more complex since it deal with a set of entities from where there should be picked a certain entity that has the "best" properties.

This chapter gives an analysis of real world interactions, discusses the existing approaches to certificates, public key infrastructures and trust management systems. Finally a solution to the security challenge of open grid systems is proposed.

The result of direct interactions are called: "experience". When "experience" is distributed it is called "recommendations".

3.1 Analysis of the real world

Interactions in the real word happens all the time. When you go to the grocery store to buy milk, when you lend sugar to your neighbour. But trust is likewise build from recommendations. We even make trusting choices in entities that we have never had any direct relationship with before. Recommendations from friends are a central way of reasoning about such new entities.

Trust, identity, reprocication and certificates are all elements of real world interactions. They are analysed in the following sections.

3.1.1 Identity in the real world

In order to talk about people, things, shops, etc. the term identity has been used. In the situation of a person we often refer to the physical person as the "identity". In general we argue that the identity is: "a combination of properties that presumably are connected". This is a rather vague definition, but it seems to be appropriate. Properties that often are referred to as an identity are: age, name, nationality, height, size, hair colour, voice, signature, smile, DNA, blood type, clothes, facial features. Some of these properties are obviously more escapable than others. Some properties are linked together in some kind of identity certificate, such as the drivers licence that (in Denmark) contains: Age, Name, facial features and the information that states what kind of vehicle that the identity is allowed to drive.

None of these properties are by themselves in-escapable, but the combination of them makes it harder for another person to use the certificate, thereby binding the "driver property" more closely to the person. This system is by no means perfect, the pictures are often several decades old and the name and age are not unique. The particular certificate is a printed plastic card, with a detailed pattern, thus making it hard to create them at home.

The legal structures support the drivers certificate in order to make the system more solid. It is illegal to make fake drivers licenses. And the licenses can be withdrawn if the traffic laws are not respected. All these features combined leads to the mechanisms behind the drivers licence.

In most situations, there are no actual certificates involved, but the identity is defined by a loose set of properties in the mind of other people, most often linked to the non-unique, escapable property; "the name of the person".

3.1.2 Interactions in the real world

The human notion of trust is not particulary well defined, but given a situation almost every person would be able to make a decision and probably argue reasonable about it.

Situation - A real world purchase Imagine a situation where a costumer needs to buy milk (or other purchase). Often the costumer faces a choice of which store to purchase from. When making the decision several factors are typically taken into consideration.

- Past experience with the stores.
- Rumours about the store (recommendations)
- Time available to accomplish the purchase.
- General availability of stock.

But several stores will probably be directly excluded from the decision process due to:

- Store doesn't full fill reasonable demands, like accepting popular credit cards or they employ illegal labour.
- Return/reclamation policy is not acceptable

Interesting properties from this interaction are:

- The transaction is purely anonymous, seen from the consumer.
- It is the costumer that selects the store, thereby making the decision.
- The level of identification between the customer and the store is solely a decision made by the costumer.

This is a situation of real world interaction where we take many factors into consideration even though it is a small decision to make. The risk imposed by a wrong choice is not very high, only a small waste of time and a product that could be unusable.

Given the theory of Paul Marsh described earlier, utilisation is an issue, that are used when making decisions. The argument is; given a situation where the risk is bearable and the damage is reasonable, the decision could favour an unknown entity that full fills the basic demands of the interaction.

This is a logical implication of the fact that no-one has global knowledge about all entities in the system and even less knowledge about the potential behaviour of all entities, thereby seeking information to answer the question whether this "new entity" is better than the currently known entities. The interaction does not reveal the costumers identity to the system, not even if there is a payment involved.

In the situation of a alcohol purchase, where the costumer need to have a certain age before being allowed to purchase. The costumer has credentials to prove that the demands has been fulfilled, but the transaction is still anonymous in the sense that the store afterwards cannot match the purchase with other purchases done earlier by the same entity. Only in situations where the legal structure demands registration of purchases and a full identification is done as part of the interaction. Compared in terms of frequency these anonymous transactions far outnumbers the one where identification are done.

Seen from the subjective view of the consumer, there is actually a often a quite complete identification of the store, thereby allowing the consumer to express trust in the store and recommend the store to other costumers.

The trust addressed is dependent of context. No matter how much trust have been built buying milk in a grocery store, that trust will not have any significant influence on where the next car will be bought. But within the domain of grocery store, the trust are probably transferable between the different goods available.

Speaking using the vocabulary of Stephen Paul Marsh, the situations forms a relatively low basic trust, a relatively low general trust, but a relatively high situational trust.

The implications to the “trust value” of an interaction are:

- More “good experience” will make it more likely to trust the entity. Likewise with “negative experience”.
- More “positive recommendations” will make it more likely to trust the entity, though the impact would be less that if the same interaction was direct. (experience). Likewise with “negative recommendations”.
- Recommendations will be treated different depending on the entity that issues the recommendation.

Situation - Interaction between people The interactions that take place between friends have different characteristics. Where purchases are not based on payment there are typically build an implicit reprocication that are hard to describe. The identification process are typically equal for both entities, thus acting more as equal peers, that producers and consumers.

The situation is quite more abstract that the producer/consumer example above, since trust tend to be more context independent and directed directly at the entities.

Using the description of Stephen Paul Marsh this gives a more strong connection between the different trust types, thus is seems as care should be taken when aggregating these different trust types together. Marsh only vaguely describes reprocication but acknowledges that reprocication represents a major to these systems. Reprocication is left for future works.

3.2 Certificates

The ability to certify or assert that certain properties of a public key is a key part of the authentication problem. The certificates described in the previous chapter are discussed here.

3.2.1 X.509 Certificates

In the adopted X.509 public key infrastructure described in Section: 2.2.2.2, the certificates are typically used to describe identities. In the Globus grid system this is particularly the persons granted access to the grid resources or the specific grid resources. Usage of the X.509-certificate infrastructure depends on that all collaborating entities being certified by the same or the same few certificate authorities. In order to ensure cross organisational authorisation, the certificate authorities needs to trust the other organizations certificate authorities and their certification policies.

The resources trusts a (set of) root-certificates. This set needs to be big enough to ensure widespread adoption. More specific it needs to ensure that all entities has at least one certificate authority to trust. But in order to address trust in the certificates it needs to be small enough to ensure that there is no malicious certificate authorities among.

Access it based on the subject field in the certificates, thus the combination of a trusted root-certificate that can be used to verify the client-certificate and a subject string, that uniquely defines the entity within the certificate authority.

```
Subject: O=Grid, O=NorduGrid, OU=student.dtu.dk, CN=Jesper Krogh
```

This system favours secure collaboration in tightly connected systems where people need to contact a certificate authority prior to interaction, in order to get a certificate issued.

The information written in an X.509 certificate is the certificate content. When a X.509 certificate is transmitted the certificate content is always disclosed in plain text to everyone. Within the scope of one certificate authority this information uniquely defines an entity thereby privacy issues arises from the system. Since all authorisations is based upon knowledge of the entity, is it impossible to express information about an entity's relationship without revealing certificate content.

An example could be that you use the governmental X.509 PKI OCES[1] to prove that the holder of the certificate is a Danish citizen. This is exactly the purpose of the PKI, but it is not possible to disclose the certificate and prove Danish citizenship, without disclosing your name and the other content of the certificate in the same interaction.

In the deployed X.509 certificate infrastructures, the entities address trust directly in the issued certificates. This is done under the assumption that the certificate authorities don't fail. A failure of the certificate authority would lead to the specific certificate authority to be distrusted. The effect is that the certificate authorities need to aim at a rather high certificate policy to be met, before issuing certificates. As a result the number of certificate authorities rises towards the situation where every entity has its own certificate authority. The essential issue is that there only is one ultimately trusted entity. The entity itself.

The X.509 certificate infrastructure works well in systems like the current globus grid implementations where there are many consumers and few producers. But in a distributed system where the ratio between producers and consumers is more equal, the system fails to work properly.

The X.509 has been adopted to both manage the identity of hosts and individuals on the Internet. This is done by adapting a strict hierarchy of certificates that makes it possible to get a closed subset of the real world identities. The view of the principal in context of an X.509 PKI is shown in figure 3.1. This figure illustrates the fact that there isn't a global X.509 PKI that has a notion of all principals in the "universe". Different PKI's have different, not unique, subset of the principals. Every Certificate Authority guards it subset with a certificate policy and for the X.509 to be as flexible as the PGP or SPKI system every principal needs to run his own Certificate Authority.

[1] http://www.digitalsignatur.dk

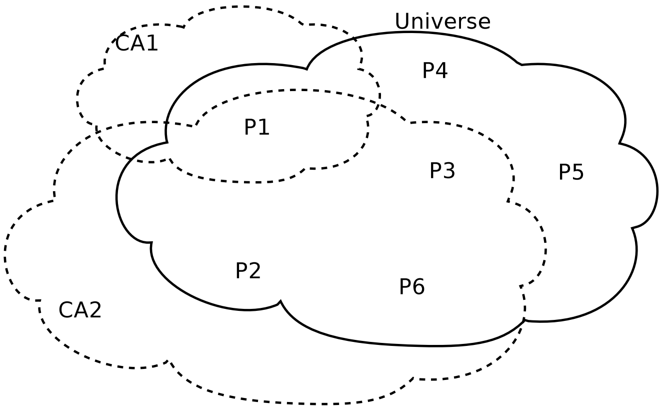

Figure 3.1: The X.509 certificate structure.

3.2.2 PGP Certificates

The PGP system described in section 2.2.2.1 is like the X.509 certificate system based upon identities. The only entity addressed in the PGP-system is humans, identified by the public key in the certificate. The PGP-system is not a centralised system like the X.509 system and thereby allows as many producers as consumers, making trust a property defined between the individual users in the system. This makes the PGP system appropriate for distributed systems, where no central authority is needed.

Trust in the PGP system is not defined by the certificates or signatures itself, which makes the PGP system more subjective. Trust is based upon the subjects own certification policies and weights given to the evaluation database where the specific signatures are given a particular weight in order to aggregate them.

The PGP-system is widely adopted and distributed, but addresses only identity. A signature in the PGP-system can only be interpreted in one way; "The issuer believes that this key,name association is correct", where names consist of human names and email addresses. The attributes connected to the public key are thereby limits the usage of the public key infrastructure significantly.

3.2.3 SPKI

The SPKI system addresses the PKI problem more widely and genericly than the other systems. SPKI doesn't limit the type of attributes that can be connected to a specific public key. Thereby there is no restriction that defines the scope and context of the certificates like the PGP and X.509 systems. The only limitation, the SPKI system imposes is that the certificate content are placed directly in the certificates. This can easily be circumvented and the SPKI system seem generic enough to provide the basic certificate infrastructure of a open grid system.

3.2.4 Probablistic PKI

Ueli Maurers approach to a PKI is simple and straightforward. It gives good reasoning about how "trust" can be aggregated together on a mathematical basis, but the assumption that "local" confidence values in certificates can be used in a global context to calculate the overall confidence values seems naive. An implication of this means that all entities have the same notion of a specific value. Trust values are given in the range of $[0, 1]$

The probabilistic PKI has no notion of context of the issued certificates. So the handling of that: A believes with 0.9 that B pays his bills can potentially be aggregated together with that B believed with 0.9 that C is good at bowling.

The probabilistic PKI describes a feasible approach to how these probabilistic trust values can be aggregated together, taking the effect of multiple and/or crossing certification paths into consideration.

3.3 Trust management

The human notion of trust consists of 2 parts, the trust in the infrastructure and the "knowledge". The most widely known example in the real-world is that trust in the legal system will uphold the law and thereby most people acts under the assumptions that almost any other entities does the same.

Trying to reflect this in the virtual world of the Internet is not an easy task, since most legal structures fails to provide anything. The usual certificate authorities (X.509) tries to map the identities of the real world to the virtual world, thereby making it easier to enforce the same structures in the virtual world. This approach is fine for uses where the mutual trust is high, like the current Globus

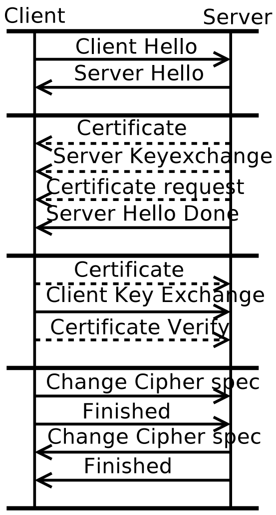

Figure 3.2: The TLS handshake protocol, dotted arrows represents optional blocks.

Based grid systems.

3.3.1 Certificate disclosure protocol

The most notable problem with this system is that it is generally not possible to be a part of two X.509 domains within the same application scope. Thereby disclosure of multiple certificates issued from several certificate authorities are not possible. The normal disclosure protocol in this context is SSL/TLS[7] where a layer is added between the application and network transport layer of the network reference model OSI[26, p.56]. The certificate handshake of this protocol is shown in figure 3.2 (From [15])

The limitations of the TLS handshake protocol are listed in [15, 3.2] and quoted here.

- Certificate are exchanged in plain text during the initial TLS handshake. An eavesdropper can thereby monitor the certificate content.
- The client and server are limited to disclosing a single certificate chain.

- The server specifies a list of distinguished names of certificate authorities that the server trusts. The client doesn't have this possibility.
- The server always discloses certificates first, thereby forced to disclose the certificate to a complete stranger.
- The client receives the certificate first, then discloses the client certificate to the server before the server has provided proof that it owns the certificate.
- There is no way to exchange more certificates.

In the paper "Advanced Client/Server Authentication in TLS"[15] is the Trust Negotiation protocol implemented to prevent the above listed limitations. This is done using X.509 certificates. A solution based on trust negotiation seems appropriate in order to circumvent the various limitations of the existing protocol.

3.3.2 Evidence and recommendations

Adopting the human notion of trust in the scope of a grid system gives several challenges. Evidence and recommendation gathering are a vital component since it represents the "memory" of the system. None of the discussed technologies are specific about how to do this. Several properties should be a part of the system.

- Evidence is gathered on the basis of observable events. Every event provides evidence, in the situation of an unsuccessful connections attempt, the actions outcome is "unknown".
- Evidence is dependent on context.
- Recommendation is evidence where the consumer is not the local entity.
- Recommendations and evidence looses value over time. Like the human mind forgets.

In the SECURE system, recommendations are trust-values (aggregated evidence) from the other system. Since both evidence and recommendations are highly context dependent, this approach seem undesired in grid systems. Distributing evidence as the aggregated values solves the privacy concerns related to the evidence but it makes it hard for the receiving entity to ensure that the evidence isn't double-counted or that the entities actually are referring to the same type of situation.

When distributing recommendations, the privacy aspect needs to be addressed. Recommendations that are too specific will leak information about transactions and recommendations whereas recommendations that are too generic will be hard aggregate together. A notion of strength and context needs to be applied to evidence so the subject evaluating the evidence can determine the value in the proper context.

3.4 Summary

Certificates in the real world are means of connecting attributes to people. The model used to transfer this model into the Internet has been to connect the attributes to a specific public keys, where only the certificate holder has access to the private part and thus can prove ownership of the key. The most adopted PKI is the X.509 system that gives a strict hierarchical approach, where the certificates needs to be issued by central certificate authorities.

This approach fits well within the scope of tightly connected systems, but in the more generic case with many different entities and no strict outlined collaboration domain, it becomes inflexible and leads to privacy issues.

The SSL/TLS protocol allows only one certificate to be disclosed in each interaction thereby making it impossible to negotiate certificate as a part of the interaction.

Automatic trust negotiation solves all issues about certificate disclosure including privacy issues that arises from the existing TLS handshake. The implementation done in [15] is based on X.509 certificates that doesn't honour a more freely certificate structure like the PGP/SPKI system very well. Although it can be adopted, it doesn't change the problem that it is not optional to disclose certificate content, which is useful for using a PGP-like certificate chain without revealing unnecessary information.

The trust system needs to have a clear definition of context in order to separate the evidence and recommendations. The modelling of events in the SECURE system, using event structures to model the outcome are very generic and thereby provides a good model, that seems very close to the human notion.

CHAPTER 4

Design

This chapter describes the design of the actual system as introduced in the analysis chapter. The following sections describes the actual design and modelling of identity, certificate, evidence, recommendations and the system to handle them.

4.1 Identity and certificates

The overall goal of the system is to facilitate secure collaboration. This involves dynamic enrolment, thus no direct interaction is needed in advance.

The system is basically split in two separate parts, the key infrastructure that allows entity recognition and secure communication and the evidence/recommendation system that allows the implementation of some kind of "memory" in the system. One of the possible attack models of open systems are the "sibyl attack" [8]. The problem is that in a system where no central authority is present, you cannot address malicious peers, since they always are able to create a new identity, thus escaping the negative evidence collected and distributed about the old identity.

Another key issue is the privacy properties of the systems. With the emergence of wireless connections everywhere an eavesdropper can easily collect all messages sent between collaborators. Even if the information is encrypted, the distribution of certificates in order to start the encryption can be enough to collect information about the participants. The "Reality Mining" project[1] has shown that information about peoples movement and activity can be used quite precisely to predict the next movements.

4.1.1 Certificate structure

Every entity needs a globally unique identifier to be addressable. The most obviously source of globally unique identifiers, without introducing a central issuing authority is a public encryption key. This has the advantage that it can be proven that a specific entity, processes the certificate using normal encryption principles as mentioned in the first chapter.

The analysis revealed demand for a subject oriented certificate infrastructure that allows separation of certified data/certificate content and certificate meta-data/certificate type. A new certificate structure "XmlCert" that fits the $\mathcal{RT}$ policy language,reveals as little information as possible and has no need for a central repository/authority is implemented. The comparison of public key infrastructures are done in table 4.1.

The certificate infrastructure described has all the ideal properties described in the table. Most interesting is that the certificate type is explicitly specified in the certificates and thus are not an implicit part of the certificate infrastructure as the PGP and X.509.

The new certificate system is illustrated in figure 4.1 (top), where the certificate holder with public key 0x123454 has certified a friend of his, with key, 0x123455 who likewise has certified his friend. The separation of certificate content allows people to address "Per Persens" friends and their friends, even proving what they are, without revealing who they are. In the $\mathcal{RT}$ policy language, this certificate chain would satisfy this policy.

$$\text{Beer.Free} \leftarrow \text{0x123454.FriendsCert.FriendsCert}$$

The policy should be read like: The rule describes that Beer.Free delegates authority over the identification to anyone who has a FriendCert issued by anyone who has a FriendCert issued by 0x123454.

[1] http://reality.media.mit.edu

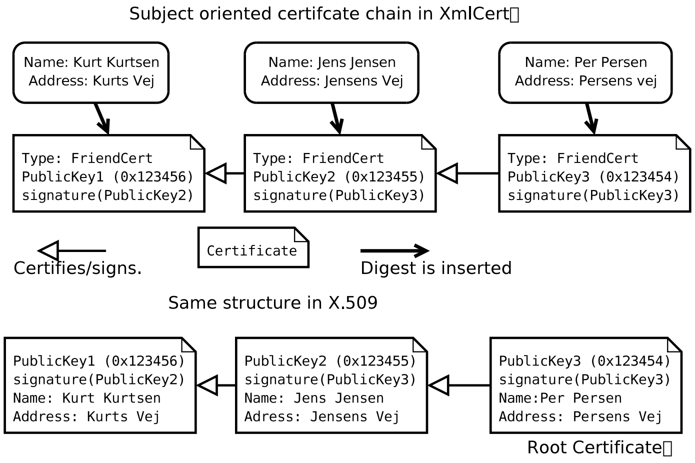

Figure 4.1: Ideal certificate structure (top), X.509 (bottom)

	X.509	SPKI	PGP	Ideal
Selective disclosure of attributes	-	-[2]	-	+
Certificate type meta-data/type selectable	-	-	-	+
Subject oriented (No strict hierarchy)	-	+	+	+
No central repository	-	+	-	+

Table 4.1: Certificate comparison chart

If the X.509 certificate infrastructure should be made to fit this scenario it could look like in figure 4.1(bottom), where the rightmost certificate is the "root-certificate". The certificate type is not modelled directly in the certificate, so it is implicit that it is a FriendCert-PKI that is modelled.

The disclosure of certificates in order to satisfy the policy rule, would not only prove that "0x123456" is friend of a friend of "0x123454" but all information about names and addresses would be disclosed within the same interaction as a result of that information is not separated.

The certificate type gives another name-space issue, since all entities should have the same notion of a FriendCert. This is implemented using XML-Name-spaces which are fully qualified by an URL. This enables everyone to define their own certificate-type or perhaps use some common certificates types. The choice of

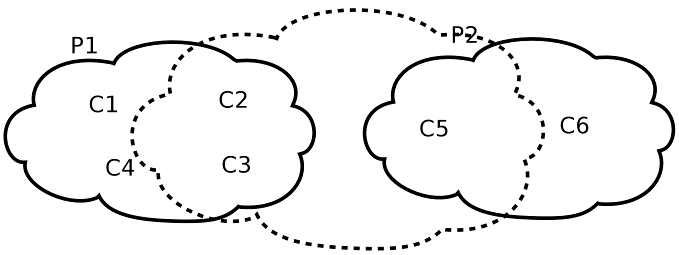

Figure 4.2: Collaboration between P1 and P2. The dashed cloud shows revealed certificates, C1,. . . are certificates

XML name spaces makes it easy to describe the exact content of the certificate attribute content using a XML-Schema. In case of parameterised roles, the actual certificate attribute must be passed on to the verifier.

In order to verify the certificate chain easily, it is proposed that the certificate contains both the public key of the subject/holder and the issuer. Since no central certificate repository is proposed the certificates should be propagated around between peers.

4.1.2 Identity

As described in the analysis chapter, real world identities holds many certificates of various importance and types. To ensure a system that allows dynamic enrolment, the system should be "open" to new entities. This allows anyone to create multiple identities in the system. Thus an "identity" or "principal" in the system, is a set of certificates.

In the real world, no one has complete knowledge about all properties/certificates of other people. People generally only have a limited view on the entity that they collaborate with. In this identity system an identity consists of the set of certificates that has been seen(and proven ownership of) in the same interaction. This identity is illustrated in figure 4.2, where an example interaction between to identities P1 and P2 can be seen. Seen from the view of P1, P2 is identified by C5, whereas seen from P2, P1 is identified by C2 and C3. The view is saved for further interactions. This could be updated by a later interaction where P1 discloses C1 and C2 to P2. The P2 local identification of P1 would be the set of all three public keys.

4.2 Certificate negotiation

As described in the analysis chapter, several issues are found in the usual certificate exchange protocol (SSL/TLS), there only one certificate disclosure can take place, thereby the situation illustrated in figure 4.2 is not reachable using the TLS-protocol.

An automated trust negotiation protocol is proposed, which allows the exchange of several certificates during the same interaction. Some certificates are more valuable than other. The imaginary situation where two secret FBI agents meet at the street with both their FBI certificates on them. They have no prior interactions with each other, but should exchange information that may not fall into the hands of a crooked 3'rd party. Neither do they wish to reveal the fact that they are FBI agents to everyone.

This problem is solved by guarding the certificates by a policy, that states something like: This certificate may first be disclosed when the recipient has provided enough confidence in his identity, so that the risk is low enough. This is implemented by guarding the certificates with an optional $\mathcal{RT}$ policy. Thus any certificate chain can be demanded prior to disclosure of the certificate.

The certificate negotiation CSP specification is shown in the tables 4.7 and 4.8. When the client initiates the interaction it has obtained the first certificate from the resource this is represented by "userinput" (SAPA) in the CSP specification. This certificate is added to the set of certificates known to belong to the other end of the communication(RS). The client then generates a new symmetric key, encrypts it using the certificate. Depending on whether it assumes the remote policy satisfied and/or the local policy satisfied it then consults the local certificatestore for certificatestore for certificates. The certificates disclosure policies must be satisfied using the certificates seen in the interaction already (RS) and not have been disclosed already(DC). If there are undisclosed certificates available the ELM-function takes a single certificate from this set and and adds it to the set of of disclosed certificates (DC) and send the combined package of a new certificate and a freshly generated symmetric key (type 1) to the other end and continues from C. Message types 2, 3 and 4 contains a OK/Failed flag that tells whether the policy in the other end of the communication are satisfied.

Both ends need to satisfy their policies in order to have a successful negotiation. If the remote policy is satisfied there are only need to disclose more certificates if the local policy is not satisfied. The additional disclosure of certificates could unlock additional certificates at the other end of the communication in order to satisfy the local policy.

Symbols	Description
$\mathcal{SK}_1^c, \mathcal{SK}_2^c, \mathcal{SK}_3^c, ..$	Secret keys belonging to c
$\mathcal{PK}_1^c, \mathcal{PK}_2^c, \mathcal{PK}_3^c, ..$	Public keys/certificates belonging to c
$\mathcal{K}^1$	symmetric key
$\{X\}_{\mathcal{K}_1}$	X encrypted using $\mathcal{K}_1$

Table 4.2: Notation used in Certificate negotiation

Type	Message type	Reply message type
1	$\mathcal{PK}_k, \{\mathcal{K}\}_{\mathcal{PK}_n}$	1 or 2
2	$\{\mathcal{K}\}_{\mathcal{PK}_n}, OK/Failed$	3 or 4
3	$\mathcal{PK}_k, OK/Failed$	1 or 2
4	OK/Failed	3 or terminate

Table 4.3: Message-types in certificate negotiation

The Client enters the “data communication stage” of the negotiation when an Ok has been received from the other end and the local policy has been satisfied.

The Resource enters the “data communication stage” when the client has send and Ok message and the local policy are satisfied and the local policy status has been send to the client.

Upon an unsuccessful negotiation the client terminates the communication.

In table 4.4 is the message-flow between the client, denoted by C and the resource, denoted by R. As seen in the table the initial message (Message 1) uses, the public key received by the resource to transmit a fresh generated symmetric key $\mathcal{K}_1$, it the expects future incoming messages to be encrypted using this key. Every time a new key is generated, the generating entity expects all further incoming communication to be encrypted using this key.

This negotiation scheme features the ability to selectively disclose certificates based on certificates earlier shown by the other entity. The protocol features 4 message types described in table 4.3. As can be seen in the table, every incoming message type has 2 different reply options. Message type 1 (disclosure of certificate and challenge of certificate), can be responded with both a new disclosure, but at least a challenge of the disclosed certificates.

The **ELM** function is the function that, from a set of certificates that has not been disclosed yet, will choose which to send to the other side next. This function could be trivially implemented, just by taking the first found, but that

Type		Direction		Content
1	Message 1	C → R	:	$\mathcal{PK}_1^c, \{\mathcal{K}_1\}_{\mathcal{PK}_1^r}$
1	Message 2	R → C	:	$\{\mathcal{PK}_2^r, \{\mathcal{K}_2\}_{\mathcal{PK}_1^c}\}_{\mathcal{K}_1}$
1	Message 3	C → R	:	$\{\mathcal{PK}_2^c, \{\{\mathcal{K}_3\}_{\mathcal{PK}_1^r}\}_{\mathcal{PK}_2^r}\}_{\mathcal{K}_2}$
1	Message 4	R → C	:	$\{\mathcal{PK}_3^r, \{\{\mathcal{K}_4\}_{\mathcal{PK}_1^c}\}_{\mathcal{PK}_2^c}\}_{\mathcal{K}_3}$
	...	Until the both ends satisfies policy or out of certificates.		
2	Message 5	C → R	:	$\{\{\{\mathcal{K}_5\}_{\mathcal{PK}_1^r}\}_{\mathcal{PK}_2^r}\}_{\mathcal{PK}_3^r}, \mathit{Failed}\}_{\mathcal{K}_4}$
	...	The resource sends an additional certificate		
3	Message 6	R → C	:	$\{\mathcal{PK}_4^r, OK\}_{\mathcal{K}_5}$
2	Message 7	C → R	:	$\{\{\{\{\mathcal{K}_6\}_{\mathcal{PK}_1^r}\}_{\mathcal{PK}_2^r}\}_{\mathcal{PK}_3^r}\}_{\mathcal{PK}_4^r}, OK\}_{\mathcal{K}_4}$
4	Message 8	R → C	:	$\{OK\}_{\mathcal{K}_6}$

Table 4.4: Example message flow in certificate negotiation

Variable		Initial value	Description
$\mathcal{A}$		Certificates	
$\mathcal{RS}$	=	$\emptyset$	Remote certificates seen.
$\mathcal{DC}$	=	$\emptyset$	Disclosed certificates.
$\mathcal{SC}$	=	$\{SC_1, SC_2, \cdots, SC_n\} \in \mathcal{A}$	Certificates hold by the server
$\mathcal{CC}$	=	$\{CC_1, CC_2, \cdots, CC_n\} \in \mathcal{A}$	Certificates hold by the client

Table 4.5: Initial configurations for a certificate negotiation

Operation	Description
$\mathcal{P}$	Policy
RP $\in \mathcal{P}$	Resource policy
CP $\in \mathcal{P}$	Client policy
$\mathbf{SATC}(\mathcal{N}, \mathcal{M}), \mathcal{N}, \mathcal{M} \in \mathcal{A}$	Gives the set of certificates in $\mathcal{N}$ satisfied by the certificates in $\mathcal{M}$
$\mathbf{SATP}(\mathcal{Q}, \mathcal{M}), \mathcal{M} \in \mathcal{A}, \mathcal{Q} in \mathcal{P}$	tests if the policy $\mathcal{Q}$ is satisfied by certificate set $\mathcal{M}$
$\mathbf{ELM}(\mathcal{N})$	Gives an element of N

Table 4.6: Operators and descriptions

Resource	$\stackrel{def}{=}$	(SAPA?(x:certificate) $\mathcal{DC} = \mathcal{DC} \cup$ x; $\rightarrow$ R)
R	$\stackrel{def}{=}$	(*left*?(rcert:certficate,sk:symkey) **if SATP**$(\mathcal{RP}, \mathcal{RS})$ **then** $\rightarrow$ Rchal[rcert,false] **else** $\rightarrow$ Rchal[rcert,true] [] *left*?(sk:symkey,s:status) $\rightarrow$ (**if** s = OK and **SATP**$(\mathcal{RP}, \mathcal{RS})$ **then** *right*!(OK) $\rightarrow$ Rdata **else** Rmore) [] *left*?(rcert:certificate,s:status) $\rightarrow$ **if** s = OK and **SATP**$(\mathcal{RP}, \mathcal{RS})$ **then** $\rightarrow$ Rchal[rcert,false] **else** $\rightarrow$ Rchal[rcert,true] [] (*left*?(s:status) **if** s = OK and **SATP**$(\mathcal{RP}, \mathcal{RS})$ **then** $\rightarrow$ Rdata **else** $\rightarrow$ Rmore)
Rchal [x:certificate, more:boolean]	$\stackrel{def}{=}$	(SK:secretkey = GENKEY(); $\rightarrow$ **if** more and **SATC**$(\mathcal{SC}, \mathcal{RS}) \setminus \mathcal{DC}$!= $\emptyset$ **then** cert = **ELM**(**SAT**$(\mathcal{SC}, \mathcal{RS}) \setminus \mathcal{DC}$)); right!(cert,$\{SK\}_x$); $\mathcal{RS} = \mathcal{RS} \cup$ x; $\mathcal{DC} = \mathcal{DC} \cup$ cert; $\rightarrow$ R **else** right!($\{SK\}_x$,**SATP**$(\mathcal{RP}, \mathcal{RS})$)) $\rightarrow$ R)
Rmore	$\stackrel{def}{=}$	(**if SATC**$(\mathcal{SC}, \mathcal{RS}) \setminus \mathcal{DC}$!= $\emptyset$ **then** cert = **ELM**(**SAT**$(\mathcal{SC}, \mathcal{RS}) \setminus \mathcal{DC}$));*right*!(cert,**SATP**$(\mathcal{RP}, \mathcal{RS})$); $\mathcal{DC} = \mathcal{DC} \cup$ cert $\rightarrow$ R **else** *right*!(failed) $\rightarrow$ R)
Rdata	$\stackrel{def}{=}$	left?(data) $\rightarrow$ right!(reply) $\rightarrow$ Rdata // negotiation done.

Table 4.7: Resource CSP specification

Client	$\stackrel{def}{=}$	(SAPA?(x:certificate) $\mathcal{RS} = \mathcal{RS} \cup$ x; $\rightarrow$ Cchal[x,**SATP**($\mathcal{CP}, \mathcal{RS}$)])
C	$\stackrel{def}{=}$	(*left*?(rcert:certficate,sk:symkey) **if SATP**($\mathcal{RP}, \mathcal{RS}$) **then** $\rightarrow$ Cchal[rcert,false] **else** $\rightarrow$ Cchal[rcert,true] [] *left*?(sk:symkey,s:status) $\rightarrow$ (**if** s = OK and **SATP**($\mathcal{RP}, \mathcal{RS}$) **then** *right*!($OK$) $\rightarrow$ Cdata **else** Cmore) [] *left*?(rcert:certificate,s:status) $\rightarrow$ **if** s = OK and **SATP**($\mathcal{RP}, \mathcal{RS}$) **then** $\rightarrow$ Cchal[rcert,false] **else** $\rightarrow$ Cchal[rcert,true] [] (*left*?(s:status) **if** s = OK and **SATP**($\mathcal{RP}, \mathcal{RS}$) **then** $\rightarrow$ right!(OK) $\rightarrow$ Cdata **else** $\rightarrow$ Cmore)
Cchal [x:certificate, more:boolean]	$\stackrel{def}{=}$	(SK:secretkey = GENKEY(); $\rightarrow$ **if** more and **SATC**($\mathcal{CC}, \mathcal{RS}$) $\setminus$ $\mathcal{DC}$!= $\emptyset$ **then** cert = **ELM**(**SAT**($\mathcal{CC}, \mathcal{RS}$) $\setminus$ $\mathcal{DC}$)); right!(cert,$\{SK\}_x$); $\mathcal{RS} = \mathcal{RS} \cup$ x; $\mathcal{DC} = \mathcal{DC}$ $\cup$ cert; $\rightarrow$ C **else** right!($\{SK\}_x$,**SATP**($\mathcal{RP}, \mathcal{DC}$)) $\rightarrow$ C)
Cmore	$\stackrel{def}{=}$	(**if SATC**($\mathcal{SC}, \mathcal{RS}$) $\setminus$ $\mathcal{DC}$!= $\emptyset$ **then** cert = **ELM**(**SAT**($\mathcal{SC}, \mathcal{RS}$) $\setminus$ $\mathcal{DC}$));*right*!(cert,**SATP**($\mathcal{RP}, \mathcal{RS}$)); $\mathcal{DC} = \mathcal{DC} \cup$ cert; $\rightarrow$ C **else** $\rightarrow$ terminate)
Cdata	$\stackrel{def}{=}$	left?(data) $\rightarrow$ right!(reply) $\rightarrow$ Rdata // negotiation done.

Table 4.8: Client CSP specification

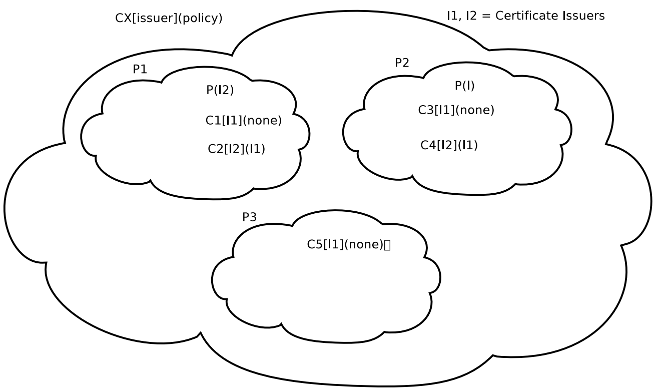

Figure 4.3: Situation where P3 can disturb the system

would not necessarily lead to the quickest satisfied state or to the state where least certificates has been disclosed, though yet satisfied. Certificate disclosure strategies are not covered within the scope of this report and are left for future work.

When a incoming message containing an encrypted session key $\{SK\}_k$ are being received, the protocol expects that all subsequent communication to the sender are encrypted using that key. This is illustrated in table 4.4. The key is not encrypted only using the last received certificate only, but using all certificates earlier disclosed in the protocol.

The first implementation had this weakness where a man in in the middle could intercept some of the first messages in the protocol, changing them with certificates of equals importance (thereby still satisfing both ends) and just pass on later messages. This would not compromise any direct communication between the entities since the encryption always uses the last negotiated keys for further communication.

The scenario where the problem arises is illustrated in figure 4.3 where I1 and I2 has certificate issuers P1 holds certificates C1 and C2 where C2 are issued by I2 and guarded by a certificate issued by I1, P2 holds C3 and C4 where C4 is guarded by a certficate issued by I1, but issued by I2. P2 has initially published C3 as the communication key and expects initial incoming communication encrypted using that key. Both resources needs certificates issued by I2 to be satisfied. At the end of the negotiation P3 has not gained access to anything interesting. But the final view from P1 would believe that the identity P2 has the certificates C5 and C4, where P2 would believe that P1 has the certificates

Step	Direction	Message	Comment
1	P1 → P3	(C1,$\{SK1\}_{C5}$)	P3 has substituted the initial certificate with C5.
2	P3 → P2	(C5,$\{SK1\}_{C3}$)	P3 passes on the key to P2, P3 has knowledge about the SK1, P3 sends C5
3	P2 → P3	$\{(C4, \{SK2\}_{C5})\}_{SK1}$	P3 has satisfied policy on C4 by C5, P3 has access to SK2
4	P3 → P1	$\{(C4, \{SK2\}_{C1})\}_{SK1}$	P3 passes on the message, can decrypt "outer layer".
5	P1 → P3	$\{(C2, \{SK3\}_{C4})\}_{SK2}$	P1 has satisfied policy on C2 by C6 and reveals C2, encrypted using SK2
6	P3 → P2	$\{(C2, \{SK3\}_{C4})\}_{SK2}$	P3 passes on the message, can decrypt "outer layer"
7	P2 → P3	$\{\{SK4\}_{C2}, OK\}_{SK3}$	P2 access policy satisfied by C2
8	P3 → P1	$\{\{SK4\}_{C2}, OK\}_{SK3}$	P3 can only pass-on the message
9	P1 → P3	$\{OK\}_{SK4}$	P1 decrypts SK4 and thus proof of C4 satisfies the P(I2)
10	P3 → P2	$\{OK\}_{SK4}$	P3 can only pass-on the message.
11	P2 → P3	$\{OK\}_{SK3}$	P2 has proof of C2 an are thereby satisfied.
12			.. data flow from here.. encrypted using SK4

Table 4.9: P3 in the "middle"

C5 and C2. This is a problem since these certificate collections are stored locally as a part of the evidence aggregation system.

The certificate disclosure could be done in many other ways, but this is the only situation where policy are satisfied and a wrong result has been reached. The modification is that in Step 7 where P2 "challenges" C2, it likewise challenges all earlier certificates disclosed, using the package $\{\{SK4\}_{C5}\}_{C2}$. This package can only be "passed on" by P3 and cannot be decrypted by P1 making the negotiation fail.

The negotiation transcript from the situation described in figure 4.3 is as shown in table 4.9.

A slightly modified situation where P3 has access to the same "quality" certificates as P2 would allow P3 to go all the way and gain access on P1. This is not considered a security violation, since P2 and P3 would be "equivalent" in the view of P1 and it is not within the scope of the negotiation protocol to ensure that the entities cannot proxy information out of the system.

4.3 Evidence and recommendations

In the real world, people remember the outcome of interactions, both positive and negative. They often ask friends and family about their opinions in order to make decisions. Evidence is first hand evidence and recommendations are evidence distributed between the different entities.

Evidence and recommendations are means of evaluating the performance of peers. The performance does not have any direct influence on the security, but is a vital component to the overall trust evaluation system. As described in the analysis chapter, evidence and recommendation were found to be context specific. A balance between generalism, so that we actually are able to aggregate different evidence together in a sane way, needs to be found. So the question to be answered using this information is: "Given a set of evidence and recommendations" give the identity most optimal.

Optimality is not a static subject either.

- On low risk, low throughput tasks, utilisation could be optimised.

- On high risk, high throughput tasks, performance could be optimised.

As described by Stephen Marsh, trust is self-reinforcing. Using this system and optimising for performance will render the system almost useless. The identity that performs most will probably be the one with most evidence related to. When choosing the peer with the strongest evidence you create more evidence related to the peer, thereby making it stronger.

Another approach would be to optimise for utilisation. This will have the benefit that you would choose a new peer for every interaction, thus creating evidence on the least rated peer every time.

The optimisation function needs to be specified in each interaction, since it depends on the importance of the actual task to be performed.

Name	Values	Description
Cpu	(int)	Cputime (perhaps measured in M-Flops)
Disk	(int,int)	Disk-space (time and space)
Net	(int,int)	Net bandwidth (time,data-size)
Grid	(Cpu,Disk,Net)	Combined component

Table 4.10: Example components of service level agreement

4.3.1 Service level agreements

We focus on resource consumption as the level where the evidence and recommendation should be transmitted. This reveals very little information about the actual agreement and seems generic enough. Basic example service level agreements are shown in table: 4.10.

Generally this system is build in an open way, so every entity can invent their own basic or combined service-level-agreements. Complete knowledge is not necessary in order to aggregate them together in a safe way. As recommendations, these service level agreements are build with a time-stamp, a random identifier, both peers public keys and signatures from the consumer part. If the interaction in any way would result in recommendations from the producer, this entity should just issue its own recommendation.

Evidence are produced by the local entity thereby the trustworthiness are indisputable.

4.3.2 Comparing evidence

The above component structure makes it possible to aggregate similar, but not equal information together. For an example given the evidence Grid(Cpu(10),Disk(20,30),Net(10,20)) and Disk(20,50). When queried for the strength of this evidence a context needs to be supplied. In context of a Grid-sla, the result would be only the Grid-sla, since the Disk-sla contains to few information to be taken into the consideration. In the context of Disk both the Disk(20,30) and Disk(20,50) would be taking into account.

Evidence is taken into account in a context if there is a subcomponent of the evidence that matches the context.

Symbol	Description
$\mathcal{E}$	All evidence
$\mathcal{C}$	All components
$e_1, \cdots, e_n \in \mathcal{E}$	Single piece of evidence
$c_1, \cdots c_n \in \mathcal{C}$	Specific contexts
$context(e_x) = c_1, .., c_n \in \mathcal{C}$	components of a piece of evidence.
$consumer(e) = (\mathrm{C})$	Gives the Consumer of the evidence/recommendation.
$producer(e) = (\mathrm{P})$	Gives the Producer of the evidence/recommendation.
$outcome(e) = (\mathrm{O})$	Gives the weight of the outcome

$$context(e) \cap c \quad \neq \quad \emptyset \tag{4.1}$$

Given a set of evidence in the same context. The evaluating entity would supply a function:

$$f : e \rightarrow x, x \in [0, \infty[\tag{4.2}$$

This function is evidence specific and should reflect the cost ratio between components of a piece of evidence if it consists of several components. An example could be the disk-sla above, where the evaluating entity believes that a “time-unit” is twice as expensive as a “space-unit”, the supplied function would then use its knowledge of this property to evaluate the strength of the evidence.

A natural choice for this function would be an exponential decreasing function. The human notion of memory seems to have a few properties; Interactions done in the past, tend to have less influence that new interactions and interactions are probably not “forgotten” before the information is considered so old that it doesn’t have any importance any more. The exponential decreasing function seems to fit this behaviour Evidence is than aggregated together in a given context, given weight using the supplied “strength”-function.

$$st(e,t) = f(e) \times e^{(-\frac{ln2}{t_{\frac{1}{2}}} t - time(e))} \tag{4.3}$$

time(e) extracts the timestamp from the evidence. If this strength function should be used to model the decay, then the half-life time $t_{\frac{1}{2}}$ should likewise be supplied by the user. What the appropriate half time is would depend much on the application scenario since some interactions would take place several times every hour and other would run for days.

Given all the evidence of the relationship between any 2 identities, the strength is the sum of the evidence.

$$evidence(P1, P2, t) = \sum st(x, t) * outcome(x) \quad (4.4)$$
$$\forall\, x, x \in \mathcal{E} \wedge consumer(x) \subseteq P1 \wedge producer(x) \subseteq P2 \quad (4.5)$$

Where P1 and P2 are the principals that the evidence is between. With every evidence the outcome is associated. The implemented system has a static outcome set, thus using an event structure is left for future works. The outcomes are "success","failure" and "dontknow". The outcome function extract the outcome from the evidence and consults the user supplied outcome map to get the weight of the outcome.

Figure 4.4 show and example where P1 has the certificates C1,C2 and C3, P2 has the certificates C4,C5 and C6. The figure shows four pieces of evidence of the same type. The three pieces of evidence E1,E3 and E4 are directed with P1 as consumer and P2 as producer and E2 the opposite direction. Given the strength-function f the total value of evidence directed from P1 to P2 would be:

$$\begin{aligned} E &= f(E1, t) * outcome(E1) && (4.6)\\ &\quad +f(E3, t) * outcome(E3) + f(E4, t) * outcome(E4) && (4.7) \end{aligned}$$

Where t is the current time.

An example output map is in figure 4.11 where the three outcomes are mapped to different weights. This map can only be assumed static within the scope of a single interaction type since some interactions generally would have large consequences from a failure, thus the a failure should be given weight much lower than a success and so on. The human interpretation of this map is quite intuitive; "Given the same interaction, how much worse is a failure than a success".

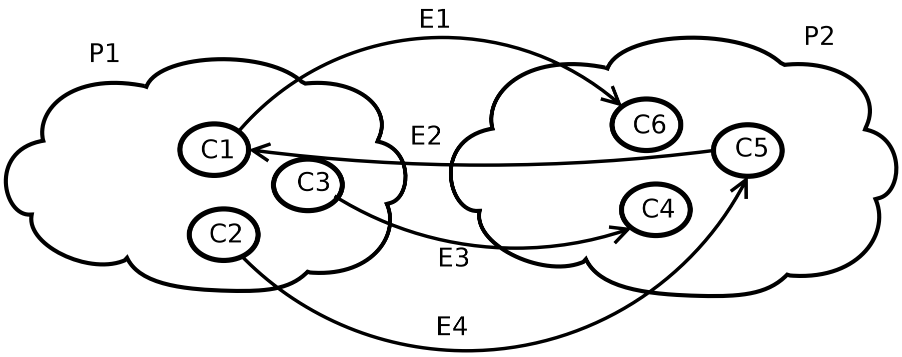

Figure 4.4: Evidence between the entities P1 and P2

Outcome	Weight
success	2
failure	-5
dontknow	-1

Table 4.11: Example outcome map

4.3.3 Evidence and recommendation evaluation

The evidence and recommendations are now aggregated into one value per edge in the recommendation graph as shown in figure 4.4. An example graph with several peers can be seen in figure 4.5. Since the system has taken evidence context into account, this graph seems feasible to use the trust evaluation algorithms suggested for a PKI by Ueli Maurer. This allows all paths to be taken into account in the system. This system operates on probabilities, the values thereby needs to be normalised into the interval [0,1].

The value 0 represents the edge in the complete system that has the lowest weight, which seems reasonable enough. The value 1 would represent the edge that has most evidence. This is problematic when compared to the human notion of trust. The value 1 would represent that evidence could be propagated between every peer in the system in one long chain and still not loose value, thus making the effect of recommendations equal to evidence. This is solved by normalising the graph into a smaller range of the probability range. In figure 4.5 the target range is [0,0.75], which tells that evidence that the evaluation entity is one part of are not degraded. Evidence on the number 2 step on the path are degraded with 0.75 and likewise further. The trust values are initially

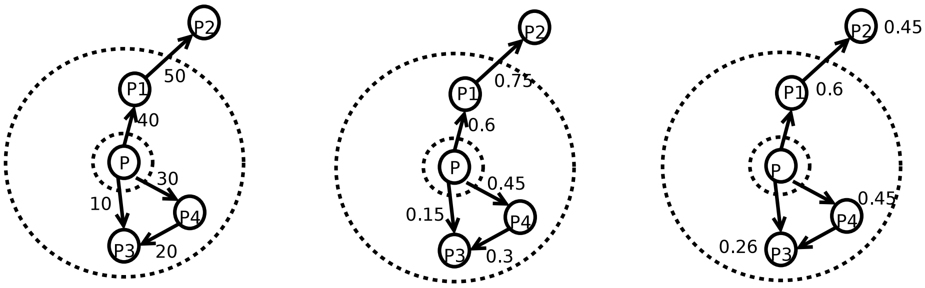

Figure 4.5: Left: Original strength-values on the edges, Middle: Normalised into the interval [0,0.75], Right: End strengths of peers.

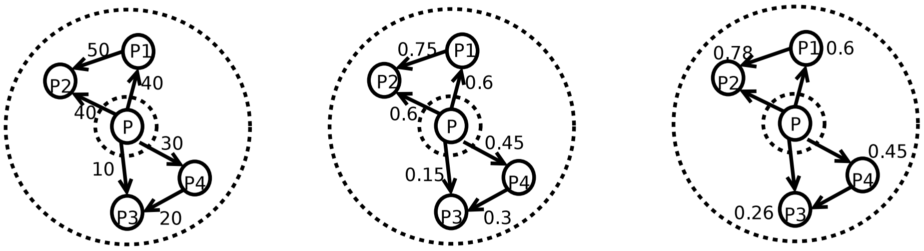

Figure 4.6: Figure 4.5 with an additional edge between P and P2

set to zero in all entities unknown to the system.

In figure 4.5 (right) the final strength-values are given, making this ordering of the peers(descending): (P1,P4,P2,P3) If the system should optimise for performance then P1 would be the optimal choice if the utilisation is the choice then P3 would be the best peer. Choosing P2 and having an successful interaction with strength 40 would make the system remodel the graph, to look like then one shown in figure 4.6, which basically reorders the system to (P2,P1,P4,P3)

In a system with generally well behaving well performing peers it seems like a reasonable approach to focus on the least valued peer.

4.3.4 Negative evidence and new entities

Generally new entities enter the systems as a result of recommendations from other entities. But initially the entities are assigned the evidence with strength

0 in order to be a part of the evaluation system. This may seem strange since it can lead to new entities having a stronger trust value than existing entities. Compared to the human notion of trust this seems quite similar, since people would probably choose to favour the unknown entity over known bad entities, thus seeking new collaboration partners.

Negative evidence will be formed as a result of bad outcome of interactions. It is essential for behaviour of the system to notice the connection between escapable identity (the ability to form a new identity) and the amount of negative evidence addressed to the entity. If an entity has the knowledge that much negative evidence has been issued about the behaviour the motivation of escaping the identity will increase. If the specific interaction relies on easily obtainable certificates only this process is very easy. Thus in systems where the certificates are easy escapable, much negative evidence will lead to a system with many short-living entities.

4.3.5 Trust in peers

The evidence and recommendations now form a graph between the different peers in the system. When the edges has been normalised into the appropriate interval, the strength of the individual entities in the graph are calculated using the principals from Ueli Maurers PKI described in section 3.2.4. Given the graph in figure 4.7 and the evidence values in table 4.12 the trust calculation for P4 would be calculated on these evidence paths(minimal subsets):

$$\mathcal{V}_1 = P \rightarrow P1 \rightarrow P3 \rightarrow P4 \quad (4.8)$$

$$\mathcal{V}_2 = P \rightarrow P2 \rightarrow P3 \rightarrow P4 \quad (4.9)$$

$$\mathcal{V}_3 = P \rightarrow P4 \quad (4.10)$$

$$(4.11)$$

Thus the inclusion-exclusion calculation of the trust value would be:

$$st(PX, PY) = \sum_{i=1}^{k} \mathcal{P}(\mathcal{V}_i) \quad (4.12)$$

$$- \sum_{1 \le i_1 \le i_2 \le k} \mathcal{P}(\mathcal{V}_{i_1} \cup \mathcal{V}_{i_2}) \quad (4.13)$$

$$+ \sum_{1 \le i_1 < i_2 < i_3 \le k} \mathcal{P}(\mathcal{V}_{i_1} \cup \mathcal{V}_{i_2} \cup \mathcal{V}_{i_3} \quad (4.14)$$

$$- \ldots \quad (4.15)$$

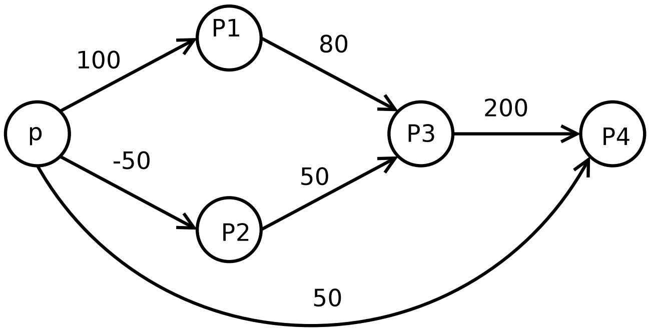

Figure 4.7: Example graph used for path calculation.

Using the example data the result of the strength between P and P4 is calculated (ns(Consumer,Producer) is the normalised strength from table 4.12):

$$
\begin{aligned}
st(P,P4) &= ns(P,P1)*ns(P1,P3)*ns(P3,P4) && (4.16)\\
&\quad +ns(P,P2)*ns(P2,P3)*ns(P3,P4) && (4.17)\\
&\quad +ns(P,P4) && (4.18)\\
&- (ns(P,P1)*ns(P1,P3)*ns(P3,P4) && (4.19)\\
&\quad *ns(P,P2)*ns(P2,P3) && (4.20)\\
&\quad +ns(P,P1)*ns(P1,P3)*ns(P3,P4)*ns(P,P4) && (4.21)\\
&\quad +ns(P,p2)*ns(P2,P3)*ns(P3,P4)*ns(P,P4)) && (4.22)\\
&+ (ns(P,P1)*ns(P1,P3)*ns(P3,P4) && (4.23)\\
&\quad *ns(P,P2)*ns(P2,P3)*ns(P,P4)) && (4.24)\\
&= 0.42*0.35*0.75+0*0.25*0.75+0.25 && (4.25)\\
&- (0.42*0.35*0.75*0*0.25 && (4.26)\\
&\quad +0.42*0.35*0.75*0.25 && (4.27)\\
&\quad +0*0.25*0.75*0.25) && (4.28)\\
&+ 0.42*0.35*0.75*0*0.25*0.25 && (4.29)\\
&= 0.33 && (4.30)
\end{aligned}
$$

In this graph P4 is the most complicated entity since there only is one effective certification path to P1 and P2. The effect of the zero value on the path between P1 and P2 makes contribution zero when calculating the strength of P3, thus P3 is only a result of the path containing P1. The other values are: P1 = 0.41, P2 = 0, P3 = 0.14 using the same calculation scheme.

This gives as complete order of all entities known to the evaluating entity. Based

Consumer	Producer	Strength	Normalised strength
P	P1	100	0.42
P	P2	-50	0
P1	P3	80	0.35
P2	P3	50	0.25
P3	P4	200	0.75
P	P4	50	0.25

Table 4.12: Strength values for figure 4.7

on which optimality goals the interaction has, it is now relatively easy to pick the optimal entity.

The evidence aggregation system is as pointed out by Ueli Maurer quite computationally expensive. Using the algorithm described it has quite high complexity but given the situation where an additional path is degraded with 0.75 it could make sence to limit the size of the paths. Experiments in this area are left to future works.

4.4 Complete system

In the former sections many components, their behaviour and relations has been described. This section tries to give an overview over the complete system.

The system does not depend on an index to work, just initial peers to work will be enough, since evidence and certificates can safely be changed between peers, but some kind of initial information must be exchanged in order to get the system started. In table 4.13 there is an example interaction with a small index that maintains a list over available resources.

It is not intended for the negotiation-process to use any of the trust values during the trust-negotiation. But a future extension could implement a system where the policy on the system would be lowered if an certain threshold in the evidence-system has been reached on basis of the in-negotiation disclosed certificates.

The system has many configurable parameters and components table 4.14 describes the scope of the different components. The certificates (without corresponding attributes) and the evidence may freely be exchanged between the entities in the system. Both evidence and certificates contains digital signatures

Step	Entity	Action
1	Resource	Sets up local policy for trust negotiation
2	Resource	Opens socket for connections
3	Client	Creates service-level-description
4	Client	Sets up local policy for trust negotiation.
5	Client	Queries service discovery for entities
6	Client	Evaluates local evidence/recommendation-store to rank entities.
7	Client	Selects entity based on the rank and optimality goals.
8	Client	Starts trust-negotiation with selected resource.
9	Client	If satisfied, send job to resource, else exclude resource from ranking and go to 8
10		Perform job.
11	Client	Updates evidencestore, distributes evidence.
12	Resource	Updates evidencestore, distributes evidence.
13		Finished.

Table 4.13: Example interaction in the complete system.

that ensures that modifications not are possible. The Principal association is local information that cannot be proven to other entities. The strength function and outcome map are application specific although it could make sense to implement some local defaults for these. The certificate attributes are not supposed to be distributed other than the situation where the actual interaction demand disclosure of the attributes. This is the only component that an entity can exchange without the owners approval. The certificate disclosure policies are only of interest to the local entity.

The system tries to support all interactions that can take place in a system. The $\mathcal{RT}$ policy language can easy implement a system similar to the existing

	Global	Local	Application
Certificate	X		
Certificate attributes		X	
Evidence	X		
Principal association		X	
Strength function		(X)	X
Outcome map		(X)	X
Certificate policy		X	

Table 4.14: Components and there scope in the framework

hierarchical X.509 public key infrastructure thus at least all situation that could be handled in the X.509 infrastructure can likewise be handled here. This system is able to scale from this strict hierarchical situation to the situation where no specific certificates are demanded but evidence and recommendations are build on a network of pure self signed certificates.

4.4.1 Weaknesses

The system only has a few weaknesses the most notably is the assumptions that entities newer exchange private keys/certificates with each other. As long as the certificates are reasonable valuable as in the X.509 system or the PGP system, this assumption holds true. But in a situation where the system consists mostly of self signed certificates the loss for an entity to trade the certificates is minimal. This would break the evidence aggregation system, since the assumption that the Principal association is “local” in scope no longer will be true. This problem could be reduced significantly by no accepting self signed certificates in the persistent principal store.

The other meekness is the situation where the evidence system consists merely of negative evidence. This situation enables malicious entities to be taken into consideration merely by discarding all certificates and create a new perhaps self signed certificate. The evidence aggregation system behaves best if there is a surplus of positive evidence. Compared to the real world, it doesn’t seem as a big mistake to assume that.

Chapter 5

Implementation and test

This chapter describes the actual implementation and test.

The code base are spilt in 4 essential parts and 2 test parts.

- **TrustCore -** The trust related code, Policy ($\mathcal{RT}$), evidence and evidence-store.
- **XmlCert -** the Certificate infrastructure. This implements a public key infrastructure directly in XML, including private encryption key store, key generators, dynamically attributes loading.
- **XmlIo -** The trust negotiation layer as described in the design section.
- **Sla -** The Service level agreements.
- **Grid -** Test stubs that allow creation of grid client and resources. The Grid is not a part of the design but only stubs to enable testing of the trust framework.
- **Indexserver -** Indexserver that enables initial usage of the system. The indexserver is not a part of the design but only stubs to enable service discovery.

Component	Implemented	Tested
XmlCert	+	+
XmlIo	+	+
TrustCore	+	+
Grid (stub)	+	+
Indexserver (Stub)	+	+

Table 5.1: Overall status of implementation

The overall status of the components are described in table 5.1

The indexserver and grid components are not essential to the trust evaluation system nor the certificate infrastructure and are written as stubs for testing purposes. The indexserver could easily be a system like the LDAP-databases of the Globus/Nordugrid system or the UDDI system of the Webservice world.

5.1 General implementation details

XML has been selected as the storage and transport format of all data within the system. Every class that needs to export data to a filestore or socket implements two methods.

- **setXmlElement -** Which parses a piece of XML into the object.

- **asXmlDocument** - Exports the object as an XMl element

The two methods takes care of exporting the object in to XML for network I/O of filesystem storage.

The usage of XML throughout the system makes all components of the certificates and the evidence system extendable with little effort. XML maps all documents into a global name space using URI's to identify the precise name space of attributes. This enables everyone to build their own "service level agreements" and certificate-type/attributes in their own XML name space.

5.2 Certificate infrastructure

The Certificate infrastructure are implemented in the "XmlCert" package. The certificate infrastructure that allows separation of attribute types are implemented in using XML-structures where the attributes are fully qualified by a name-space. All subcomponents of the certificate: Issuer and Subject public key and attributes are likewise fully qualified by its own name space.

5.2.1 Attributes

The attribute type represents the actual certificate type in the system. In the prototype implementation there has been implemented several certificates, but since parametrised roles has not been implemented this can currently be viewed as different certificate types, where the actual content cannot be used.

A certificate attribute implements this interface:

```
public String getDigest();
public MessageDigest getHashalgorithm();
public void setEncoding(Encoding enc);
public void setHashalgorithm(MessageDigest md);
public void setDigest(String dig);
public Encoding getEncoding();
public Document asXmlDocument();
public Namespace getNamespace();
public void setXmlElement(Element e) throws InvalidAttributeException;
public boolean equals(XmlCertAttribute xca);
public String toString();
```

The **getDigest()** gives the messagedigest encoded using the encoding set by **setEncoding()**. The algorithm used internally for computing the is set by **setHashAlgorithm()**. An example attribute looks like this:

```
<driver:attribute xmlns:driver="http://www.krogh.cc/attributes/driver"
  encoding="base64"
  hashalgorithm="sha-512"
  digest="fc2zFlOQLtvR  ....==">
   <driver:driver>
      <driver:licencenumber>1234567890</driver:licencenumber>
```

```
      <driver:name>Jesper Krogh</driver:name>
      <driver:country>Denmark</driver:country>
      <driver:noice>YXNmc2FmYXNmYXNmYXNmYWZzYQ==</driver:noice>
   </driver:driver>
</driver:attribute>
```

Which represents the attribute of the drivers certificate. If the certificate is distributed without the corresponding attribute content it will look like:

```
<driver:attribute xmlns:driver="http://www.krogh.cc/attributes/driver"
  encoding="base64"
  hashalgorithm="sha-512"
  digest="fc2zFlOQLtvR  ....==" />
```

The digest value represents a messagedigest of all information in the certificate attributes. There it will be a hash of the content of the "licencenumber", "name", "country" and the "noice" XML attributes. This "stripped" certificate fully describes what type the attribute is of and contains the messagedigest, which is necessary for validation of the certificate signature. The purpose of the "noice" attribute is to insert junk data in the attribute to prevent that a person who doesn't have access to the attribute content can brute force test the actual values.

The fully qualified XML-names-pace in this example are identified by the URI: "http://www.krogh.cc/attributes/driver", which can be linked together with an XML-schema to validate the actual XML-document contained in the attribute. Attributes implemented for test in the prototype are: NonDisclosureAttribute, GovNFPAttribute,DriversAttribute, BlindAttribute, ResourceAttribute.

All attributes implements the **XmlCertAttribute** interfaces and inherits the **DefaultAttribute** class. The DefaultAttribute class implements get/set routines for all common usage of an attribute including:

```
get/setHashalgorithm()
get/setDigest()
get/setEncoding()
```

If any special behaviour of these methods are needed, they can be overloaded in the implementation of the interface.

5.2.2 Encryption keys

The encryption keys are implemented using an interface for the corresponding public and private key pair. The only currently implemented key types are the RSA keys. The interface a public key needs to implement are:

```
public void setXmlElement(Element e) throws InvalidCertificateException;
public Document asXmlDocument();
public Namespace getNamespace();
public PublicKey getPublicKey();
public void setPublicKey(PublicKey pk);
public byte[] digestData();
public byte[] encrypt(byte[] plaintext) throws XmlEncryptException;
public boolean equals(XmlPublicKey xpk);
public String getFingerprint();
public String toString();
```

The XmlRSAPublicKey is a class that implements this interface. In the XML-document an RSA-key looks like this:

```
<rsa-pub:publickey
  xmlns:rsa-pub="http://www.krogh.cc/rsa/public">
    <rsa-pub:modulus>2065568454448507997..</rsa-pub:modulus>
    <rsa-pub:exponent>65537</rsa-pub:exponent>
  </rsa-pub:publickey>
```

The modulus are truncated since the actual size are in the order of 600 digits. Again this namespace of the RSA-key are fully qualified. All needed for the certificate infrastructure to accept a new key type is that it needs to implement the interface and export a fully XML representation of the key in the “publickey” attribute.

The private key has a slightly different interface:

```
public void setXmlElement(Element e) throws InvalidCertificateException;
public Document asXmlDocument();
public Namespace getNamespace();
public void setPrivateKey(PrivateKey pk);
public PrivateKey getPrivateKey();
public byte[] decrypt(byte[] ciphertext) throws XmlDecryptException;
```

The actual XML are like the public key but just defined in the “privatekey” attribute.

A key pair is the represented by a public and a private key put together in a key pair. And the actual keystore is implemented as an XML-document with several keypairs.

5.2.3 Certificates

As described in the design chapter, the certificates contains very little information. The XML-dump of a certificate are:


```
<xmlcert:certificate
  xmlns:xmlcert="http://www.krogh.cc/xmlcert">
  <xmlcert:policy>
    <disclosure:disclosurepolicy
       xmlns:disclosure="http://www.krogh.cc/policy/disclosurepolicy"/>
  </xmlcert:policy>
  <xmlcert:subject><rsa-pub:publickey ..
   ..</rsa-pub:publickey>
  </xmlcert:subject>
  <xmlcert:issuer><rsa-pub:publickey ..
   ..</rsa-pub:publickey>
  </xmlcert:issuer>
  <xmlcert:issuedate>2005-07-29</xmlcert:issuedate>
  <xmlcert:expiredate>2005-07-29</xmlcert:expiredate>
  <xmlcert:issuersignature encoding="base64">MfDRs5r4C..
  ..xOVRPnjdQ==</xmlcert:issuersignature>
  <xmlcert:attributes>
    <nda:attribute
      xmlns:nda="http://www.krogh.cc/attributes/non-disclosure"
      encoding="base64" hashalgorithm="md5"
      digest="4nlsYQv7KmOBXYhNn/nfFA==">
        <nda:nda>
  <nda:name>Entity</nda:name>
  <nda:noice>YXNmYXNmZHNhZmFzZg==</nda:noice>
</nda:nda></nda:attribute>
   </xmlcert:attributes>
  </xmlcert:certificate>
```

The issuersignature is a signature of:

$$\begin{aligned} hash &= digest(issuedate & (5.1)\\ &\quad +expiredate + attributedigest + attribute - namespaceuri & (5.2)\\ &\quad +issuerpublickey + subjectpublickey) & (5.3)\\ signature &= E(PK_{issuer}, hash) & (5.4) \end{aligned}$$

The signature is encoded using the encoding algorithm specified in the "encoding" attribute of the issuersignature element. This enables the certificate to be verified with or without the actual attribute content attached to the certificate.

The disclosure policy of the certificate are not related to transmitting the actual certificate but are related to proving the ownership of the certificate in a certificate negotiation situation. In the example certificate, this policy is empty, which represents that the certificate always are allowed to be disclosed. An disclosure policy are an simple $\mathcal{RT}$-rule representing the certificate that should sign the satisfying certificate and the certificate type.

```
<xmlcert:policy>
  <disclosure:disclosurepolicy
    xmlns:disclosure="http://www.krogh.cc/policy/disclosurepolicy">
    <disclosure:root>RKtCjQG</disclosure:root>
    <disclosure:namespaces>
      <nda:ns xmlns:nda="http://www.krogh.cc/attributes/non-disclosure"/>
    </disclosure:namespaces>
  </disclosure:disclosurepolicy>
</xmlcert:policy>
```

The policy example is satisfied using a "nda"-certificate, signed by the issuer identified by the fingerprint "RKtCjQG". The usage of fingerprints of certificates are only implemented as convenience and should be changed to the actual keys before real world usage.

Like the keystore, there is a certificatestore implemented that takes a set of certificates and store them in a single XML-document for persistent storage.

5.2.4 Administration tool for keys and certificates

As a part of the implementation, there has been build a command line administration tool for keys and certificates called "XmlCertAdmin". This takes care of:

- Key generation of various bit sizes (RSA only)
- Certificate generation.
- Signing of certificates.
- Setting certificate policies.
- Import and export of certificates.
- Print status and content of the key and certificate store.

An example output of the listing looks are:

```
Loading certificatestore
Private Certificateid  isssuerid Attribute   Policy
  X     RKtCjQG        RKtCjQG   nda         \\
RKtCjQG nda:http://www.krogh.cc/attributes/non-disclosure  - \\
Certificate verified

Loading keystore
Keyid: RKtCjQG
```

The first lines tell that the certificatestore contains the key with key id "RKr-CjQG", which is "selv-signed". This contains a "nda" attribute with a $\mathcal{RT}$ rule attached. This rule defines that the certificate may be disclosed only the certificate itself. The situation doesn't make much sense, but illustrates how the system works.

A complete helpfile is included in the README file with the source code for the system.

5.3 Service level agreements

The Service level agreements are implemented in the package "Sla". This is the service level agreement as described in the design chapter. This represents an XML-document, that describes the actual "service" that has been exchanged between the entities. It is important that it describes the service and not any specifics of the interaction. An example could be the GridSla, that consists of two components. The DiskSla that describes the actual disk consumption and the CpuSla that describes the CPR consumption. These are sufficient specific to address the various types of interactions but leaks no evidence of the actual interaction. Thereby the privacy issues of the information are reduced. In the following is a DiskSla-example:

```
<disk:sla xmlns:disk="http://www.krogh.cc/sla/disk">
  <disk:space>20</disk:space>
  <disk:time>30</disk:time>
  <sla:genericsla xmlns:sla="http://www.krogh.cc/sla">
    <sla:uuid>9e9b6662-0350-4a49-95cd-751fabae0cb7</sla:uuid>
  </sla:genericsla>
</disk:sla>
```

This it the XML-representation of a DiskSla where, the uuid is a globally unique identifier generated when the Sla was created. This enables the evidence evaluation to see if a piece of evidence has been seen before. The interface that needs to be implemented in order to introduce a new Sla in the system are:

```
public String toString();
public Namespace getNamespace();
public Document asXmlDocument();
public void setXmlElement(Element elm) throws InvalidSlaXmlException;
public boolean equals(Sla s);
public String getUuid();
public double strength(Sla s);
public byte[] digestData();
```

The implemented service level agreement inherits the general functionality from the GenericSla class and implements the interface.

5.4 Trustcore and evidence evaluation

The Trust and evidence evaluation system are implemented in the "TrustCore" package. The trustcore, evidencestore and evidence evaluation relies on the certificate infrastructure describes in the former section. The trustcore of the system is the part that take care of recognising principals, storing evidence.

Outcome	Weight
success	2
failure	-5
dontknow	-1

Table 5.2: Map representing the implemented outcomes.

5.4.1 Principals

The core "entity" in the system are a Principal (Implemented in Principals.java) , which is merely an association of certificates proven to be held by the same entity. This represents a subjective view on every principal in the system and this association contains no information that can be used to reveal the association to 3'rd parties. This is if P1 has C1 and C2, P2 has seen C1 and C2 in a earlier interaction and "knows" that P1 has these. P2 has no way of proving to P3 that C1 and C2 belongs to the same entity, without giving the private keys of its own certificates to P3. There is a basic assumption that private keys never are exchanged.

If a set of certificates (C1,C2,C3) are seen in one interaction and (C3,C4,C5) are seen in a subsequent interaction. The principal is known to hold all five certificates, since the proof of C3 is in both interactions.

5.4.2 Outcome

The implementation of an "Outcome" are very trivial. This is represented by an XML-document with one element containing a string with the representation of the outcome. Each service level agreement has a distinct set of potential outcomes. It is up to the evaluating entity to provide a mapping between the outcomes and their given weights in the evaluation situation. Outcome are thus specific to each service level agreement all though the three outcomes in table 5.2 are implemented and used for all service level agreements in the prototype implementation.

5.4.3 Evidence

The system implements evidence as signed service level agreements. This service level agreements are signed by the "consumer" of the interaction. The consumer is the one that benefits from the interaction. The evidence class consists of the producer and consumers public keys. The consumer issues a signature on on the service level agreement where the signed information is a hash of:

$$\begin{aligned} digest &= Hash(date, outcome, sla, producerkey, consumerkey) && (5.5)\\ signature &= E(Priv_{consumer}, digest) && (5.6) \end{aligned}$$

The evidence it the said to be “directed” from the consumer to the producer. The following example is a concrete piece of evidence XML.

```
<evidence:evidence
   xmlns:evidence="http://www.krogh.cc/trustcore/evidence">
     <    SLA-XML HERE >
     <evidence:date>2005-07-29</evidence:date>
     <evidence:outcome>success</evidence:outcome>
     <evidence:producer>
       <    PRODUCER PUBLICKEY HERE >
     <evidence:consumer>
       <    CONSUMER PUBLICKEY HERE >
     </evidence:consumer>
  <evidence:signature
    encoding="base64"
    algorithm="SHA512withRSA">NTes98/omwzb.. =</evidence:signature>
</evidence:evidence>
```

The reasoning behind this model of a evidence, it that the interaction concerns the same thing, no matter which outcome there will come from the interaction. Thus the description (SLA) of the interaction is negotiated (or decided) before the interaction, whereas the actual outcome is first known after.

5.4.4 Evidence evaluation

The evidencestore implements the evidence evaluation system with a few additional classes. The EvidenceEdge and EvidencePath classes.

In order to start the evidence evaluation an strengthfunction needs to be supplied. This strengthfunction can map the strength of a evidence into the interval $[0, \infty[$

The first step in the evidence evaluation is to join the information from the principalstore with the information from the evidencestore and make the combined evidence between two principals.

The pseudo code for this is:

```
for all evidence,e in evidencestore
  prod = lookup_in_principalstore(producer(e))
  cons = lookup_in_principalstore(consumer(e))
  weight = lookup_in_outcomemap(outcome(e))
  add weigth*strength(e) to evidenceegde(cons,prod)
done
```

The evidenceedge is implemented in the "EvidenceEdge"-class in the TrustCore package. This gives the sum of all evidence between two principals. It is important to notice that this is done within the view of the evaluating principal, thus the set of principals evaluated are those known to the local principalstore. All other evidence would show up as "principals identified by a single publickey".

The next step is to find all path from the evaluating principal(self) to any principal in the graph.

```
find all paths from self -> target
initialise path-set with path starting with ''self''
for all paths in path-set
  for all edges in edge-set
    if edge matches end of path and not in path all-ready
     and end of path not matches target
       create new path with the edge in the end
  if end of path not matches target
     remove path from path-set
repeat until no paths are updated
```

The evidencepath is implemented in the "EvidencePath"-class in the TrustCore package. This pseudo-code finds all paths to a given entity in the evidence graph. This is done for all entities under consideration. The next step is to normalise the edges into the interval $[0, degradefactor]$. The degrade factor is a user-supplied configuration value that represents the penalty for being "one step away" or being a "recommendation" instead of direct evidence. This is done by this pseudo-code.

```
for all edges in all paths
```

```
   set min to the minimum value
   set max to the maximum value

if the minimum value is greater than zero
  set it to zero.

set size to the difference of min and max.
for all edges in all paths
  add abs min to edge
  divide by size
  multiply by degradefactor
```

This makes the normalisation of all values in the graph, thus not taking evidencepath that does not lead to any principals not in consideration. The actual ordering are then done by the inclusion-exclusion principle described in the design chapter.

5.5 Trust negotiation

The trust negotiation layer has been implemented in the XmlIo package. The trust negotiation layer should be as transparent to a usual socket connection as possible. This is done by substituting the normal SSL/TLS layer of the OSI reference model with a trust negotiation layer. The keyexchange are implemented as a separate class, that can be dumped in a transport-safe XML-document.

The complete trust negotiation has been implemented in the XmlIo java package. This consists of the following classes.

- **KeyExchange -** Implementation of the negotiation packages described in the design.

- **XmlServerSocket -** This class inherits the ServerSocket class from Java.

- **XmlSocket -** This class inherits the Socket from the Java this is where the actual negotiation is being done.

5.5.1 XmlServerSocket

The XmlServerSocket is very simple. It inherits the ServerSocket from Java thereby making sure that the socket instantiated when a new connection is accepted is an XmlSocket. The actual code is short enough to be shown here:

```
public class XmlServerSocket extends java.net.ServerSocket
{
    public XmlServerSocket() throws IOException{
        super();
    }
    public XmlServerSocket(int port) throws IOException{
        super(port);
    }
    public Socket accept() throws IOException {
        XmlSocket socket = new XmlSocket();
        implAccept(socket);
        return socket;
    }
}
```

The implAccept call is used to tell the ServerSocket framework that it is not the standard socket that should be returned but a subclass of it.

5.5.2 XmlSocket

The XmlSocket class implements the complete trust negotiation system described in the design chapter. The blocks thats sent over the network are implemented in the class KeyExchange. This is the message-types from table 4.3.

The keyexchange follows the CSP specification described in the design chapter. Both client and server need to be configured with a initial Policy, that may be empty.

At the lowest level of the XmlSocket there are two routines readDocument and write that takes care of the low level I/O from the socket. The current implementation can only transport XML-documents. These routines take care of the encryption of traffic (write) and decryption (readDocument). Every time a new symmetric key is generated and send to the recipient the readDocument method is updated to only accept incoming data encrypted using this key.

All certificate received from the other peer is collected into a "Principal" and the trust negotiation continues until this principal satisfies the local policy (Success) or there are no more certificates to disclose and the policy has not been satisfied (fail).

The certificate disclosure strategy is very naive implemented as a first found policy. The pseudo-code looks like:

```
find-certificate
  for all certificates in certificatestore
    if certificateowner is myself and certificate has not been disclosed \\
      and disclosure policy is satisfied by remote principal.
         stop search and use this certificate.
```

The trust negotiation code are described in the design chapter and are implemented according to that.

5.6 Grid and Indexserver stubs - Overall test

There has been implemented test of the various individual modules in most classes in TrustCore, XmlIo and XmlCert. Either as a "main" method directly in the classes of as a separated implementation name; ClassNameTest.

The Grid and IndexServer packages serves as overall test for the system. The IndexServer provides resource registration and discovery for the system and the IndexServerClient implements an interface to register and query the server register.

5.6.1 The Grid Resource

The class Resource in the Grid package implements a basic resource in the "grid" system. In order to use the system both client and resource needs to have the appropriate certificatestore, keystore and principalstore available. It is only the client that does the evidence evaluation since it it only the client that actually

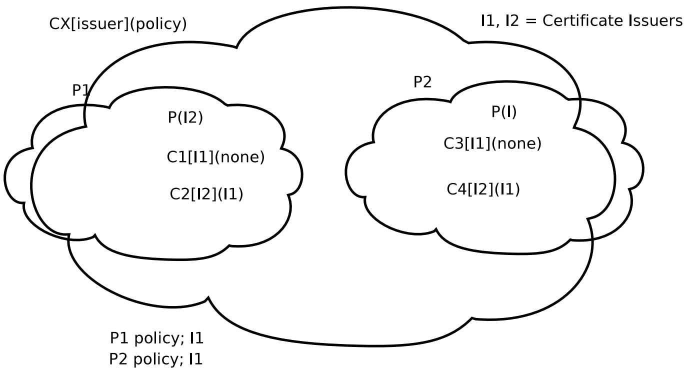

Figure 5.1: Test scenario. All certificates are of the same type. Both parties needs to disclose both certificate to get a successful negotiation.

faces an “ambiguous path”. The current resource implementation “echo” the XML-document send over the socket when a successful negotiation has occured.

5.6.2 Test of the negotiation layer

The trust negotiation layer has been tested using the example described in figure 5.1. This situation has been verified successfully and the negotiation succeeded as intended.

Several situations with unsuccessful negotiations due to unsatisfied certificate policies has been tested an found to work as intended.

5.7 Summary

The system described in the design chapter are fully implemented and unit-test to a certain extend has been performed during development. Most components contains an internal or external test-routine.

The overall systemtest has been done by developing a grid resource-class a grid client-class and an indexserver that facilitates the resource discovery for the system. Given a set of policies and certificates that can satisfy them the system can complete a negotiation.

Package	Lines of code
Grid	442
TrustCore	1492
XmlCert	2316
XmlIo	666
IndexServer	341
Sla	605

Table 5.3: System size in terms of code-lines (with comments).

The system totally consists of 104 classes (including exceptions) shared in in the 6 packages. The size can be seen in table 5.3.

CHAPTER 6

Conclusion and future work

This chapter summarises the report in the results obtained form the prototype implementation and conclusion. The last section describes the future work.

6.1 Conclusion

Existing systems like the X.509 or PGP system enables secure collaboration by mapping a real world identity into the system. This approach is feasible in a system where there is a need for this high level of mutual trust between all participants.

Collaborations in the real world does not have this high demand for identification all the time thus this is a significant restriction to the types of collaboration that can take place.

The certificate infrastructure designed and implemented (XmlCert) allows entities to bind any type of properties to a public key, this allows participants to form certificate chains of any type within the system.

The reliance on several certificates of various importance makes it harder for entities to track the behaviour of a particular identity and the separation of certificate and certificate content allows entities to disclose the the belonging of a certificate without disclosing the actual certificate content.

The Certificate negotiation protocol enables entities to prove the combined ownership of a set of certificates within an interaction. This is done in a secure manner that lets the disclosing entity in control of the certificate by defined disclosure policies on the certificate to protect possible sensitive certificates from disclosure. The knowledge of the combined ownership of certificates is not transferable, thus the only entities that has knowledge of the combined set of certificate are the entities that has been directly exposed to the certificates during a certificate negotiation.

This leaves the certificates holder in complete control as much as possible. This resembles the state in the real world where the subject is in control over which information that interacting entities should be allowed to have access to.

It is rare for people in the real world to have complete knowledge of the entities they collaborate with. This notion of a "view" on an entity has likewise been implemented in the Principal that for each entity within the system contains a limited view of other entities.

On top of the system a notion of trust based on evidence and recommendations has been implemented. Evidence is implemented in a generic way that allows to aggregate similar evidence. The diffuse notion of principals allows distribution of evidence in a way that minimises the privacy concerns of the evidence distribution.

The evidence aggregation system has been designed to resemble several features found in the real world evidence and recommendation system.

- Evidence and recommendations looses value over time.
- Evidence and recommendations are highly dependent of context.
- Recommendations can never resemble more trust than the entity that issues the recommendation can be attributed.
- More positive evidence makes the entity more "trustworthy".
- More negative evidence makes the entity less "trustworthy".
- Several independent recommendation are taken into account.

The evidence aggregation system has done in a manner that neglects to reason about the value of peers in an absolute scale but only implements a complete ordering of the peers available to a specific application scenario.

As a whole, the system proposed, tries hard to resemble the human notion of trust in a distributed computing context.

The system has been tested and found to work as expected. This has been done through limited test cases that only proves the that the correctness and functionality of the system thus leaves large scale testing as future works.

6.2 Future works

The system implements two essential mechanisms for reasoning human-like trust in a distributes computing network (grid-system) but many details are left out. The most important are listed here.

6.2.1 Reprocication

In a system where evidence and trust values are used to model behaviour, the notion of Reprocication need to be investigated. It seems intuitive that Reprocication would make more peers willing to share and not consume, thereby raising the overall performance and utilisation of the system.

6.2.2 XML implementation

- The XML implementation of the complete system is very good for development. This enables easy and quick modifications of the system. The verbosity of XML and demands for a proper XML-parser makes it hard to use in “small devices”. This is not a show stopper for the system, since the reliance on XML is not a core part if the system. If the fully qualified name spaces used the complete system can be made inter operable with just about any other “transport format” available.

6.2.3 Malicious entities publishing private keys

The system could contains very malicious entities that are willing to trade there own private keys with others. This is a basic assumption in the system that no-one does that. There should be taken countermeasures to ensure this.

6.2.4 Event structure as outcome

- The “Outcome” implementation should be replaced by an Event structure as described in the SECURE-framework. The Event structure represent a more real world like, representation of the outcome of interactions. This current implementation represents the situation where there is one distinct outcome of each interaction. The system can trivially be extended to use this instead.

6.2.5 Advanced roles

- The $\mathcal{RT}$ policy language describes advanced roles as parametrised roles and roles that describes situations where several different certification paths can be chosen. The system can trivially be extended to do this.

6.2.6 Verification of system

- The system need to be verified by other people. This is an essential part of a the security in a security framework that it can pass a security review. The actual implementation should likewise be reviewed.

6.2.7 Complexity of evidence aggregation

- The complexity of the evidence aggregation algorithm is quite high. An analysis on the impact of trivial optimisations such as limiting the maximum path length or dropping paths under a certain threshold should be explored.

Bibliography

[1] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The KeyNote Trust-Management System Version 2. RFC 2704 (Informational), September 1999. http://www.ietf.org/rfc/rfc2704.txt.

[2] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. Technical Report 96-17, 28, 1996. citeseer.ist.psu.edu/blaze96decentralized.html.

[3] Marco Carbone, Karl Krukow, Mogens Nielsen, and Andrew Twigg. Secure - validation of operational trust model. 2004. http://www.dsg.cs.tcd.ie/uploads/category243/188.pdf.

[4] Dwaine Clarke, Jean-Emile Elien, Carl Ellison, Matt Fredette, Alexander Morcos, and Ronald L. Rivest. Certificate chain discovery in spki/sdsi, 1999. To be published, November 1999.

[5] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing, 2001. citeseer.ist.psu.edu/czajkowski01grid.html.

[6] M. Deutsch. Cooperation and trust: Some theoretical notes. 1962.

[7] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), January 1999. Updated by RFC 3546.

[8] John R. Douceur. The sibyl attack, 2002.

[9] M. Elkins, D. Del Torto, R. Levien, and T. Roessler. MIME Security with OpenPGP. RFC 3156 (Proposed Standard), August 2001. http://www.ietf.org/rfc/rfc3156.txt.

[10] C. Ellison. SPKI Requirements. RFC 2692 (Experimental), September 1999. http://www.ietf.org/rfc/rfc2692.txt.

[11] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. RFC 2693 (Experimental), September 1999. http://www.ietf.org/rfc/rfc2693.txt.

[12] Nataraj Nagaratnam Philippe Janson et al. Security architecture for open grid services, July 2002. http://www.cs.virginia.edu/ humphrey/ogsa-sec-wg/.

[13] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration, 2002. citeseer.ist.psu.edu/foster02physiology.html.

[14] Ian Foster. What is the grid: A three point checklist. 2002. http://www.mcs.csuhayward.edu/~tebo/Classes/6580/papers/WhatIsTheGrid.pdf.

[15] A. Hess, J. Jacobson, H. Mills, R. Wamsley, K. Seamons, and B. Smith. Advanced client/server authentication in tls, 2002. citeseer.ist.psu.edu/hess02advanced.html.

[16] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280 (Proposed Standard), April 2002. http://www.ietf.org/rfc/rfc3280.txt.

[17] Ian Foster; Carl Kesselman. *The Grid 2, Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.

[18] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust management framework. In *Proc. IEEE Symposium on Security and Privacy, Oakland*, May 2002. citeseer.ist.psu.edu/533810.html.

[19] Ninghui Li, William H. Winsborough, and John C. Mitchell. Distributed credential chain discovery in trust management: extended abstract. In *ACM Conference on Computer and Communications Security*, pages 156–165, 2001. citeseer.ist.psu.edu/article/li01distributed.html.

[20] Stephen Paul Marsh. *Formalising trust as a Computational Concept*. April 1994. Department of Computing Science and Mathematics, University of Stirling.

[21] Ueli Maurer. Modelling a public-key infrastructure. In *ESORICS: European Symposium on Research in Computer Security*. LNCS, Springer-Verlag, 1996. citeseer.ist.psu.edu/maurer96modelling.html.

[22] Charles P. Pfleeger and Shari Lawrence Pfleeger. *Security in Computing*. Pearson Education, 2002.

[23] K. Seamons, M. Winslett, and T. Yu. Limiting the disclosure of access control policies during automated trust negotiation, 2001. citeseer.ist.psu.edu/seamons01limiting.html.

[24] Keent E. Seamons, Marianne Winslett, Ting Yu, Lina Yu, and Ryan Jarvis. Protecting privacy during on-line trust negotiation, 2002. citeseer.ist.psu.edu/594256.html.

[25] Jen-Marc Seigneur, Stephen Farrell, Christian Damsgaard Jensen, Vinny Cahill, Elizabeth Gray, and Chen Yong. Secure - entity recognition scheme. January 2003. http://www.dsg.cs.tcd.ie/uploads/category243/49.pdf.

[26] Robin Sharp. *Principles of protocol design - Draft second edition.* 2002. IMM / Informatik og Matematisk Modellering.

[27] IanFoster; Carl Kesselman; Steven Tuecke. *The Anatomy of the Grid, Enabling Scalable Virtual Organization s.* Intl J. Supercomputer Applications, 2001.

[28] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. RFC 3820 (Proposed Standard), June 2004. http://www.ietf.org/rfc/rfc3820.txt.

[29] Ninghui Li W. H. Windsborough. Protecting sensitive attributes in automated trust negotiation, 2002. lhttp://mason.gmu.edu/~wwinsbor/index_files/Papers/sensitiveAttributes_WPES.pdf.

[30] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yo. Finding collisions in the full sha-1, 2004. http://theory.csail.mit.edu/ yiqun/pub.html.

[31] W. Windsborough. An introduction to trust negotiation, 2003. http://dais.cs.uiuc.edu/pubs/winslett/itrust03.ps.

[32] W. Winsborough and N. Li. Towards practical automated trust negotiation, 2002. citeseer.ist.psu.edu/winsborough02towards.html.

[33] Ting Yu, Marianne Winslett, and Kent E. Seamons. Interoperable strategies in automated trust negotiation. In *ACM Conference on Computer and Communications Security*, pages 146–155, 2001. citeseer.ist.psu.edu/yu01interoperable.html.

Appendix

.1 Article - Security in open Grid systems

.2 Sourcecode