Trust Evaluation for the Grid

David O'Callaghan

A thesis submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Doctor of Philosophy

April 2007

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this or any other University, and that it is entirely my own work. The Library may lend or copy this thesis upon request.

David O'Callaghan

Dated: April 12, 2007

Abstract

This thesis presents models and techniques for automatic evaluation of trust in certification authorities for grid computing infrastructures. The desire for automatic trust evaluation arose out of the need to assess the growing number of certification authorities as they join the grid infrastructure and periodically thereafter.

Public key infrastructure is widely used as the basis for authentication in grid middleware. Grids must decide which certification authorities to trust. The International Grid Trust Federation (IGTF) develops authentication policies and accredits certification authorities that meet their requirements. There has been previous research into trust evaluation but it has not met the needs of the grid community.

The thesis introduces a semi-formal model for an evaluation of trust in a certification authority's policies and practices against a set of rules based on the policies of a relying party. The model includes algebras which specify the meaning of the evaluation results for a set of rules. The model supports chaining of multiple sets of rules that reflect, for example, the chain of organisations from the IGTF to the relying party.

This model is a suitable basis for the design of software to provide on-demand trust evaluation. A trust evaluation system was implemented by the author – partially in a functional programming style – as a proof-of-concept of the model. The central component of the design is a trust evaluation engine. Two applications of the engine, graphical trust matrices and online validation services, are considered in some detail. The performance of the implemented components was found to be acceptable for their intended uses.

Establishment of trust is the first step towards interoperability between different grid middlewares. A scenario has been explored, involving a Metagrid system that would allow secure interoperability between various grid and workflow middlewares. A number of Metagrid security components have been designed and implemented as prototypes.

Suggestions for future research arising from this thesis include formal development of the trust evaluation model and software, the application of the engine to a variety of evaluation domains, and the closer integration of trust evaluation with other services for security interoperability.

Acknowledgements

I would like to acknowledge the support and encouragement of my supervisor Dr Brian Coghlan. He also provided me with many opportunities to make contacts in the grid world.

I have been fortunate to be a participant in some of the seminal European (EDG, LCG, CrossGrid, EGEE) and national (CosmoGrid, WebCom-G) grid projects during my postgraduate studies. In addition, I have acted as deputy manager of the Grid-Ireland Certification Authority, VOMS manager for Grid-Ireland, and have attended CACG and EU Grid PMA meetings for Grid-Ireland. These projects have enabled me to get a good understanding of current grid security practices and provided many opportunities to discuss my work with experts in the field. The knowledgable members and other attendees of EU Grid PMA meetings have been very helpful as a source of ideas and encouragement in the pursuit of automatic trust evaluation.

I would like to thank Science Foundation Ireland for their support in funding this research as part of the WebCom-G project.

The members of the Computer Architecture Group, in particular those involved in the grid research area, have been another source of questions and suggestions which have helped my research. I have enjoyed participating as a Social Agent in the Pub Model.

My parents William and Margaret, my sister Liz, my friends Nick and Laura, and, not least, my newly acquired in-laws have all helped to keep me on the right track. A collective thank you to all the others I have neglected to mention.

Finally, I would like to thank my wife Joan for all her love and support – and for agreeing to marry me – during the course of this research. This thesis is dedicated to her.

David O'Callaghan
University of Dublin, Trinity College
April 2007

Contents

Chapter 1

Introduction

1.1 Motivation

Trust and security have been crucial components of grid computing since its inception. The main aim of grid computing is to enable secure sharing of resources between organisations and therefore across administrative domains. As grids expand and bring in new users and resources the number of trust domains also increases. Public key infrastructures (PKIs) often form the basis for trust in grids on an organisational and national level. The efforts of standards bodies have helped PKI for the grid to become interoperable internationally.

The common policies that have been established allow consistent assessments of grid certification authorities (CAs) to be made and allow relying parties to trust all CAs accredited as meeting the profile. However the process of assessment is entirely manual and some automation is desirable. This would help in the existing assessment process for new CAs but also in the regular re-assessment of all CAs. The focus has often been on the addition of new CAs but the maintenance of trust in previously approved CAs is vital.

There is a desire to make use of existing identity and authentication services for grid authentication purposes, and a system for automatic trust evaluation could help in this regard by establishing equivalence between different classes of authentication system. Automatic trust evaluation would also allow trust policies to be tailored to the requirements of relying parties. This would allow a level of policy-based control over the extent to which authentication providers are trusted by a virtual organisation or a grid site.

Any system for automatic evaluation of trust will have to be trusted itself. For this reason it should be based on a clearly defined model, and the use of formal methods would further enhance trust. It should also take into account practical aspects of performance as its use should not introduce an excessive overhead.

1.2 Overview of the Thesis

This thesis aims to address the issues outlined above. The next chapter introduces public key infrastructure with reference to its use in grids (§2.2); presents the state of the art in grid computing and in grid authorization and authentication (§2.3); and online PKI services (§2.4); and presents the foundations of computational trust (§2.5). The role of an automatic trust evaluation system in grid authentication is discussed (§2.6).

In Chapter 3 the concepts and principles of trust evaluation are introduced. A semi-formal model of the evaluation (§3.2) is used to discuss a number of evaluation algebras (§3.4.2) and an approach to policy chaining (§3.5).

In Chapter 4 the design and implementation of a trust evaluation system based on the ideas from the preceding chapter are described, including a trust evaluation engine, graphical trust matrices, and online validation services.

The system is evaluated in Chapter 5 through a number of experiments which assess the performance and hence practical usability of the system.

Chapter 6 considers the issues that remain to be tackled in order that grids can interoperate, once trust has been established.

Chapter 7 discusses in some detail the contributions of this work and the possible future directions for research.

Chapter 2

Grids, Security & Trust

2.1 Introduction

Grid computing represents the vision that a user will be able to simply 'plug in' to get access to computing power as needed – an analogy to the electrical power grid [41]. The definition of a grid has expanded over time to cover any system for large-scale computing, network and data resource sharing, particularly across multiple domains of administration and security. Grids are intended to allow "flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources" [78]. Foster gives a grid check-list which states that a grid is a system that:

1. coordinates resources that are not subject to centralized control ...
2. ... using standard, open, general-purpose protocols and interfaces ...
3. ... to deliver non-trivial qualities of service.

(taken, in edited form, from [76])

Two other broadly accepted definitions of a grid are given by the Open Grid Forum (OGF)[1] and the CoreGRID Network of Excellence. The OGF definition of a grid is:

> "A system that is concerned with the integration, virtualization, and management of services and resources in a distributed, heterogeneous environment that supports collections of users and resources (virtual organizations) across traditional administrative and organizational domains (real organizations)." [83]

[1]formerly known as the Global Grid Forum or GGF

The CoreGRID definition states that a grid is:

> "A fully distributed, dynamically reconfigurable, scalable and autonomous infrastructure to provide location independent, pervasive, reliable, secure and efficient access to a coordinated set of services encapsulating and virtualizing resources (computing power, storage, instruments, data, etc.) in order to generate knowledge." [162]

These definitions are taken from a survey of grid experts [162]. The conclusion of the survey report is that among grid researchers there is a general consensus on what constitutes a grid. The most important characteristics of grids include collaboration, aggregation and virtualization of resources, service orientation, heterogeneity, decentralized control, standardization and interoperability, access transparency, scalability, reconfigurability, and security. This thesis touches to a greater or lesser extent on many of these characteristics of grids.

2.1.1 Overview

This chapter presents the state of the art in grid computing and its existing authentication and authorisation systems, and gives an introduction to the field of computational trust.

Public Key Infrastructure (PKI) forms the basis of the grid authentication systems of greatest interest in this thesis. Section 2.2 introduces PKI with reference to its use in grids.

At present there is a wide variety of grid software and standards, of distributions and definitions. In the field of grid security, in particular authentication and authorisation, there are some common approaches but there is still much diversity. Section 2.3 presents the state of the art in grid computing, and in grid authentication and authorisation.

There has been some previous work in the area of on-line services in PKI which extend or enhance the standard PKI validation mechanisms. In addition there has been work done in the area of trust evaluation of CAs. These areas of research are are presented in Section 2.4.

Computational trust is a relatively young and currently very active area of research. The crossover between the more traditional cryptography-based trust in PKIs and the more experimental form of trust based on reputation, rules and evidence is something that should be explored. Section 2.5 presents the foundations of computational trust.

In conclusion, Section 2.6 discusses the role of an automatic trust evaluation system in grid authentication.

2.2 Public Key Infrastructure

Public Key Infrastructure lies at the heart of many security systems on the Internet and on grids. In essence, PKI allows public cryptographic keys to be bound to identities. Entities identified in a PKI can then be assigned privileges or roles. By way of introduction to the topic of PKI, the relevant aspects of cryptography are presented here.

2.2.1 Cryptography

Cryptography is the mathematics of secret writing. Cryptographic techniques are of interest because they can be used to create secure communication and identification mechanisms. Cryptography can be divided broadly into two categories based on the properties of the keys used: *symmetric-key* and *asymmetric-key*, or *public-key*, systems.

Symmetric-Key Cryptography

With a symmetric-key cryptography system a single key is used for encryption and decryption of messages. To use symmetric-key cryptography to encrypt communications the key must be known by all parties to the communication. This leads to the problem of key distribution. There are a number of solutions to this problem, some of which are based on public-key cryptography, described below. Examples of symmetric ciphers include AES [127], DES [128] and Blowfish [150].

Public-Key Cryptography

In a public-key cryptography system, distinct keys with a mathematical relationship to each other are used for encryption and decryption of messages. A message encrypted with the *public key* of the recipient can only be decrypted with the recipient's corresponding *private key.* Conversely, a message encrypted with the private key can be decrypted with the corresponding public key alone and so it can be verified that the message must have originated with the holder of the private key. Public keys can be shared widely while private keys must be kept secret. Public-key cryptography forms the basis for digital signatures and certificates. Examples of asymmetric-key systems include RSA [146], DSA [129] and ElGamal [63].

Symmetric-key operations are typically much faster than asymmetric-key operations offering a comparable level of cryptographic strength. For this reason symmetric-key and public-key cryptography are often combined in protocols and applications to take advantage of the strong points of each.

Cryptographic Hash Functions

Hash functions play an important role in cryptographic systems. Digital signatures are typically created by encrypting a hash of a message with the sender's private key since encrypting the short fixed-length hash is more efficient than encrypting a potentially long message. Cryptographic hashes are used in their own right to check the integrity of messages or files.

Cryptographic hash functions are designed to be resistant to retrieving the message from the hash, to finding another message that gives the same hash, or to finding two messages which produce the same hash. Commonly used cryptographic hash functions include MD5 [145] and SHA-1 [130], although weaknesses have been shown in both of these in recent times (see, for example [176]).

The basics of cryptography have been introduced above. The techniques described in the following sections allow an identity to be securely bound to a public key pair, thus creating a basis for authentication. A *web of trust* provides this ability in a distributed fashion while a *certification authority* takes a more centralised approach.

2.2.2 Web of Trust

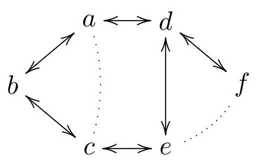


Figure 2.1: Web of Trust
Solid lines represent verification and dotted lines represent indirect trust.

One approach to binding identities to keys is a *web of trust*. In a web of trust, public keys are published along with the details of their owner (typically name and email address). To verify the link between the key and the identity an interested person can contact the owner 'out-of-band' and exchange information to verify the owner's identity and that they hold the key-pair in question. For example, the interested party might arrange to meet in person with the key owner and check some identity document, such as a passport, and that the owner's public key (or its 'fingerprint') matches the published version. If the interested party is satisfied with the information provided they can then use the owner's public key to send private communications.

If this interested person also has a key pair then they can cryptographically sign the public key of the first key-owner to attest that they have verified the link between identity and key-pair. A third key-owner who has previously verified the second can choose to use this signature to acquire trust in the identity of the first key-owner. As more key-owners verify the identity of others and sign their public keys, a web of trust is formed. Key-signing parties may be arranged to allow large groups to cross-sign each others keys in this manner.

This system has the advantage of being distributed and decentralised. No central authority is needed to verify identities or to maintain the web as a whole. However, these can also be considered as disadvantages: rogue members can publish or sign unverified or fraudulent information, and it is not clear that trust should be purely transitive over a large number of hops in the graph.

The web of trust concept is implemented in software such as PGP [180] and other software that follows the OpenPGP standard [33], such as GnuPG [13].

2.2.3 Certification Authority

In contrast with a web of trust, a *certification authority* (CA) is a central authority that issues verified attestations of the bindings between identities and a public keys, known as *public key certificates*. A CA acts as a *trusted third-party* which facilitates secure communications between others. The common standard for this form of certificate is X.509 [96].

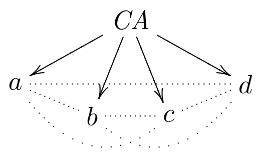


Figure 2.2: Certification Authority and Subscribers
Solid lines represent verification and dotted lines represent indirect trust.

The CA must assure relying parties that it is run as a high quality service – that it can be trusted. A central authority like this has the advantage that it can be trusted more widely than a single entity in a web of trust as verification should be consistent for all certificate holders, whereas in a web of trust the verification depends very much on the particular member performing the verification. However, the CA is of course a large single target for attackers: the certificates of a compromised CA are worthless, whereas a single compromised signer in a web of trust does not necessarily affect the entire web.

CAs typically have a number of *registration authorities* (RAs) who perform the verification and registration of new subjects on behalf of the CA. The RAs operate according to policies specified by the CA. A widely distributed network of RAs has the advantage of allowing a CA to operate over a large area. However, delegating the responsibility of verification does tend to re-introduce some of the risks of a web of trust, as a rogue RA may subvert the CA.

A typical process for issuing a new certificate is as follows.[2] First, the subscriber will securely generate a new public/private key pair. The subscriber communicates the public key to the CA along with out-of-band proof of identity, and demonstrates proof of possession of the private key by digitally signing a message, for example. Once the subscriber's identity and its link to the public have been verified (often by an RA) the CA can issue the certificate binding the identity to the public key. The certificate is signed with the CA's private key and this allows the certificate to be verified with the CA's public key. The subscriber is given a copy of the certificate and the CA may also publish it in some form of repository.

Revocation

If a subscriber's private key is lost or stolen or otherwise *compromised* then the bond between the subscriber's identity and the public key has been broken as it is potentially possible that another person could use the private key to impersonate the subscriber. In such a case, the CA should

[2]It should be stated that this is an *example* process influenced by the policy requirements of the IGTF outlined in Section 2.2.6.

revoke the certificate in question to indicate that it is no longer to be trusted. Other occasions for revocation are when the data asserted in the certificate become invalid, when an employee leaves a company, for example, or when a subscriber no longer has a need for their certificate.

Certificate Revocation Lists

When a certificate has been revoked the CA must make available this information so that relying parties can make use of it and block the use of the revoked certificate. The primary mechanism for publishing revocation information is the X.509 Certificate Revocation List (CRL) [96, 92]. A CRL is produced by a CA whenever a certificate is revoked. It contains a list of the serial numbers of certificates that have been revoked by the CA. If the CA issues certificates intended for authentication then the CRL need not include entries for revoked certificates which have already expired. Where historical revocation information is required the CRL may contain all past revocations. The CRL is signed by the CA so that its authenticity can be verified. Delta CRLs can be used to indicate the changes since the last full CRL was issued.

If a CA publishes CRLs then a relying party must regularly download (potentially large) files in order to have up-to-date revocation information. In order to minimise the risk of accepting a revoked certificate the relying party should, at least in theory, update the CRL before every verification. In practice in the grid community it is common to download CRLs several times per day.

On-line Certificate Status Protocol

An alternative or complementary approach to revocation is provided by an on-line service using, for instance, the On-line Certificate Status Protocol (OCSP) [125]. Instead of downloading a full CRL, a relying party sends a request for the status of a particular certificate (or set of certificates) to an OCSP *responder*. The OCSP responder signs its responses with the CA's private key or that of another authorised key pair. The main advantage of such a system is that potentially more up-to-date information is provided and a lower bandwidth is used compared with CRLs (depending on the pattern of revocation checking). The disadvantage is that the responder must be highly available as many relying parties will depend upon it and any network disruption would leave them unable to verify certificates.

Policies & Practices

A CA's policies and practices determine the quality of the service it provides and its overall trustworthiness. While the cryptographic strength of the CA's keys and ciphers is certainly an important consideration, it is relatively straightforward to choose reliable and well-tested algorithms and software implementations. What is more difficult is to set out policies and practices which will ensure that, as an overall system, the PKI established by the CA is at least as secure as necessary.

A CA's policies and practices are often described in the form of a *certificate policy* (CP) and certification practice statement (CPS). A detailed description of the purpose of these documents, recommendations for their contents and for their structure are found in RFC 3647 [42]. According to the X.509 standard (and via RFC 3647) a CP is "a named set of rules that indicates the applicability of a certificate to a particular community and/or class of applications with common security requirements." According the American Bar Association (again via RFC 3647) "a CPS is a statement of the practices which a certification authority employs in issuing certificates." The RFC explains that a CPS may cover the entire certificate lifetime and other business and technical issues related to the CA. While the CP sets out what the CA and subscribers must do, the CPS describes how these policies are implemented. Among the grid CA community, these two documents are often combined into a single CP/CPS document.

The CP in force when a certificate is issued can be recorded in the certificate in the form of an *object identifier* (OID, a unique value that identifies the document globally) stored in a Certificate Policies extension.

The framework sets out nine components for a CP or CPS:

1. Introduction
2. Publication and Repository
3. Identification and Authentication
4. Certificate Life-Cycle Operational Requirements
5. Facilities, Management, and Operational Controls
6. Technical Security Controls
7. Certificate, CRL, and OCSP Profile
8. Compliance audit
9. Other Business and Legal Matters

(taken from [42])

A detailed outline is also provided. The common structure allows convenient comparison of policies. The framework describes in some detail what information should be provided in each sub-section of the document.

2.2.4 Authentication Policy Management

In the above description of a Certification Authority as a central authority for identity verification there is an assumption that there is only one CA for a particular community of relying parties. In practice it is common and even desirable to have multiple CAs.

At the start of the European DataGrid project (EDG) [82] in 2001 it was considered necessary to create a large international X.509 public key infrastructure for grid authentication. The choice of PKI for this was due in large part to the requirements of existing software. The *Certification Authority Coordination Group* (CACG) was established by EDG to coordinate its operation. The member CAs covered many of the EU member states and also Canada, Russia and the USA.

During the lifetime of EDG, several other international projects such as CrossGrid [84], DataTAG [52], and GridLab [152] adopted its authentication infrastructure and trusted the CAs accredited by CACG.

As EDG drew to a close in 2004 it became clear that the practices, requirements and policies of the group – and most importantly the established trust relationships – should be carried forward [14].[3] The *European Policy Management Authority for Grid Authentication in e-Science* (EUGrid PMA) was founded, with the blessing of the European Commission's eInfrastructures Reflection Group [62], to coordinate authentication for EGEE [61], DEISA [54], LCG [105] and SEEGRID [151]. *The Americas Grid Policy Management Authority* [167] and the *Asia-Pacific Grid Policy Management Authority* [12] have since been established to coordinate grid authentication policies in those regions. In October 2005 the *International Grid Trust Federation* (IGTF) was established to coordinate policies and practices between the grid authentication PMAs [95]. The Global Grid Forum (now OGF) established a CA Operations working group to resolve international issues and establish policies and procedures for grid certification authorities [134].

2.2.5 Frameworks for PKI Organisation

It is useful at this point to discuss the organisation of large-scale PKIs such as the one established by the EDG CACG and coordinated today by the grid authentication PMAs and the IGTF.

Hierarchical PKI

$$\begin{tikzcd}
 & & Root\ CA \arrow[dl] \arrow[dr] & \\
 & Sub.\ CA_1 \arrow[dl] & & Sub.\ CA_2 \\
Sub.\ CA_{1.1} & & &
\end{tikzcd}$$

Figure 2.3: CA Hierarchy
The arrows represent a certificate signed by the source.

The conventional approach to building a large-scale X.509 PKI is to set up a hierarchy of certification authorities with a single root CA and a number of subordinate CAs (possibly at multiple levels) with various assurance levels, purposes, catchment areas, etc. It is not normally possible to include existing certification authorities into such a hierarchy without modifications.

When EDG was starting, there were some established CAs serving the relevant research and development communities in a number of European countries and it was considered important that these should continue to provide their services for the project. For this reason, amongst others related to the technical problems of supporting hierarchical PKI in the available software, a non-hierarchical approach was favoured. There are a number of methods for building a non-hierarchical network of trust between multiple CAs.

[3]The author of this thesis was a major contributor to the cited paper, coordinated the editing process, and presented it at the European Grid Conference in 2005.

Cross-Signed

In a *cross-signed network of trust*, each CA agrees to sign the certificates of some or all of the other CAs. An end-entity certificate issued by one CA in the network can be accepted by relying parties of a trusted CA since there is a path of trust from the end-entity certificate, through the cross-signed certificate, to the trusted CA. This requires cross-certification support in the software performing the verification. However, this is not supported by the Globus Grid Security Infrastructure (GSI) [75] and OpenSSL [50] libraries commonly used by grid middleware unless all the cross-signed certificates issued by the trusted CA are installed as well as the trusted CA's root certificate.

$$\begin{tikzcd}
 & CA_1 \arrow[dl, <->] \arrow[dr, <->] \arrow[dd, <->] & \\
CA_2 \arrow[rr, <->] \arrow[dr, <->] & & CA_4 \arrow[dl, <->] \\
 & CA_3 &
\end{tikzcd}$$

Figure 2.4: Cross-Signed CAs

Bridge

An alternative is a *bridge CA*, which signs the public keys of CAs that are to be considered equivalent. Each CA distributes a self-signed root certificate, and a certificate signing the public key of the bridge CA. That is, each CA cross-signs with the bridge CA. An end-entity certificate issued by a CA who is a member of the bridge can be accepted by a relying party who trusts the bridge CA, since the certificate chain provided by the end-entity can be verified all the way back to the issuing CA's root certificate. This is supported in the GSI and OpenSSL libraries only if all the cross-signed certificates issued by the bridge CA are installed as well as the bridge CA's root certificate and the trusted CA's root certificate. GSI has been tested and found to work in this configuration [91].

$$\begin{tikzcd}
 & Bridge\ CA \arrow[dl, <->] \arrow[d, <->] \arrow[dr, <->] & \\
CA_1 & CA_2 & CA_3
\end{tikzcd}$$

Figure 2.5: Bridge CA

Policy-Based Trust

Another alternative to a full cross-signed network of trust is a *policy-based network of trust* arrangement, which eliminates the cryptographic cross-signing aspect. In a policy-based network of trust each CA evaluates every other CA and, if the evaluation is positive, agrees to treat the other CAs as equivalent. Relying parties that want to accept certificates issued by all member CAs of the network of trust must install the root certificates for all the CAs in the trusted store. No special software support is required for this kind of network of trust.

In theory, with a bridge or cross-signed network the signature from the bridge CA or a trusted CA is enough to allow the verification of end-entity certificates from all CAs who are cross-signed with the bridge or with each other. In practice however, current software requires that all the cross-signed certificates be installed, as for sub-CAs, and so there is no great advantage in using the more complex bridge and cross-signing approaches. It could be argued that cross-certification poses a greater risk than the policy-based approach since sites in different trust domains (i.e. with a different local CA) install a different collection of cross-signed certificates and this limits the scope to verify the authenticity of these certificates by comparing them to copies from other sources. However, this risk must be balanced against the ability to revoke cross-signed CAs in a bridge or network of trust.

PMA as Bridge

The $N \times N$ evaluation involved in fully cross-signed or policy-based networks of trust has complexity of $O(N^2)$. However, by establishing a common set of requirements that each CA must meet in order to be accredited in a policy-based network of trust the complexity is reduced to $O(N)$. In effect, this puts the policy management authority in the role of a *policy-based bridge*. This allows the group to more easily scale up to tens of members and was one of the original reasons for the CACG adopting this structure and establishing minimum requirements for grid CAs.

2.2.6 Accreditation

International Grid Trust Federation

The IGTF authentication profile [94] sets the requirements for the policies and operation of 'traditional' X.509 Certification Authorities for grid authentication.[4]

1. **PKI Structure** – Within each country, large region or international organisation there should be a single certification authority with a wide network of registration authorities.

2. **Identity** – A subject's distinguished name must be linked to exactly one end-entity for the lifetime of the CA.

3. **Certification Authority** – The CA hardware must be physically secure and operated off-line unless it uses secure cryptography hardware. Each CA must have a CP/CPS. There are restrictions on the CA key, certificate and certificate revocation lists. Records must be kept and the CA must allow audits to take place. The CA must have a disaster recovery plan in place.

4. **Registration Authority** – The RA must validate the identity of a person based on photographic identification and/or official documents. The RA must communicate securely with the CA and must keep records.

[4]The IGTF also develops authentication profiles for other types of authentication systems, such as the Short-Lived Credential Services profile.

5. **End-entity Certificates** – Private keys must have a minimum length of 1024 bits, a maximum lifetime of 1 year and must be generated by the subscriber.

The EU GridPMA accreditation process [66] – in outline – is as follows. A new CA wishing to be accredited must distribute draft Certificate Policy and Certification Practice Statement (CP/CPS) documentation [42] to the PMA for comments. The PMA appoints a number of its members to review the CP/CPS in detail. The reviewers provide feedback to the CA on any internal inconsistencies or failure to meet the requirements set out in the authentication profile and the CA has an opportunity to make changes in line with the reviewers' recommendations. Once the documentation is ready, the CA will make a presentation to the PMA members describing in detail "the authentication and vetting procedure and the physical security measures, record persistency, procedures and such." To establish trust, it is important to convince relying parties that the CA was founded in good faith and the human element is very important for this. The PMA as a group has the opportunity to question the applicant about the policy and operation of the CA. If the CA is approved then relevant details, including the CA certificate, must be securely conveyed to the PMA chair for distribution. TAGPMA and APGrid PMA have similar accreditation processes which fulfil the requirements set down by the IGTF.

In general, fully-independent third-party audits are not required for grid CAs but there has been some recent work done in GGF by APGrid PMA members to devise an audit check-list that is based on the WebTrust programme (see below) and is consistent with the IGTF requirements [168].

The IGTF requirements and accreditation procedures for grid CAs have counterparts in the wider IT context. These are discussed below.

WebTrust Program for Certification Authorities

The WebTrust Program for Certification Authorities [3] is one of the more widely recognised accreditation standards for CAs. This programme was developed jointly by the American Institute of Certified Public Accountants and the Canadian Institute of Chartered Accountants in consultation with PKI providers and users. It is cited by both Microsoft and Mozilla in their CA requirements (see below).

In order to acquire the WebTrust seal of approval a CA must be audited by an approved WebTrust auditor and must meet all the requirements set out in the programme. At a high level, WebTrust sets out three principles for CAs.

1. **CA business practices disclosure** – The certification authority discloses its key and certificate life cycle management business and information privacy practices and provides its services in accordance with its disclosed practices.

2. **Service integrity** – The certification authority maintains effective controls to provide reasonable assurance that:
   - Subscriber information was properly authenticated (for the registration activities performed by the CA).
   - The integrity of keys and certificates it manages is established and protected throughout their life cycles.
3. **CA environmental controls** – The certification authority maintains effective controls to provide reasonable assurance that:
   - Subscriber and relying party information is restricted to authorised individuals and protected from uses not specified in the CA's business practices disclosure;
   - The continuity of key and certificate life cycle management operations is maintained; and
   - CA systems development, maintenance, and operation are properly authorised and performed to maintain CA systems integrity.

(list taken, in edited form, from [3])

The programme expands these principles into detailed criteria and provides illustrative example report values.

Application and Operating System Trust Stores

CA root certificates typically must meet certain standards for inclusion in the trust stores of popular operating systems, web browsers and other applications, such as Microsoft's Windows family and the Mozilla Foundation's suite of products, and so inclusion can be considered a form of accreditation.

The Microsoft Root Certificate Program [119] requires CAs to complete a WebTrust Program for Certification Authorities audit or an equivalent third-party audit. Microsoft also require that the CA root certificate must have a sufficiently long lifetime (it must not expire before 1st January 2010 at the time of writing) and the CA "must provide broad business value to Microsoft platform customers", which excludes internal organisation-specific CAs and possibly other non-commercial CA providers.

The Mozilla CA Certificate Policy [89] requires CAs to meet ANSI [8], ETSI [64, 65], or WebTrust criteria for CA operations. It also imposes some additional requirements excluding CA certificates that may cause technical difficulties, and setting minimum standards for request verification, for example.

The CA root certificates included with application software and operating systems are *de facto* trusted by most users to the extent that only a small fraction will independently assess the trustworthiness of the included CAs themselves and disable those that do not meet an acceptable standard.

2.3 Grid Computing & Security

There is a spectrum of wide-area distributed computing systems from light-weight peer-to-peer and volunteer systems to more heavy-weight grid and metacomputing systems. The term *wide-area distributed computing system* is used here to refer to general-purpose systems which can run multiple different distributed computing applications without major reconfiguration. A system's *weight* refers, at least in part, to the burden of software installation and configuration required on the system's nodes.

In this section some common grid security concepts are covered, followed by a description of a number of pieces of grid and distributed-computing software with a particular emphasis on their security components. Some specific security systems are dealt with at the end of this section.

2.3.1 Grid Security Concepts

Authentication and authorisation are two important aspects of grid security systems. Authentication corresponds to proving 'who you are' and authorisation to 'what you are allowed to do.' Sometimes the boundary between authentication and authorisation is clear, while sometimes the two are somewhat entangled. Other aspects of any useful security system include auditing and accounting ('what you have done,' perhaps). In this thesis the focus is on authentication and authorisation.

Authenticated Communications

Two common approaches to authenticated communication rely on authenticated channels and on individually authenticated messages.

SSL/TLS Channels

Secure Sockets Layer (SSL) or Transport Layer Security (TLS) channels [56] typically make use of both symmetric-key and public-key cryptography. When establishing a channel between peers authentication is performed using public keys. This may be one-way authentication in the case where only the client authenticates the server, or it may be two-way mutual authentication in which both parties authenticate each other. A peer signs some message to prove possession of the private key associated with a particular public key tied to its identity. After successful authentication a symmetric session key is shared between the peers to take advantage of the lower computational overhead of symmetric-key ciphers. This is the most common form of authenticated communication in current grid middleware.

Signed Messages

Individual messages can be secured with encryption and sent over some, possibly insecure, channel. If integrity of the messages is important, a signature can be made by encrypting a cryptographic hash of the message with the sender's private key. If confidentiality of the message is also considered important then the entire message can be encrypted with the receiver's public key. Alternatively, a symmetric encryption key can be used to encrypt the message body and then this message key can be encrypted with the receiver's public key and sent along with the message. In cases where an encrypted messages is to be sent to multiple recipients it is more space-efficient to encrypt the message once with a symmetric key and then encrypt that key with the public key of each recipient.

Authorisation

Authorisation is concerned with granting an entity rights or permissions to a resource. The grid security systems presented later in this chapter are largely based on the concepts presented here.

In discretionary access control (DAC) systems each resource (a file, for example) has an owner who is responsible for permitting or denying access to the resource. An example is the typical Unix file permissions. Each file has an owner and the owner can decide if it should be readable by no-one else, by a chosen group, or by everyone. Mandatory access control (MAC) systems impose access restrictions on their users and the permissions they can grant. Resources and users have security levels and a user's clearance level must be compatible with that of the resource to gain access. A system policy can enforce that a user is not permitted to lower the security level of a resource.

These forms of access control centre on the user. Role-based access control (RBAC) grants permissions to particular *roles* and any user who assumes a particular role will have those permissions. Attribute-based access control (ABAC) extends this to any attribute a user might have. While roles and attributes for an operating system can be assigned to users by a local administrator, for distributed systems a network service may be provided that can issue credentials representing a user's roles, attributes, or permissions.

In DAC-type systems, the policy governing authorisation is usually closely tied to the resource: for example, the permissions stored in the Unix file-system. However, it is often desirable to manage policies independently of the resources, particularly when using an RBAC or ABAC model of access control. In such a system, the authorisation *decision* to permit or deny access may be made separately from the *enforcement* of the authorisation.

For example, a user wishing to access a resource contacts a service. The service contains a *policy enforcement point* (PEP), which contacts a *policy decision point* (PDP). The PDP evaluates the user's identity, attributes, etc. against an authorisation policy and returns a decision to the

PEP. The PEP enforces the decision and permits or denies access [175].

Delegation

It is a common occurrence in the real world for a person or organisation to delegate authority to an agent: another person or organisation who will act on their behalf. In the context of distributed computing systems such as grids, users will often need to delegate their authority to a service which will act on the user's behalf to carry out some work. In turn, this service may need to delegate authority further to complete the task.

Delegation of authority can, at its most coarse-grained, give the delegate – i.e., the agent – full control and the ability to impersonate or act on behalf of the delegant. More fine-grained forms of delegation limit the authority of the delegate in time and scope.

In the case of impersonation, the actions of the delegate are indistinguishable from those of the delegant. In slightly controlled but still coarse-grained delegation the delegate may act with the identity of the delegant but with a clear indication that it is acting as an agent. Delegation may be made more fine-grained by specifying the actions that may be performed or the roles that may be assumed by the delegate under the name of the delegant. In particular, the delegant may limit the right to further delegate authority.

In the above description of delegation it has been assumed that the delegant's identity is an important part of what is delegated but this need not be so. Instead, authority for individual actions or roles may be delegated in such a way that the delegated permissions and the authority of the delegant can be verified but without a direct dependence on the delegant's identity. With this approach the delegate can operate under its own identity or even anonymously as long is it can show that it has been delegated appropriate authority.

Identity Mapping

In general authorisation can be performed by checking a given identity against a list of allowed identities. In some cases it may be necessary to map an identity from one form to another. An approach used for grid authorisation is to map from the identity presented in a client certificate to a local system account. This is appropriate where resources are protected by the local operating system. The identity can be mapped to an individual account, which the administrator of the system could use to uniquely identify the user, or it could map into a pool of accounts shared among the users belonging to a virtual organisation or some other grouping. Alternatively, accounts can be generated automatically for authorised users.

Virtual Organisations

An important concept in grids is the *virtual organisation.* A virtual organisation (VO) consists of a collection of people and institutions who agree on resources sharing rules, together with associated computing, storage and network resources [78]. A VO could be formed for a long-term project with many participants or for a brief collaboration between individuals.[5] Methods and processes for automatically negotiating agreements between resource providers and potential collaborators and dynamically establishing VOs are being researched. A single grid typically supports multiple virtual organisations, and it is quite possible that an individual might be a member of several VOs within that grid.

2.3.2 Grid Computing

There is currently a huge variety of grid middleware with varying approaches to management of computational work and data and to security. In this section a sample of this middleware is described.

Globus Toolkit

The Globus Alliance produces widely-used grid middleware known as the Globus Toolkit [77, 74]. The Globus software closely reflects the ideas of Foster, Kesselman, Tuecke *et al.* due to their involvement in the project. The Globus Alliance also has a strong connection with the Open Grid Forum. Globus evolved from the I-WAY project [53].

The Globus Toolkit, now at version 4, consists of a collection of services which can be deployed to provide a working grid. These consist of information services; data management services; and computing resource services. The information services keep track of grid metadata for the purpose of monitoring and allowing discovery of data and computation resources. Versions 1 and 2 of the Monitoring and Discovery Service (MDS) are based on LDAP [72]. Resource information from individual Information Providers is collected by Grid Resource Information Services (GRIS) and these are aggregated by a Grid Index Information Service (GIIS). Version 3 of MDS uses OGSI mechanisms, while version 4 uses mechanisms based on Web Services.

The data components include GridFTP, an enhanced file transfer system supporting third-party transfers, striping, and additional security features [5]. The Open Grid Services Architecture Data Access and Integration project (OGSA-DAI), developed in collaboration with Globus, provides access to various database resources on the grid [9].

[5]While the VO concept is intended to apply to any size of collaboration, currently most VOs on the major European and international grids are associated with relatively large projects. The administrative and technical overhead of establishing a VO is currently quite considerable and so smaller collaborations are excluded to some extent. Attempts are being made to reduce at least the technical overheads of administering a VO. One simple solution is to allow the new VO to act as a 'sub-VO' of an established VO, although this is only appropriate for collaborations having some association with the higher-level VO.

The main computing resource component is the Globus Resource Allocation Manager (GRAM) [51]. The GRAM Gatekeeper receives requests written in a Resource Specification Language. If the request is authorised and can be fulfilled it is passed to a job manager, which passes the work to a batch queueing system.

Early versions of the Globus Toolkit, up to and including version 2, used communications protocols which were well-documented but otherwise non-standard. The Globus developers should not be faulted for this: at the time practically every class of networked software had its own protocol. From version 3 onwards there was a clear shift to developing services as *web services* which operate over HTTP or HTTPS on the standard ports of 80 and 443. Web services bring certain benefits such as implementation independent and machine readable (and semi-human-readable) protocol specifications in the form of WSDL and circumvention of port-based firewall restrictions (obviously not a benefit from everyone's point of view).[6] Version 3 required a modified web services engine to host the Globus *grid services*. Version 4 aimed to produce standards-compliant web services that could be used in any compliant web services container.

While Globus Toolkit 4 is the current major version, the unsupported version 2.4 is still widely used (e.g. in LCG, described below).

Security

Globus uses X.509 certificates for authentication and can support a variety of authorisation systems. The simplest authorisation system uses a *grid mapfile* to map certificate subject names to local account names. Another system is the Community Authorization Service described below.

A string representation of the certificate subject name is used to perform this mapping. The string representation can cause problems with names that use unusual distinguished name components and the support for non-ASCII characters is rather poor. The certificate's issuer name is not included in the mapping; it is assumed that different CAs will not issue certs with the same subject distinguished name to different entities and this uniqueness requirement is enforced by the agreements on CA namespaces.

In the context of Globus each virtual organisation is expected to set up its own grid infrastructure. The Globus software is not explicitly aware of VOs since it is assumed that all communicating users and services are within the same VO.

Single Sign-on & Delegation

A goal of the Globus Grid Security Infrastructure (GSI) was to provide a 'single sign-on' capability for users [75]. This means that a user must sign on to the grid only once per session no matter which resources are used. Another requirement was to allow delegation of authority from a user to

[6]In essence, the level of abstraction has simply been changed and so instead of implementing binary protocols over TCP/IP it is now possible to implement (possibly non-standard) text-based protocols over HTTP. If Web Services grow in popularity then packet-inspecting firewalls will be more widely deployed in order to control traffic at the HTTP level.

a grid service acting on the user's behalf. This takes the form of delegation of the user's identity.

In Globus GSI, single sign-on and delegation are accomplished by the use of *proxy credentials*. To implement single sign-on, a new key-pair is generated on the user's system and a certificate is created with identity details based on the user's own certificate. The proxy certificate is signed with the user's private key. The proxy credential's private key is stored unencrypted and so the user does not need to re-enter the pass-phrase of their long-term private key.

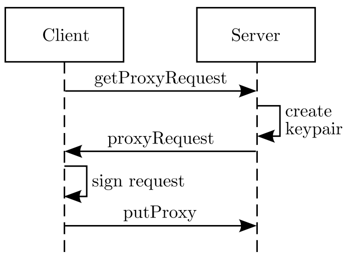


Figure 2.6: Proxy Delegation

To delegate a credential to a remote service the user and the service mutually authenticate and the user indicates a wish to delegate. The service generates a new key-pair and returns a certificate signing request for the user to sign. The user signs the request with their private key to create a proxy certificate and returns it to the service. The private key of the delegated proxy credential is not transferred across the network.

Proxy credentials are a form of impersonation and so effectively have all the rights of the originating user. In addition, the delegated private keys are typically stored unencrypted and protected solely by the operating system's own measures. Although this form of delegation allows a great amount of flexibility it also reduces the level of security by increasing the risk of theft of the delegated credentials.

To limit the exposure, a proxy certificate has a short lifetime (typically 12 or 24 hours). Revocation of proxy certificates is not directly supported. The short lifetime can, however, cause problems for long-running grid jobs and so in practice mechanisms for renewing proxies are required. The proxy certificate may contain a flag to indicate that further delegation is not permitted.

In the validation of this proxy certificate, the `basicConstraints` attribute of the user's certificate (which states that it is not a CA certificate) is deliberately ignored, going against normal validation procedures. GSI proxy certificates have been standardised in the Internet Engineering Task Force (IETF) [174] and are now supported in widely used software such as OpenSSL. Proxy certificates are widely used by other grid software which is based on, or interoperates with, Globus.

Community Authorization Service

The Community Authorization Service (CAS) is responsible for managing the policies that govern access to the resources of a virtual organisation or community [138]. The service addresses the requirements for scalability in administering VOs; flexibility in the expression of requirements in policy; and allows for hierarchical policy management. The service keeps a record of membership of a community as well as the relevant policies. A user wishing to access a resource contacts the CAS server and requests the appropriate rights. The server checks the policies and grants appropriate *capabilities* to the user, which can then be presented to the resource. The resource examines the capability and grants access if the request is authorised.

Capabilities are issued in the form of certificate extensions contained in proxy certificates. The capability policy and an indication of the policy language used are included. Such a credential is known as a *restricted proxy.* The CAS service does not dictate any particular policy language: this is left to the community to decide.

Legion

Legion is an object-oriented distributed computing system. The Legion design aims to provide a secure, easy-to-use, high performance system for parallel processing which would take advantage of resource heterogeneity [86, 85]. From a user's perspective, the aim is to provide a single virtual computer with a consistent object name space and transparent access to distributed resources.

Security

The Legion project made some clear security assumptions: that the system will be running across a very wide area; that it will be running on top of different host systems, which may get compromised or even be malicious; and that there will not be a single owner of all the systems involved and so there may be varying policies and levels of security [177].

From these a number of high-level principles are developed: do no harm; let the user beware; and small is beautiful. The first means that Legion should not significantly increase the risk to a resource. The second means that users are responsible for their own security policy. The third means that a simple security model is to be preferred.

In the Legion model, every object has a `MayI` method which determines if a particular call is permitted. There is a distinct set of rights attached to each of the object's methods.

The 'I' in question can be the Calling Agent, who initiated the call in question; the Responsible Agent, often the ultimate 'user' who initiated the chain of events leading up to the call; or the Security Agent, a third-party policy enforcement agent. The `MayI` method can make a decision based on any subset of the Agents. The identity of a calling agent can be checked with an `Iam` method. Each object also has a `Delegate` method which is used to delegate particular rights to another object. Restrictions can be applied to the delegation.

Every Legion object and user has a Legion Object Identifier (LOID) which incorporates an X.509 certificate [69]. This allows other objects or users to verify messages from, or send encrypted messages to, an object or user with a known LOID. The default implementation of `MayI` checks the caller's credentials against a list of LOIDs to accept or deny.

UNICORE

UNICORE (**Uni**form **I**nterface to **Co**mputing **Re**sources) was created to improve access to high-performance computing (HPC) and data resources [6]. The stated motivations for the project were that existing HPC systems were often incompatible in terms of hardware and software; that file systems were fragmented across multiple systems; and that security systems in particular were inconsistent. The aims of the project were deliberately kept simple. Only direct, batch-mode access to existing HPC systems would be provided. Interactive access, presenting the appearance of a single virtual computer, and resource brokerage were not dealt with. UNICORE is currently used by the DEISA supercomputing grid infrastructure [54]. The UNICORE project took advantage of the languages, protocols and standards used on the Web at the time of its initial development such as Java, HTML, and HTTPS. In the UNICORE model, a piece of work is described in an Active Job Object (AJO).

Security

X.509 CAs provide certificates for UNICORE users and services. These are used to authenticate users and to digitally sign the UNICORE applets. AJOs are transferred using the UNICORE protocol layer (UPL), based on SSL. When an AJO is transferred to a server it must be accompanied by two certificates; one from an *endorser* and the other from a *consignor*. The endorser corresponds to the originating user, and the AJO is signed by the endorser. The consignor is the entity transferring the job. When a user submits an AJO initially they will act as both endorser and consignor. A server may consign the AJO to another server but the server can only consign jobs that have already been endorsed. The destination server will verify the endorser's signature and make an authorisation decision based on the identity and the local policy. Explicit Trust Delegation has been proposed to make delegation more flexible [158]. A *user* role is introduced which identifies the originating user, but the endorser role can now be delegated to an agent or server. This allows the agent to create actions and AJOs on behalf of a user in a controlled manner.

European DataGrid

The European DataGrid Project [82] was set up to develop middleware for a *data grid* to support scientific research. Data management, including handling of data replicated between sites, was one of the main goals of EDG. The Replica Location Service (RLS), developed in collaboration with Globus, provides mappings from logical names to replicated data.

There was also development in the area of computation management. EDG developed a resource broker (that is, a service to act as an intermediary between providers of computing and data resources, and their consumers) with support for retrieving data and returning results to replicated data storage. A relational grid-monitoring and information system, R-GMA [71, 47], was also developed within EDG.

Security

At the start of the project security requirements were collected and documented [49]. These included seventeen requirements for authentication. Three important instances are: that a user should need to authenticate just once per session; that authentication should be interoperable between many grids and applications; and that it should be possible to revoke authentication credentials in the event of loss or compromise. It was considered important to keep authentication and authorisation separate. The primary reasons for this were to allow interoperability of authentication tokens with developing authorisation systems and to allow single sign-on across multiple VOs. The authentication and delegation system is based on the Globus Grid Security Infrastructure, described above. As described in Section 2.2.4, a coordinated group of peer CAs appeared to be the most suitable arrangement for a PKI. The European DataGrid security architecture is described in [48].

Virtual Organisation Membership Service

In a Grid environment it is natural to grant access to resources according to a user's membership of an appropriate virtual organisation supported by the resource owner. With the standard Globus GSI approach, access depends on each user having a corresponding entry in a grid mapfile on the target resource. This requires the site administrators to set up individual user accounts for all authorised users. Synchronisation of these mappings across all the sites involved in a VO is a serious administrative overhead, similar to synchronising Unix `passwd` files. This approach also limits users to belonging to one VO at a time.

An improvement within EDG was to create a virtual organisation membership LDAP server which allowed VO information to be stored centrally [169]. In this way an administrator for a virtual organisation could add users as members, and manage groups within a VO. To remove the need to create individual accounts for all VO members at each site, a system was devised to lease accounts from a pool [118]. A Grid mapfile can be generated automatically from the information stored on the VO LDAP server and updated at regular intervals.

There were still a number of problems with this system. When the VO information is only updated daily, a new user has to wait up to 24 hours for all sites to update. The same problem occurs if a VO manager wishes to make changes to group membership.

In response to these issues, and also to provide more fine-grained authorisation, EDG, in conjunction with DataTAG [52], developed a new VO service, the Virtual Organisation Membership Service (VOMS) [4].

With VOMS the user contacts the server to acquire a credential that contains virtual organisation information attributes. This means that a user can acquire a VO credential as soon as she has been added to the virtual organisation. When the VO administrator modifies her group membership, allowed roles or capabilities, she can acquire an updated credential as soon as the change is made. A user can contact multiple VOMS servers to accumulate credentials for several VOs.

VOMS provides a relatively convenient administrative interface. An authenticated and authorised virtual organisation manager can add users, define group membership, and assign roles and capabilities. Any authenticated user can request virtual organisation or group membership, roles and capabilities and the administrator can grant these as appropriate.

To perform a Grid operation, a user requests a short-lived credential from the VOMS server for the virtual organisation in question, specifying which groups, roles and capabilities he wants to use. If the user is authorised, the service generates and signs a short-term attribute certificate [68] containing the VO information as an optional extension to the GSI proxy certificate. The VOMS-extended credential may be used in a backwards-compatible fashion with standard GSI services.

VOMS can be used in a backwards compatible fashion to generate a grid mapfile. This still suffers from the problem of having only a single entry per user. However, services which are VOMS-aware can make use of the VOMS credentials directly. LCAS and LCMAPS can be used to add such support to services.

LCAS & LCMAPS

The EDG Local Centre Authorisation Service (LCAS) [159] is an authorisation decision engine that can add advanced access control to underlying Grid services such as the Globus gatekeeper or GridFTP server. LCAS provides a plugin framework to allow flexible authorisation set-up. The authorisation plugins work together to reach a collective decision based on the resources requested, the identity of the requester from the delegated proxy certificate, and any further credentials the requester may have, such as VOMS virtual organisations, groups, roles and capabilities. Plugins exist which support

- banned and allowed user lists;
- access control lists based on VOMS attributes; and
- ‘opening hours’ access based on the time and date when a request is received.

The EDG Local Credential Mapping Services (LCMAPS) [160] provides a flexible system to map Grid users to local credentials on Grid sites. LCMAPS supports plugin modules to support a variety of mappings:

- static mapping from a Distinguished Name onto a local Unix account and group;
- mapping to leased account from a pool of accounts;
- mapping VOMS groups, roles and capabilities onto Unix groups, possibly from a pool of groups, analogous to the pool of accounts; and
- mapping from a Distinguished Name onto local AFS tokens.

The credential mappings that are in effect for the execution of a Grid job are recorded in a job repository, which allows some level of auditing.

Java Trust Manager

To support Java services within EDG (and later in EGEE's gLite) the Java Trust Manager and Java Authorisation Manager were developed [60]. The Trust Manager provides Java support for GSI proxy credentials, and the Authorisation Manager handles authorisation for services running in a Java servlet environment such as Tomcat. These libraries are used in the EDG Spitfire database server [19], the VOMS Admin service [4], and in the R-GMA grid information system.

CrossGrid

CrossGrid [84] was a sister project of EDG with an emphasis on interactive jobs: keeping the user in the loop. For example, two CrossGrid applications were emergency flood modelling and blood flow modelling for bypass surgery. CrossGrid enhanced the EDG resource broker with better match-making and support for advanced reservation, important features for allowing a user to interact with a submitted job in near-real-time. A number of tracing, debugging and analysis tools were developed to aid grid application writers and users.

The EDG security policies and mechanisms were carried over to CrossGrid due to the shared software and objectives of the projects. As mentioned earlier, CrossGrid accepted the EDG CACG CAs and those CrossGrid CAs that were not already members joined the group.

Large Hadron Collider Computing Grid

The Large Hadron Collider (LHC) at CERN will be used for a number of large-scale particle physics experiments due to begin in 2007. The LHC Computing Grid (LCG) will process the output of these experiments [105]. The first version of the software for LCG was provided by EDG. Some of the more experimental (or unfinished) pieces of EDG were dropped for the sake of overall reliability. The LCG security architecture is based on that of the earlier European DataGrid software outlined above [48].

Enabling Grids for E-science

From its inception, EGEE was seen as the follow-on project from EDG and a parallel project to LCG.[7] Its primary goal was to run a production grid service to support a number of scientific computing domains. A lightweight, portable grid middleware, gLite, was to be developed, at least partly to reduce dependence on the Globus software.

In 2006 the software releases from LCG and EGEE were merged. gLite 3.0 consisted of the latest version of the LCG software with a number of gLite components added. At the time of writing the LCG/gLite software still depends on external grid and cluster components such as the Globus Toolkit, Condor-G, and the Torque batch scheduler.

[7]EGEE originally stood for 'Enabling Grids for E-science in Europe' but was later changed to reflect the project's global nature. The extra 'E' was retained and is attributed to the final letter of 'E-science'.

Security

The EGEE security architecture reflects the evolution of the earlier EDG architecture to take note of developments and in particular the move to web services and service-oriented architectures [124]. The architectures aims to be modular, implementation agnostic, and standards compliant.

While web-services protocols can make use of message-level XML-based security, transport level (i.e., TLS) is favoured because of the superior performance offered at the current state of development.

One of the new components that has been developed is a delegation web service (and a standard interface for such delegation web services). Existing software originating in EDG and related projects has been maintained and enhanced by EGEE. This includes VOMS, LCAS, LCMAPS, and the Java Trust Manager.

There are plans to develop privacy and pseudonymity services and encrypted storage systems, and to move to OCSP for revocation information. Credential stores, which take private keys out of the hands of users, are favoured for their potential to improve private key hygiene.

2.3.3 Volunteer & Peer-to-Peer Computing

Volunteer computing systems allow computer users to donate the idle time on their computers to be used for distributed computing tasks. This can be for a specific application, as in the case of SETI@home [102] and the Distributed.net RC5 challenge [57]. With other systems, such as BOINC, based on the SETI@home software, a user can download one piece of software that manages the client's participation in the distributed computation, and then choose which individual applications to run using it [7]. For example, a BOINC user might contribute CPU time to LHC@home [37], Folding@home [154], or the ClimatePrediction.Net project [45].

With the systems mentioned above the user chooses what applications will be run on their system but otherwise has little control over what happens beyond simple throttles on CPU and network usage. This is understandable as the originator of the computation needs to be sure that the results coming from untrusted systems can be trusted.

There are quite a number of Java-based web volunteer computing systems such as Bayanihan [149], Charlotte [17], and Javelin [43]. Such systems have the advantage that experiment software will be portable to any system running the same volunteer computing software. However, the disadvantage is that the software must be written in Java and this limits the usefulness for running or even linking to existing software written in other languages.

Condor [107, 170] started life as a volunteer computing system, which would allow work to be distributed to idle workstations. Its support for check-pointing allows work to be migrated between hosts when a workstation user begins interacting. Over time Condor has been extended to support dedicated clusters and to allow desktop resources to be integrated with dedicated resources transparently. Condor-G can be used to schedule work to be sent to Globus resources and can manage directed acyclic graph (DAG) workflows with its DAGMan component [81].

Volunteer systems are sometimes used at the organisational level, with forced 'volunteers', rather than at desktop level. In this approach large numbers of computer systems in, for example, a university computer laboratory or a commercial office are configured to act as computing nodes. The system configuration is controlled by system administrators and the applications run on the system are determined by those who have been given permission to use the resources.

While the volunteer computing systems mentioned above generally have a manager node sending work to worker nodes, it is also possible to arrange the computation in a less hierarchical fashion. Peer-to-peer computing systems transfer units of work and results between connected peers. The peers may use central discovery services or may be purely distributed.

WebCom

WebCom [120] is an implementation of the *condensed graph* model of computation [121]. A condensed graph represents the structure of a computation in terms of dependencies between operands, operators and destinations.

Each condensed graph (CG) has an *enter node* and an *exit node* which provide the inputs and collect the output of the graph, respectively. Each node in the graph has *ports* for, or dependencies on, one or more operands, an operator, and one or more destinations. Connections between node may initially be *stemmed*, meaning that the source of the connection will not be evaluated until it is *grafted* to a destination. In Figure 2.7 the connection between the × node and the if node is stemmed. The left-hand-side of the graph, which leads to the × node, will be grafted and hence evaluated if the result of the = operation is false, otherwise the right-hand-side of the graph (with value 1) will be grafted. The configuration of the graph can allow eager, lazy or imperative scheduling of operations.

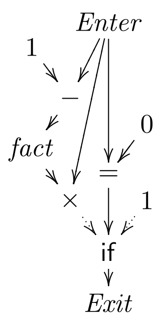


Figure 2.7: Condensed Graph
The recursive CG *fact* computes $n!$.

WebCom distributes evaluation of condensed graphs over a peer-to-peer network of computers. There is support for fault tolerance, load balancing, and scheduling of nodes. Nodes can be targeted at particular hosts or matched with resources that meet certain requirements by using a ClassAds matching system, originally developed for Condor [143].

WebCom provides a graphical user interface which can be used to create graphs from a palette of existing operations, or from code snippets provided by the user [122].

Security

WebCom allows for secure communication between instances with authentication based on X.509 certificates and authorisation using KeyNote credentials [73]. The WebCom environment interacts with KeyNote – described below – through its API, meaning that each WebCom graph node does not need to make any security decisions internally. Synchronisation, concurrency and trust management are all handled transparently by the WebCom environment.

WebCom instances are connected in a peer-to-peer topology. Each peer connection entails mutual authentication, by verifying the X.509 public key certificates using the appropriate trusted CA root certificates. Each WebCom instance can perform authorisation when communicating with another instance. In WebCom, authorisation credentials can be granted to permit: submission of a specified graph to another instance; execution of a specified graph received from another instance; execution of a specified operand, return of results to another instance; etc. X.509 certificates and KeyNote authorisation credentials must be created and distributed to WebCom instances. WebCom does not provide user authentication, user authorisation or credential delegation.

WebCom-G

The WebCom-G project [123] aims to combine WebCom with established grid middleware from the LCG project and to make WebCom more useful as a piece of grid middleware in its own right.[8] WebCom, and the condensed graph model it implements, is very suited to creating workflows of grid jobs that themselves can run existing software without the expense of porting to the WebCom condensed graph format. In particular WebCom-G aims to allow submission to multiple grids from a single workflow.

WebCom-G has encouraged research and development of a more general approach to interoperability between existing grid and workflow middleware. This approach is given the name Metagrid [140]. As part of WebCom-G, WebCom has been enhanced to address some of the security issues described above. In addition, work has been done in the course of the research for this thesis to bridge the gap in security between WebCom and existing grid middleware. Metagrid and WebCom-G security are discussed in more detail in Chapter 6.

2.3.4 Other Authentication & Authorisation Systems

Shibboleth

Shibboleth [153] is an architecture that supports the creation of federations within or between institutions for web-based single sign-on. A Shibboleth *Identity Provider* at a user's institution

[8]The name *WebCom-G* is used here to refer to the collaborative research and development project involving researchers in University College Cork, Trinity College Dublin, and National University of Ireland, Galway, and to the overall products of this research. The name *WebCom* is used to refer to the condensed-graph–based distributed computing software developed in UCC. Outside of this thesis *WebCom-G* may sometimes be used more specifically to refer to the versions of the WebCom software developed in UCC during the WebCom-G project.

can provide SAML attribute assertions about the user to a *Service Provider*, which can make use of the attributes for authentication and authorisation. The assertions can be anonymous, so the Service Provider need not be aware of the user's true identity.

There have been a number of efforts to integrate Shibboleth with existing grid security systems. GridShib allows Globus services to request user attributes from Shibboleth Identity Providers to make authorisation decisions [18]. Shibboleth is also used to federate institutional identity systems to issue X.509 certificates that can be used directly with grids [165].

Kerberized Certification Authority

The Kerberized Certification Authority (KCA) provides a automated mechanism for an organisation with an existing Kerberos infrastructure to generate X.509 credentials for use in PKI-based authentication systems [101]. The KCA software is distributed by the NSF Middleware Initiative.

The KCA consists of a secure server which communicates with a client to generate PKI credentials. The KCA service is attractive for sites operating a Kerberos-based authentication infrastructure. The user is not issued a long term private key, and proxy renewal uses the existing Kerberos infrastructure. The administrative overhead and possibility of error or deliberate attack on a separate RA is removed. Since the KCA issues only short-lived certificates, there is no need to distribute CRLs. Compared to a well run off-line service the danger of signing key compromise is increased. In the context of long-running jobs in the grid the problem arises of how to renew a proxy certificate derived from a user's Kerberos token which is typically valid for about one day.

Virtual Smart Cards

The SLAC Virtual Smart Card system [88], provides an on-line credential store analogous to a physical smart card. As it is often unwise to trust users to keep private keys secure it is argued that they should not be given the private key. VSC can provide stronger security guarantees with a central restricted-access server than individual untrustworthy users, and it allows users to generate proxy certificates from anywhere that has access to the VSC server. The disadvantages are that the private keys are concentrated in one place, therefore giving a single point of failure, and the authentication for the whole system is only as strong as the authentication with the VSC server, so this must be of high quality, e.g. a well-administered Kerberos set-up.

A-Select

The A-Select Authentication System [1] is a framework for user authentication with web applications. It supports multiple Authentication Service Providers including Radius and LDAP as well as others, for example, based on authentication with Internet banking services.

KeyNote

Trust management [23, 21] is an approach to security which unifies the specification of security policies, trust relationships and credentials. A credential directly represents the subject's authorisation as delegated by some authority, and it will be respected if that authority is recognized by local policy. This is in contrast to a traditional identity-based access-control approach where access is granted based on who is making a request, and the local access control policy for that request. In such cases it is often necessary to know in advance about all possible users. In a trust management system cryptographic keys are used to identify authorisers and licensees. An authoriser creates a credential containing the licensee's public key and the appropriate authorisation attributes, and signs it with his private key.

KeyNote [22, 24] is an implementation of a trust management system. It provides a compliance checker which is used to verify credentials against the local security policy, and a simple application programming interface. The KeyNote compliance checker provides a standard mechanism for verifying credentials against policy, taking this task away from the application programmer. Credentials and policies (collectively assertions) have a simple, expressive format. The only difference between a credential and a policy assertion is that a local policy is unconditionally trusted. Assertions are created and managed independent to the application, separating the security policy from the application functionality.

PERMIS

PERMIS (for **P**rivil**e**ge and **R**ole **M**anagement **I**nfrastructure **S**tandards Validation) provides policy-based authorisation services based on X.509 attribute certificates [40]. PERMIS supports role-based access control of resources and, like KeyNote, is a trust management system [22]. Authorisation policies are specified in an XML format (support for XACML is under development).

A *privilege allocator* defines policies and issues role assignment attribute certificates. The *privilege verification* system can be used to verify PERMIS credentials presented by a user. As part of the TrustCoM project [173] support for including *reputation* information into both policies and decisions is being added to PERMIS.

MyProxy

MyProxy is a credential store for grid proxy certificates [131]. The service allows users to delegate a proxy credential to it and assign a username and password. A user can later request that a proxy is delegated back from the MyProxy service by presenting their username and password. Proxies stored on a MyProxy server typically have a longer lifetime than those normally used (e.g. seven days rather than one day). For this reason MyProxy can be used to support proxy renewal for long-running jobs. In this case, the MyProxy server is configured to accept requests for delegation without a username and password provided that the request comes from an authorised host and the requester is in possession of a valid proxy to be renewed.

MyProxy has been enhanced to support the use of PAM to authenticate users logging in. It is also possible to use MyProxy as an on-line CA which issues certificates based on configured user logins [18].

CredEx

A "Credential Mapping Service" is one of the proposed services in the OGSA architecture to allow translation between security realms, which may use different authentication credentials for good reasons [156].

CredEx [55] is an open-source standards-based web service for credential exchange. Tokens are exchanged using a service based on the WS-Security and WS-Trust standards. CredEx is implemented with the Axis web services libraries. The initial version allows setting, getting and removing credentials and supports exchange of username–password pairs and X.509 proxy credentials. In its current version, messages to and from the services are not encrypted so it is advisable to use an encrypted HTTPS channel for the service.

2.4 PKI Services

IN this section are described some of the services that are most closely related to the central topic of this thesis: on-demand trust evaluation. PKI validation services perform certificate validation tasks on behalf of clients. Potentially, this can provide a more reliable, centrally managed (although possibly distributed) service than local certificate validation. Trust evaluation systems attempt to evaluate the trustworthiness of CAs or other entities.

2.4.1 Validation Services

Fraunhofer New Security Infrastructure

The New Security Infrastructure (NSI) project aimed to increase interoperability between PKIs and reduce the complexity of PKI use in applications and services [79]. The main focus of the project was on a *PKI server* (PKIS) which centralises the tasks of certificate validation, signature verification, certificate retrieval, and certificate path building. A PKI Server can play a variety of roles depending on client requirements. Some clients may only use the PKI server for path building and perform certificate validation locally. Others may use the PKI server for all certificate validation and signature checking, and in this case must completely trust the server. Clients may choose how much work to offload on the server based on their varying computing capabilities.

A PKI server can support multiple validation policies which specify “trust anchors, revocation checks and constraints of the constructed certificate path.” An administrator can centrally manage these policies, and clients can choose which policy to request depending on the validation context.

A novel feature of the NSI approach is that PKI servers can exchange information with other PKI servers about certificate paths, although trust anchors are not shared. The PKI server-to-server protocol propagates certification path information in a manner similar to that used by IP routers to exchange routing tables.

DNV Validation Authority

Det Norske Veritas Research is developing a *Validation Authority* (VA) [135]. The Validation Authority is a central, on-line service which can be queried to validate certificates. The intention is to allow greater interoperability between PKIs by hiding the multiple CAs from users and services. The VA becomes the trust anchor for relying parties.

The DNV report claims that “So far, no effort at establishing large-scale PKI interoperability has succeeded.” However, the experiences of PKI use with tens of CAs and thousands of users in the European and international grids would tend to contradict this statement (as indeed would the use of browser trust lists to some extent). The VA approach to interoperability is an alternative to building trust structures among CAs (such as cross-signing and bridges, as discussed in section 2.2.5 above), which the author claims are “inherently incapable of solving the problem.”

The VA will be able to give a “classification” (or “quality indicator”) for certificates passed for validation. This is the subject of ongoing development work [136]. Objective criteria for classification are to be devised based on existing standards coming from the US Federal Bridge Certification Authority, EU Qualified certificates, and ETSI, for example. The classification of a CA is based on policy and other documentation, perhaps including commercial and financial background of the CA operator. “Quality” could be indicated as a numerical value or a more fine-grained data structure. The VA will compare the requirements of a relying party with the classification and return a result based on the comparison. This represents a step towards a trust evaluation system.

The implementation provides a web service based on XKISS from XKMS for the relying parties [87]. The VA fetches CRLs from CAs and also uses OCSP to do on-line status checking of certificates. It may also use LDAP to fetch certificates from the CA when only a reference is passed to the VA. Historical information is stored locally by the VA which can be used for audit purposes.

A mechanism for revoking trust in the VA itself is not described in the technical report. If the VA uses a conventional host or service certificate issued by a CA that is trusted by all RPs then the VA’s certificate may be revoked by the CA in the usual manner. However, if the VA uses a self-signed certificate then presumably the relying parties will have to rely on out-of-band methods for receiving information on the VA status. This may be acceptable in the case where a VA accepts financial and legal liability for breaches.

The author of the DNV VA White Paper, is a co-author of a US patent application for *Single Sign-on Secure Service Access* [148], which appears to cover the validation service described above. However, the NSI PKI server described above could be considered to be prior art for the VA.

CertiVeR & OGRO

CertiVeR offers commercial certificate revocation status services using the OCSP protocol and has promoted the service to the grid community [38]. CertiVeR have also developed an OCSP client toolkit known as the Open Grid OCSP (OGRO) library, which provides support for validating proxy certificates, and meets many of the OCSP client requirements from the OGF CA Operations working group. The OGRO client can be configured with a simple validation policy which determines which OCSP responders to trust and what parameters to include in requests.

The OGRO authors have also looked into techniques to cache the results ofOCSP validation as attributes in proxy certificates, providing a form of pre-validation. This can provide an increase in performance over standard OCSP use [110].

US Grid Research & Higher Education Validation Service

There is some work under way in the US among PKI experts in the grid research and higher education communities to establish a validation service for production use in these areas [90].

2.4.2 Trust Evaluation

Chadwick, Ball & Basden

Chadwick, Ball & Basden describe a trust evaluation system based around an expert system, IStar [16]. The expert system requests CP/CPS details either directly from a relying party or automatically from a *CPS server*, which stores CP/CPS documents in an XML format. The expert system uses this information to reach an answer.

In contrast to the *static* evaluation of a CA based on its CP/CPS the authors identify several sources if information for *dynamic* evaluation. These include: publicly available information; information available to subscribers (e.g. details of the certificate enrolment process); information available to relying parties (e.g. CRLs); information available to external auditors (e.g. private CA operation logs); and information from an audit of the CA software and hardware. An *audit certificate*, which consists of an attribute certificate that contains the results of an audit presented in the previously mentioned XML format, is used to include external audit information in a dynamic evaluation.

This trust evaluation system has the advantage of support for static and dynamic evaluation. However, dynamic evaluation only makes use of CRLs and audit certificates, and the system is not set up for use as an on-line service. Future work suggested by the authors includes adding the concept of *reputation* to the system and including several of the currently unsupported sources of dynamic audit information.

Casola et al.

Casola et al. present a methodology to automatically evaluate and compare CA security policies for purposes of cross certification [35]. The authors present a Reference Evaluation Model. This consists of three parts: *formalization*, choosing the representation of the security policy; the evaluation *technique* to be used to compare policies, based on the formalization; and *reference levels*, to which the policies can be compared.

Some example applications of the methodology have been conducted. The initial formalization step produces an XML representation of an RFC 3647 CP/CPS [42]. The authors present two evaluation techniques: one based on fuzzy sets and another based on a metrical space technique.

The fuzzy technique assigns fuzzy judgements to each provision in a security policy. The individual judgements are aggregated to give an overall judgement (using Ordered Fuzzy Number Weighted Averaging). Various graphs can be produced to help the evaluator. The metrical space technique represents the policy space as an $n \times m$ matrix where n is the number of provisions in the policy and m is the number of security levels. A distance function provides a measure of the difference between two such matrices. A matrix is created for each policy to be compared. The distance function is used to calculate if a given policy is stronger than another.

The reference levels are set by creating policies according to the appropriate technique and which correspond to some externally set security level. The authors give some examples of applying the methodology to various situations such as establishing hierarchical and bridge certification models.

The work of Casola et al. represents a significant contribution to research in the area of automatic trust evaluation of CA policies. The formalization and evaluation techniques have been investigated rigorously. CertiVeR, mentioned above, are working with Casola et al. with a view to incorporating this evaluation system into the CertiVeR on-line validation services [109]. The approach is limited to evaluation of a CA's *policies* and does not involve other sources of information which might reveal the CA's actual practices as determined by an auditor or by examining certificates and other such material.

Trust Matrices

An automatic trust evaluation system was developed in the European DataGrid CACG by Brian Coghlan during 2002 [46], and later presented in [14].[9] This grew out of the manual assessment of CAs in the CACG.

[9]On-line CA Trust Matrices: `http://www.cs.tcd.ie/coghlan/cps-matrix/cps-matrix.cgi`

As mentioned in the description of the grid CA accreditation process, each CA must have CP/CPS documentation. Some of the *features* from the CA's policy and practice statements were encoded in a CA report file. The CA report file was written as a tree structure with key–value pairs at the leaf nodes. The *rulesets* against which the CAs were evaluated were written in the same form. The language was designed to enable later extension for full expression evaluation, polymorphism and matching capability to allow formal analysis, but was initially very simple.

Features are evaluated relative to rulesets. Rules are specific to a particular feature but any number of rules can be defined per feature. The concept of assurance levels is accommodated to allow rulesets to be defined for each level specified by GGF CA Operations working group [30]. Manual third-party evaluations (which can be considered a form of audit) are provided for within an appendix section to each report file.

A *default ruleset* was defined based on the EDG CACG minimum requirements. Each Virtual Organisation (VO) can also define their own rules that override and extend the default ruleset, and each CA can do likewise, overriding and extending the former rulesets. This *ruleset inclusion principle* extends from the general to the specific. It can be extended to users, hosts and even specific services simply by defining the appropriate ruleset. Thus a typical chain obeying this principle might be: default ruleset → VO ruleset → host ruleset. It is not necessary for a typical subject to have all possible rulesets in their possession, only those rulesets in the inclusion chains that they are interested in.

The software was written as a Perl CGI script. It reads in the CA report files from a specified directory and they are parsed with a regular-expression-based parser which handles the CA features in a fixed order. Feature values are recorded in a multi-dimensional associative array. To produce the feature matrix these features can be output as a HTML table. The acceptance matrix is produced by evaluating the features against the rulesets. The software initially had performance issues that resulted in evaluation times of the order of 30 seconds; eventually this was reduced to approximately one second.

Rulesets can be specified in the same form as the reports. In this case the feature values are replaced with conditionals (e.g. $=, \neq, <, \leq, >, \geq$) a match value and a rule result as a triple of security level, severity class and a weight between 0.0 and 1.0. The model used for trust evaluation by the trust matrices is described in more detail in Chapter 3.

The author's principal work in this thesis follows on from Coghlan's work on trust matrices for grid CAs described above.

2.5 Computational Trust

In the real world, trust is sometimes seen as something difficult to pin down, although we generally know what it means to trust or not to trust. Computational trust is the field of research that tries to deal with trust as a computational concept – as something that can be computed and used by computer systems.

An important and widely-cited work which essentially founded the field of research of computational trust is Marsh's doctoral thesis [113]. Marsh investigates the concept of trust as it is used in philosophy, psychology and the social sciences and produces a formalism which allows the concept to be discussed precisely. He also presents a simple implementation of the formalism in the context of artificial agents to demonstrate its practical use.

In Marsh's formalism, trust is divided into basic trust, general trust, and situational trust. Basic (or *dispositional*) trust represents an agent's trusting disposition. This is denoted T_x for an agent x. An optimist would have a high basic trust which will apply in all situation and with all agents, whereas a pessimist would have a low basic trust. This level of basic trust can be altered by an agent's experiences. General trust is one agent's trust in another. Agent x's trust in agent y is denoted $T_x(y)$. An agent x may start with an initial trust level in agent y based on x's dispositional trust and thereafter x's general trust in y will depend on y's behaviour towards x. Situational trust introduces the notion of a *situation* in which the trust relationship takes place. An agent's trust in another agent may of course vary depending on the situation or circumstances. Agent x's trust in agent y in situation α is denoted $T_x(y, \alpha)$.

2.5.1 Trust Metrics

Marsh uses a range of real values from -1 to $+1$, where -1 represents complete distrust and $+1$ represents absolute, blind trust. The extreme value of $+1$ is excluded from the range in the formalism since it denotes absolute trust and hence a lack of real consideration of the object of trust: in short, credulity. The issues of sensitivity and subjectivity of such trust assignments are discussed

In Marsh's system 0 represents a 'neutral' trust which is distinct from the complete distrust represented by -1. Other approaches use a range real values from 0 to 1, where again 1 represents complete trust, but 0 can now be taken to mean complete lack of trust or neutral trust based on lack of *information* about trust. The Ebay feedback rating system has a range of $-\infty$ to $+\infty$ with each unit corresponding to a single negative or positive rating by a unique member. In practice the score is summarised by the positive rating percentage, that is, the number of positive ratings from unique members divided by the total number of unique members who have left a rating [59]. This sort of trust metric is a measure of a user's *reputation.*

An Ebay member rating can also be thought of as a vector in three dimensions of positive ratings, neutral ratings, and negative ratings. The SECURE methodology uses a tuple of values for supporting evidence, inconclusive evidence, and contradictory evidence [32]. The important point is that positive ratings, rather than directly cancelling out negative ratings, simply add to the available evidence for some trust value.

2.6 Discussion

Some previous work in the field of trust evaluation has taken the approach of comparing one policy against another or against an ideal policy as a means of evaluation [35]. The approach of applying a set of rules to a set of features appears to be an alternative that has not been widely considered for this purpose. In practice, such rules may implement a direct policy comparison, but they can also be more generally used to implement other forms of evaluation.

The existing trust evaluation systems are not (yet) integrated with validation services. The trust evaluation systems can be used to aid decision-making in establishing trusted sets of CAs or cross-signing agreements, but they cannot be used as part of on-demand validation. A trust evaluation component for the DNV service discussed above is under development. However, the DNV project was unknown at the time the work for this thesis began and is apparently still not complete at the end of the thesis work. The work of Casola et al. may be incorporated into the CertiVeR on-line validation service but this is also not complete at the time of writing.

Other than Coghlan's trust matrices, the trust evaluation systems mentioned above have not been developed specifically to meet the requirements of the grid community. While the PKI systems used in grids are not greatly different from those employed elsewhere, the technical differences, such as the use of delegation with proxy certificates, intermediate services and credential stores, and the large-scale operation across administrative boundaries give rise to requirements that may not have been addressed in other work.

The state of the art suggests that there is an opportunity for a form of on-demand – that is, on-line and automatic – trust evaluation that has been developed with grid authentication in mind.

2.6.1 Summary

This chapter has given a broad overview of grid computing, existing authentication and authorisation systems, and an introduction to the field of computational trust. The discussion indicates that there is a place for automatic trust evaluation in grid authentication. In the next chapter some principles and a model of trust evaluation are introduced.

Chapter 3

Principles of Trust Evaluation

3.1 Introduction

The *policy bridge* approach to large-scale public key infrastructure, described in Section 2.2.5, has some of the features of a true cross-signed bridge but without the direct cryptographic support for verification offered by cross-signing. This makes it more flexible in the sense that compliance with a stated policy and procedures is all that is required for inclusion in the infrastructure. However, it does mean that it lacks an automatic system for verifying if a given CA has been accredited, or meets the criteria for accreditation.[1] There are at least two events at which such an automatic evaluation would be useful: when assessing a new CA for accreditation by the Policy Management Authority; and when verifying that a previously accredited CA still meets the requirements. The first of these would also be useful in establishing true bridge CA infrastructures. The second corresponds approximately to verification of the bridge CA signature.

Such evaluation of CAs is already performed by Policy Management Authorities when admitting new members, but the process is certainly not automatic and could possibly benefit from a level of formalisation and automation. The process is currently governed by human-readable documents written and interpreted by the PMA members. The approach to evaluation presented in this thesis is based on a semi-formal model where sets of *rules* are applied to the *features* of a CA and the results form an assessment of the CA. The *rulesets* (sets of rules) and CA *descriptions* (describing the CA's features) will still be produced by fallible human beings but the formal approach and machine-readable documents allow for a more consistent and automatic assessment.

This evaluation for accreditation, however it is performed, can be considered an evaluation of *trust* in the CA. The CAs policies and procedures, its cryptographic features, the security of CA hardware and software, its physical environment and personnel, and other qualities all contribute

[1] Throughout this thesis the objects of evaluation are are X.509 Certification Authorities. However, there is nothing in the model that would preclude its use in evaluating other forms of credential authorities, validation services, etc.

to its trustworthiness. Conversely if the CA has been accepted then its identity assertions in the form of certificates can be trusted and so it can be relied upon for authentication.

The trust evaluation is done by or on behalf of a *relying party*, an entity that wants to make some decision based on the trust established in the CA. The relying party could be a service provider granting access to a computing resource, or a user wishing to connect securely with a service.

In practice, the decision to trust the CA is often removed from individual users or services and is made at the level of an organization (real or virtual). In a grid context there may be multiple levels to the decision based on policies of, say, international projects, virtual organizations, member sites, and individual hosts or users. These policies may be merged to give an overall effective policy for a particular entity.

3.1.1 Overview

This chapter describes a model for trust evaluation. In the following sections the model is introduced (§3.2) and the creation of rulesets and CA descriptions is discussed (§3.3). The definition of ruleset algebras is described and some example algebras are presented (§3.4). Finally, several approaches to chaining multiple rulesets are introduced (§3.5).

3.2 A Model of Trust Evaluation

This section outlines the basic concepts of the model of trust evaluation. As mentioned above, in this model sets of rules are applied to the features of a CA and the results form an assessment of the CA. Rules are grouped into *rulesets* and a CA's static features are presented as a CA *description.*

A *feature* can be any attribute of the CA that can be observed and that is useful for assessing its trustworthiness. This will include information gathered from the CA's published procedures and policies, internal documentation, audit reports, issued certificates and certificate revocation lists, and external knowledge about the CA's software, hardware, etc.

A ruleset is written to encode the trust policy of a relying party, an organisation, or a policy authority. The ruleset consists of a number of rules each of which is a function that maps one or more CA features to a result of some sort (ruleset results are discussed in Section 3.4). Rules within a ruleset are independent of one another and multiple rules may refer to the same CA feature.

A formal model makes it possible to reason about trust evaluation independently of a particular implementation. The following table introduces the basic concepts of trust evaluation using a subset of the notation of the Irish School of the Vienna Development Method (*VDM*♣) [112] – see Appendix B for an overview of the notation. It should be noted that the *VDM*♣ formal method has not been used in developing the models which follow.

Feat	Set of features
$\mathit{Desc} = \mathbb{P}\mathit{Feat}$	Set of CA descriptions
Result	Set of rule result values
$\mathit{Rule} = \mathit{Desc} \rightarrow \mathit{Result}$	Set of rules (functions from description to result)
$\mathit{Ruleset} = \mathbb{P}\mathit{Rule} = \mathbb{P}(\mathit{Desc} \rightarrow \mathit{Result})$	Set of rulesets (sets of rules)
$\mathit{Aggregate}$	Set of aggregate evaluation result values
$\mathit{Acceptance}$	Set of acceptance result values

$\mathrm{Eval} : \mathit{Desc} \rightarrow \mathit{Ruleset} \rightarrow \mathit{Aggregate}$	Evaluation of a CA description against a ruleset
$\mathrm{Agg} : \mathbb{P}\mathit{Result} \rightarrow \mathit{Aggregate}$	Aggregation of rule results to overall result
$\mathrm{Acc} : \mathit{Aggregate} \rightarrow \mathit{Acceptance}$	Acceptance of result

In this model the term *aggregation* is used to mean any form of processing or transformation of the individual ruleset result values, $r \in \mathit{Result}$, to produce an aggregate value, $a \in \mathit{Aggregate}$. The aggregate a could be a single scalar value, a sequence composed of the individual result values, or a more complex multi-dimensional structure.

The term *acceptance* is used to mean the comparison or other checking of the aggregate result against some threshold, template or ideal result. Note that the object of comparison does not appear in the function signature for the Acc operator. In practical applications of trust evaluation the acceptance step may not be required or may be performed outside of the trust evaluation system. An application may make direct use of the aggregate result.

The Eval operation is defined to apply each rule in the ruleset to the description, and to aggregate the results.

$$
\begin{aligned}
&\mathrm{Eval} : \mathit{Desc} \rightarrow \mathit{Ruleset} \rightarrow \mathit{Aggregate}\\
&\mathrm{Eval}(d, R) \mathrel{\widehat{=}} \mathrm{Agg}(\mathbb{P}\mathrm{Eval1}(d)R)\\
&\quad \textbf{where}\\
&\qquad \mathrm{Eval1} : \mathit{Desc} \rightarrow \mathit{Rule} \rightarrow \mathit{Result}\\
&\qquad \mathrm{Eval1}(d, r) \mathrel{\widehat{=}} r(d)
\end{aligned}
$$

In this definition of Eval, the Eval1 operation is curried with the description argument and the curried function is applied to each member of the set of rules to produce a set of results, which are then aggregated by the Agg operation.

3.3 Rulesets & Descriptions

Rulesets and descriptions are central to this model of trust evaluation and so the methodology used for defining them is very important. For both rulesets and descriptions, the choice of relevant features has a strong influence the nature of the evaluation. Assigning result values for the outputs of rule functions is the other major influence on the character of a ruleset.

3.3.1 Producing Rulesets & Descriptions

A clear methodology for producing rulesets and descriptions for evaluation is required. One approach is to develop the rulesets first and then to create the descriptions based on these. The following describes a process for ruleset development.

1. The policies which will form the basis for the ruleset are identified. At this stage a ruleset algebra and a system for assigning values to rule results should also be decided.
2. The policies are analysed and individual provisions are extracted. This may be easier if the policy has a clear structure, such as the template provided by RFC 3647 [42]. More work may be required if the policy is written in an informal style.
3. For each provision in the policy the relevant *features* are determined. These are the qualities and quantities that will be evaluated. It is important to strike a balance between features which are very specific to a single rule and those which are more general but will require more complex rule calculations – see Section 3.3.2 below.
4. For each provision a rule function is determined which maps the feature values to a rule result. This step ties the ruleset to a particular ruleset algebra: if it were later decided to use a different ruleset algebra, each rule would potentially have to be rewritten to use a different set of result values with their own semantics.

A related procedure is used to develop CA descriptions.

1. The rulesets which will be applied to the CA description are chosen. It may also be appropriate to refer to the original policies, or to some general template such as RFC 3647.
2. A set of features is extracted from the rule functions of the rulesets, and the other policies and templates. There may be some iteration required between this step and the feature-determination step (step 3) of the ruleset creation methodology.
3. A template or schema for the CA description is created based on the set of features.
4. CA descriptions are written based on the template using information gathered from the CA's policies and procedures.

The methodology presented here assumes that the features of interest of a CA are determined primarily by the rulesets that will be applied. This is certainly the case but it may be more appropriate for the set of features to be determined independently of the creation of the ruleset or the CA descriptions. For example, a high-level organisation such as a PMA might agree a standard set of features based on the RFC 3647 template. These features can then be used for the rule creation step of the ruleset process (step 4) and the template creation step of the CA description process (step 3).

3.3.2 Features

For a CA to be evaluated against a particular ruleset, the ruleset determines which CA features must be known. There are a number of types of CA features:

- Static – information from CA policy
- Dynamic – information from CA practices, such as certificates and CRLs
- Historical – comparison over time, such as issuance of CRLs on time or issuance of certificates with non-unique serial numbers
- Global – comparison based on all CAs, such as global uniqueness of CA DNs

CA policy documents can be considered *static* since they are slowly-changing information published by the CA. Actual CA practices, as revealed through issued certificates, certificate revocation lists, responses from OCSP and other services, can be considered *dynamic* since they are more rapidly-changing.

Ball et al. distinguish between *static trust calculation*, based on information published by the CA, and *dynamic trust checking*, where the actual performance of the CA is evaluated [16]. In their system several sources of information are used for dynamic trust checking:

> "what the [relying party] already knows about the CA, what the CA's external auditor publishes about the CA's operations, and finally what the CA makes known about itself through the publication of its Certification Revocation Lists."

Additional sources of information that can be used include the properties of the CA root certificate and end-entity certificates, such as key length, validity period, and certificate extensions. These can be compared against the CA's published policy and against the requirements of relying parties and PMAs. Furthermore, a certificate issued by a CA must, for example, have a unique serial number and must fall within the relevant namespace declared by the CA.

Certificates can be evaluated on a historical basis to check that all certificates seen from a particular CA meet these requirements. Similarly for CRLs, the update period can be evaluated with reference to the CA's published policy and RP requirements, and also on a historical basis to assess a CA's past performance in this matter.

While some of these factors do not directly relate to the cryptographic aspects of a CA, the presence of such provisions in policies indicates that they are considered important for a CA's acceptance by the community of relying parties. One example is the key-length of end-entity certificates issued by a CA. If the concern is only with the cryptographic validity of the CA's signature on the end-entity certificate then the end-entity key length does not come into the decision of whether to trust the CA. In theory, it is up to the relying party to decide if it is willing to accept an authentication based on a certificate with a short key length, and therefore weak cryptographic strength. In practice, however, it is often the case that a RP will treat all certificates from a CA as equally valid, and so the responsibility falls back on the CA – if it is willing to accept it – to ensure the quality of end-entity key material.

Feature Complexity

When developing rulesets and descriptions there is a choice to be made between complex features and complex rules. Suppose a policy requires that a CA has attributes a_1 to a_n all with suitable values in order to satisfy a provision P. One approach is to represent this as a complex feature p which indicates whether or not the CA satisfies the provision P. A rule r_p needs only make a simple check that p is TRUE. Such a rule would still return a value appropriate for the ruleset algebra in use, which might be Boolean, numeric, symbolic or something else. The complex feature p effectively hides the values of the relevant attributes. Since they are not exposed as features in their own right they cannot be evaluated by other rules.

If the attributes are exposed directly as simple features f_1 to f_n, then r_p must become more complex and it must evaluate the features against the requirements of provision P. It would seem reasonable that this should be done wherever possible: it means the evaluation will be expressed in a formal way in the relevant rule. However, there may be some instances were it is difficult to express a rule in any way other than "does the CA comply with the provision P?" There may also be cases where extracting multiple features from a policy provision is unnecessary if the rule implementing the provision must simply combine them again and there is no conceivable use for the features apart from that rule.

Complex features essentially require that the provision is interpreted and applied to the object of evaluation when the description is being created. This ties the feature to a particular policy meaning it will probably be of little use to other policies. There is also the issue of who decides the value for such a feature. If the CA creates it then it is essentially evaluating itself.

This introduces the issue of who creates CA descriptions. Typically a CA is responsible for producing its own CP/CPS documents, and the static CA description will often be seen as a reformatting of these. The description could be written by an auditor, in which case it becomes a form of audit report. The description might be written by or on behalf of the CA and then approved as accurate – and perhaps digitally signed – by a PMA.

3.3.3 Results

An important part of constructing a ruleset is to decide how values are assigned to rule results. The creation of the rule functions and the assignment of result values effectively give meaning to the ruleset. This is one of the most important parts of creating a ruleset and requires a significant amount of input – possibly subjective – from the ruleset designers.

To choose suitable values the designers can either use some pre-existing classification or scoring system, or develop a new system from the policies in question and experience in the evaluation domain. Solutions to the problem may also lie somewhere in between.

For the first approach, the policy itself may provide enough information to give values to rules. At the most basic level, Boolean values could be used to correspond to pass/fail outcomes for individual policy provisions. In the case of a ruleset for the IGTF authentication profile for traditional CAs, the values could be chosen to match the use of MUST or SHOULD (as defined in RFC 2119 [27]) in the authentication profile.[2]

A selection of CP documents for CAs of different security levels could be used to provide an indication of the requirements for these security levels. Each rule could evaluate a given CA feature to determine which level it would be in. This is closely related to the approach taken by Casola et al. [35], where the policy of a CA under examination is compared using various techniques against policies of known security levels.

The second approach requires that knowledge of the policies and the problem domain are applied to develop a reasonable rating system. To initialise a trust evaluation expert system Chadwick and Basden [39] took the approach of

> "interviewing PKI experts and asking them to rank the various factors against each other in order of importance on a scale from 1 to 10. The experts could agree that some factors were more important in the calculation of trust than others, but for other factors there was no general agreement." (taken from [16])

When looking beyond Boolean rule results judgement is required to assign appropriate values. One higher level choice is between discrete values, for example, multiple security levels, and continuous values, for example, real numbers between 0.0 and 1.0. It may be easier and make more sense to assign particular evaluation outcomes to the natural – and fuzzy – classifications indicated by security levels than to decide, for example, what percentage score is appropriate.

Whether devised from scratch or based on existing policies and classification systems, rule results are defined in relation to a ruleset algebra.

[2]Such a ruleset is presented in Appendix C.

3.4 Ruleset Algebras

A RULESET algebra determines the types of the rule results and how these results are aggregated and tested for acceptance. The chosen algebra affects the importance of individual rules and extreme results in determining an aggregate result.

3.4.1 Result Types

The most basic set of rule result values is the Boolean set. Each rule result will indicate if the evaluation of that rule was acceptable or not. Boolean results for an entire ruleset can be aggregated with a logical-AND operator. However, this is a very crude approach as a single FALSE value will lead to rejection.

An alternative is to use the range of real numbers 0.0–1.0. This has precedent in probability, fuzzy logic, and the trust metrics described in Section 2.5. When working with probabilities, multiplication of independent probabilities will give the combined probability. In fuzzy logic, the fuzzy-AND is usually defined as the minimum value of a set of independent fuzzy set membership functions. An arithmetic mean or weighted arithmetic mean could also be used. Of course, these different functions will produce different overall results and so the assignment of values to rules must be made with this in mind.

The trust evaluation model is not limited to simple numerical values. Results can return symbolic values, e.g. LOW, MEDIUM, HIGH; values with attached weights, e.g. `(result (value 0.8) (weight 0.2))`; arbitrary tuples or lists; or function values. In each case the aggregation function must be chosen to handle the result types. For some applications, instead of reducing the evaluation result to a single value it may be more appropriate for the result set to be compared against a corresponding acceptance set of individual rule scores for an exact match or partial match.

3.4.2 Defining a Ruleset Algebra

To define a ruleset algebra it is necessary to define the *Result*, *Aggregate*, and *Acceptance* types and the Agg and Acc operators. The algebras are characterised by the result types and aggregation methods used. The following sections demonstrate the definition of ruleset algebras based on weighted averages, fuzzy sets, summation and threshold, severity class and level, and template matching.

3.4.3 Averages of Real Numbers in the Range 0.0–1.0

A simple ruleset algebra can be created by using real numbers as the values assigned to rule results. The value represents some measure of 'goodness' of the CA with respect to a particular rule. A simple approach to aggregation is to take the arithmetic mean of the rule results.

$$\begin{aligned}
&\mathit{Aggregate} = \mathit{Result} = \{\, r \in \mathbb{R} \mid 0.0 \leq r \leq 1.0 \,\}\\
&\quad \mathrm{Agg} : \mathbb{P}\mathit{Result} \rightarrow \mathit{Aggregate}\\
&\mathrm{Agg}(R) \mathrel{\widehat{=}} \bar{R}\\
&\quad \textbf{where}\\
&\qquad R = \langle r_1, r_2, \ldots, r_n \rangle\\
&\qquad \bar{R} = \frac{\sum_{i=1}^{n} r_i}{n}
\end{aligned}$$

An acceptance function compares the aggregate result against a single threshold and returns a Boolean value. The threshold for acceptance must be set bearing in mind that a high aggregate result may conceal a number of low results.

$$\begin{aligned}
&\mathit{Acceptance} = \mathbb{B}\\
&\quad \mathrm{Acc} : \mathit{Aggregate} \rightarrow \mathbb{B}\\
&\mathrm{Acc}(a) \mathrel{\widehat{=}} a \geq 0.80
\end{aligned}$$

Combining results with a weighted arithmetic mean allows weights to be assigned to individual rules based on their importance for the overall trust evaluation. The weights can be set based on factors such as the risk or severity of the issue.

$$\begin{aligned}
&\mathrm{Agg}(R, W) \mathrel{\widehat{=}} \bar{R}_W\\
&\quad \textbf{where}\\
&\qquad R = \langle r_1, r_2, \ldots, r_n \rangle\\
&\qquad W = \langle w_1, w_2, \ldots, w_n \rangle\\
&\qquad \bar{R}_W = \frac{\sum_{i=1}^{n} r_i \cdot w_i}{\sum_{i=1}^{n} w_i}
\end{aligned}$$

This ruleset algebra is appealing for its simplicity (especially in the unweighted form) but it has some weaknesses. As mentioned above, extreme results may be masked by a large number of results close to the mean. Analysis of false-positives and false-negatives would be needed when using such an algebra in practice.

3.4.4 Fuzzy Sets

Fuzzy sets [178] appear to present an attractive approach to evaluating trust and indeed fuzzy techniques have been applied for this purpose [34, 36].

In a simple fuzzy-set–inspired algebra, each rule in the ruleset may be considered to be a membership function for a fuzzy set, for example, the fuzzy set of CAs with a high key length. The fuzzy set membership values are given by real numbers in the range 0.0 to 1.0. Each CA will be a member of as many fuzzy sets as there are rules in the ruleset, to the degree of fuzzy membership determined by the evaluation of each rule.

Conventionally, the intersection of fuzzy set membership values is calculated as the minimum of the membership values. This gives intuitively reasonable results in many situations. For example, the intersection of two equal membership values is equal to the membership value. This contrasts with the intersection of probabilities, which is calculated by multiplication. If $\tilde{A}$ is the fuzzy set of CAs with high key size and $\tilde{B}$ is the fuzzy set of CAs with frequent CRL issuance then the intersection $\tilde{A} \cap \tilde{B}$ is the fuzzy set of CAs with high key size and frequent CRL issuance. Supposing a CA c has fuzzy membership 0.4 of $\tilde{A}$ and 0.5 of $\tilde{B}$, i.e.:

$$\mu_{\tilde{A}}(c) = 0.4 \qquad \mu_{\tilde{B}}(c) = 0.5$$
$$\mu_{\tilde{A} \cap \tilde{B}}(c) = \min(0.4, 0.5) = 0.4$$

Using $\times$, as for intersection of probabilities, would give $0.4 \times 0.5 = 0.2$.

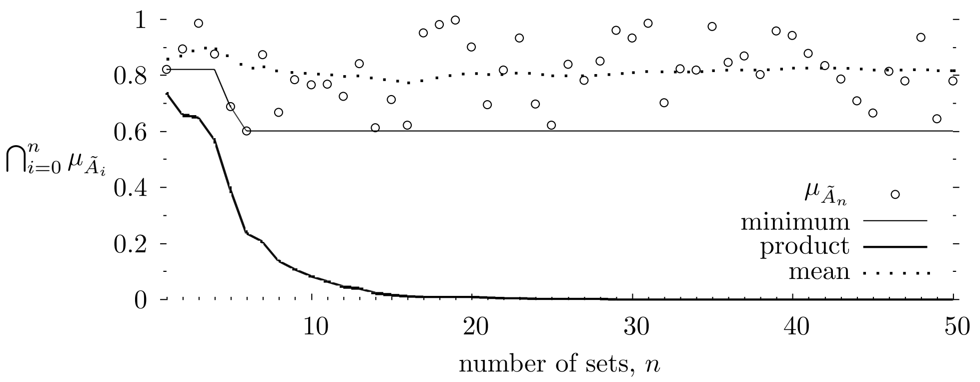


Figure 3.1: Intersection with increasing number of sets

Intersection can be applied to many sets, for example, all the rules in a ruleset. Figure 3.1 shows the minimum, the product and the mean for fifty randomly generated membership values in the range 0.6–1.0. As more sets are added to the intersection, the product tends rapidly towards zero while the minimum by definition is bounded by the lowest value. An example Fuzzy-Logic–inspired aggregation function, taking the minimum as the intersection function, is:

$$
\begin{aligned}
&\mathit{Aggregate} = \mathit{Result} = \{\, r \in \mathbb{R} \mid 0.0 \leq r \leq 1.0 \,\}\\
&\quad\mathrm{Agg} : \mathbb{P}\mathit{Result} \rightarrow \mathit{Aggregate}\\
&\mathrm{Agg}(R) \mathrel{\widehat{=}} \min(R)\\
&\mathbf{where}\\
&\quad R = \{r_1 \ldots r_n\}
\end{aligned}
$$

An acceptance function compares the aggregate result against a single threshold and returns a Boolean value.

$$
\begin{aligned}
&\mathrm{Acc} : \mathit{Aggregate} \rightarrow \mathbb{B}\\
&\mathrm{Acc}(a) \mathrel{\widehat{=}} a \geq 0.85 \;\rightarrow\; \textsc{True}\\
&\qquad\qquad a < 0.85 \;\rightarrow\; \textsc{False}
\end{aligned}
$$

A second, more complex, acceptance function classifies the result into one of three security classes.

$$
\begin{aligned}
&\mathit{Acceptance} = \{\, \textsc{Low}, \textsc{Medium}, \textsc{High} \,\}\\
&\quad\mathrm{Acc} : \mathit{Aggregate} \rightarrow \mathit{Acceptance}\\
&\mathrm{Acc}(a) \mathrel{\widehat{=}} a < 0.3 \;\rightarrow\; \textsc{Low}\\
&\qquad\qquad 0.3 \leq a < 0.7 \;\rightarrow\; \textsc{Medium}\\
&\qquad\qquad 0.7 \leq a \;\rightarrow\; \textsc{High}
\end{aligned}
$$

In an alternative fuzzy model each rule could return a tuple of membership values for a number of fuzzy sets corresponding to security assurance levels (None, Low, Medium, High). The results for each assurance level from all the rules can be combined to give a tuple of fuzzy set membership values. These can be created by a fuzzy composition such as the maximum, the sum or the mean of the values.

$$
\begin{aligned}
&F = \{\, r \in \mathbb{R} \mid 0.0 \leq r \leq 1.0 \,\}\\
&\mathit{Result} = \mathit{Aggregate} = F \times F \times F \times F\\
&\quad\mathrm{Agg} : \mathbb{P}\mathit{Result} \rightarrow \mathit{Aggregate}\\
&\mathrm{Agg}(r) \mathrel{\widehat{=}} (\mathrm{Mean}(\mathbb{P}\pi_1(r)), \mathrm{Mean}(\mathbb{P}\pi_2(r)), \mathrm{Mean}(\mathbb{P}\pi_3(r)), \mathrm{Mean}(\mathbb{P}\pi_4(r)))
\end{aligned}
$$

The projection function π_i takes the ith element from a tuple. When applied to a set of tuples with the set functor $\mathbb{P}$ the result is a set containing the ith element of each tuple. For example,

$$\mathbb{P}\pi_2(\{(0.0, 0.2, 0.7, 0.5), (0.1, 0.4, 0.8, 0.6), \ldots\}) = \{0.2, 0.4, \ldots\}$$

The fuzzy values of the four assurance levels can be further aggregated. The simplest technique is to take the maximum: the security level with the highest mean score is the security level for that evaluation.

3.4.5 Summation & Threshold

Another approach is to add rule results. If the total is below a given threshold then the CA can be accepted. 'Good' features of a CA can be given negative scores, while 'bad' features can be given positive scores with the magnitude depending on the importance of the issue. In this case there would be a maximum value over which the CA is considered untrustworthy. Alternatively, the sense of the results could be reversed and the CA must score over some minimum value to be accepted.

This example algebra is for summation of result values, positive or negative, and acceptance based on comparison with a threshold value.

$$\begin{aligned} \mathit{Aggregate} &= \mathit{Result} = \mathbb{R} \\ \mathrm{Agg} &: \mathbb{P}\mathit{Result} \rightarrow \mathit{Aggregate} \\ \mathrm{Agg}(r) &\mathrel{\widehat{=}} {}^{+}/r \\ \mathrm{Acc} &: \mathit{Aggregate} \rightarrow \mathbb{B} \\ \mathrm{Acc}(a) &\mathrel{\widehat{=}} a \geq 5.0 \end{aligned}$$

This is similar to the approach taken in spam filters such as SpamAssassin [114], where email messages are assessed against a set of rules, and the results reduced additively. Rules check the headers and body of the email message for known spam-like features, such as keywords in the Subject header, HTML text with the same background and foreground colour in the body, and use of the same address in both the To and From headers. Un-spam-like features, such as digital signatures, give negative scores. By default, if the message gets a score of 5.0 or more it is flagged as spam.

3.4.6 Severity Class and Level

This model is based on the system used in Coghlan's Trust Matrices software, described in Section 2.4.2, which was loosely based on fuzzy logic. The model used in that software was refined over the duration of the European DataGrid project and was eventually used for over twenty CAs that were members of the CACG. In this algebra a *condition* gives rise to an *issue*. An issue is of a particular *class*: minor, major or severe. An issue can also have a *coefficient* between 0.0 and 1.0 to allow weighting within the severity class. A notion of security levels (low, medium, high) can be added to the model: each issue is relevant to a specified level and to all higher levels.

The idea is that several major issues (with individual weights ≤ 1.0) might amount to a 'very' major issue, but would never amount to a severe issue. In principle the aggregation function is a sigmoid of the summation of weighted issues within a class, but in practice is limited to the maximum weight for the class. The result is a measure of severity for each class of issues. The overall severity is the aggregate coefficient of the most severe class with a non-zero aggregate coefficient. An acceptance level can be defined as the overall severity value below which the CA will be accepted, although it should be noted that the original software did not include such a step.

In the following model the Agg operation is defined in terms of reduction of the results r using the Agg1 operation. Agg1 is defined to override a maplet – i.e., a single mapping – in the result type *Aggregate*, which is a map from (*SeverityLevel* × *SeverityClass*) to *SeverityCoefficient*. The maplet is overridden with the greater value of 1.0 and $a(l, c) + x$, i.e., the sum of the passed coefficient parameter x and the existing value of the maplet for the specified *SeverityLevel* l and *SeverityClass* c in the map a.

$$\begin{aligned}
\mathit{Result} &= \mathit{SeverityLevel} \times \mathit{SeverityClass} \times \mathit{SeverityCoefficient} \\
\mathit{Aggregate} &= (\mathit{SeverityLevel} \times \mathit{SeverityClass}) \xrightarrow{m} \mathit{SeverityCoefficient} \\
&\textbf{where} \\
\mathit{SeverityLevel} &= \{\, \textsc{Low}, \textsc{Medium}, \textsc{High} \,\} \\
\mathit{SeverityClass} &= \{\, \textsc{Minor}, \textsc{Major}, \textsc{Severe} \,\} \\
\mathit{SeverityCoefficient} &= \{\, c \in \mathbb{R} \mid 0.0 \leq c \leq 1.0 \,\}
\end{aligned}$$

$$\begin{aligned}
\text{Agg} &: \mathbb{P}\mathit{Result} \rightarrow \mathit{Aggregate} \\
\text{Agg}(r) &\mathrel{\widehat{=}} {}^{\text{Agg1}}/r \\
\text{Agg1} &: \mathit{Result} \rightarrow \mathit{Aggregate} \rightarrow \mathit{Aggregate} \\
\equiv \text{Agg1} &: \mathit{SeverityLevel} \times \mathit{SeverityClass} \times \mathit{SeverityCoefficient} \rightarrow \mathit{Aggregate} \rightarrow \mathit{Aggregate} \\
\text{Agg1(l,c,x,a)} &\mathrel{\widehat{=}} a \dagger \{(l,c) \mapsto \max\{\,1.0, a(l,c) + x\,\}\}
\end{aligned}$$

In the following acceptance definition AccLevel determines if the severity coefficient is 0 for all severity classes in the specified level. AccLow is AccLevel with a curried l argument of Low and similarly for AccMed and AccHigh. AccMax determines the highest acceptance level for which the result is accepted.

$$\begin{aligned}
\text{AccLevel} &: \mathit{SeverityLevel} \rightarrow \mathit{Aggregate} \rightarrow \mathbb{B} \\
\text{AccLevel}(l,a) &\mathrel{\widehat{=}} \forall(\text{isZero})(\mathbb{P}(a(l))\mathit{SeverityClass}) \\
&\textbf{where} \\
&\text{isZero} : \mathbb{R} \rightarrow \mathbb{B} \\
&\text{isZero}(x) \mathrel{\widehat{=}} x = 0.0
\end{aligned}$$

The individual acceptance functions for each level are

$$\begin{aligned}
&\text{AccLow} : \mathit{Aggregate} \rightarrow \mathbb{B} && \text{AccMed} : \mathit{Aggregate} \rightarrow \mathbb{B} && \text{AccHigh} : \mathit{Aggregate} \rightarrow \mathbb{B} \\
&\quad \mathrel{\widehat{=}} \text{AccLevel}(\text{Low}) && \quad \mathrel{\widehat{=}} \text{AccLevel}(\text{Medium}) && \quad \mathrel{\widehat{=}} \text{AccLevel}(\text{High})
\end{aligned}$$

The appropriate security level is determined with AccMax

$$\begin{aligned}
\text{AccMax} &: \mathit{Aggregate} \rightarrow \mathit{SeverityLevel} \cup \{\,\text{None}\,\} \\
\text{AccMax}(a) &\mathrel{\widehat{=}} \mathit{AccLow}(a) \wedge \mathit{AccMed}(a) \wedge \mathit{AccHigh}(a) \;\rightarrow\; \text{High} \\
&\quad \mathit{AccLow}(a) \wedge \mathit{AccMed}(a) \;\rightarrow\; \text{Medium} \\
&\quad \mathit{AccLow}(a) \;\rightarrow\; \text{Low} \\
&\quad \text{otherwise} \;\rightarrow\; \text{None}
\end{aligned}$$

3.4.7 Template Matching

Instead of reducing the ruleset to a single value it may be more appropriate for the result set to be compared against a corresponding *acceptance set* of the same type. The previously presented ruleset algebras assume a level of equality of importance between rules, with weighting to compensate. A matching approach compares each rule result to a corresponding acceptance value. In this case, the Agg operation passes on the results unchanged.

$$Aggregate = \mathbb{P}Result$$

$$\mathrm{Agg} : \mathbb{P}Result \rightarrow Aggregate$$

$$\mathrm{Agg}(r) \mathrel{\widehat{=}} r$$

The Acc operation can be constructed from a set of expected results:

$$\mathrm{MkAcc} : \mathbb{P}Result \rightarrow (Aggregate \rightarrow \mathbb{B})$$

The Acc operation will compare each element in the result set with an element in the acceptance set.

$$\mathrm{Acc} : Aggregate \rightarrow \mathbb{B}$$

$$\mathrm{Acc}(A_g) \mathrel{\widehat{=}} \mathrm{MkAcc}(A_c)(A_g)$$

3.5 Chained Ruleset Evaluation

When evaluating trust in a CA there may be multiple policies in force which should affect the evaluation. In a grid environment, international authorities, virtual organizations and individual resource sites may all have relevant requirements for establishing trust in a CA. These policies may be evaluated separately, leaving the relying party to make a judgement on how the multiple evaluations are handled and how results are interpreted with respect to each other. Alternatively the policies may be combined or merged in some way and, conceptually at least, evaluated together. A well defined approach to merging rulesets is particularly important for automating the evaluation as it allows a single overall result, however complex, to be obtained.

In relation to the trust evaluation model proposed in this chapter, rulesets corresponding to different policies may be merged by *chaining* the rulesets together. There are two main approaches to chaining considered here: logical composition and functional composition. Logical composition means that rules – or their results – from one ruleset are replaced or overridden by those from another ruleset. Functional composition means that the rules within one ruleset are functionally composed with those from another ruleset: the outputs of one function become inputs to the next function.

Each form of chaining involves some matching between rules from different rulesets. Each rule has a name and this is used to match rules. Rulesets to be chained must have compatible result types and, more generally, ruleset algebras. This indicates that ruleset designers must keep chaining in mind.

When chaining rulesets there must be a defined order of precedence. In the following examples it is assumed that a more ‘general’ ruleset has lower precedence than a more ‘specific’ ruleset. For example, a policy from a local institution is given priority over one set by a higher level organization. This is in keeping with the common grid idea that sites should retain local control over their resources. Conversely, it is also possible to imagine a situation where local policies should have *lower* precedence than global policies.

Chaining by logical composition can take the form of replacement of rules before evaluation, or of results after partial evaluation of the chain. Result composition is considered first.

3.5.1 Chaining by Logical Composition of Results

An example of a simplified ruleset chain is presented in Table 3.1. The evaluation result values are in the range of real numbers 0.0–1.0.

Table 3.1: Example Ruleset Chain

Default Ruleset	VO Ruleset	Site Ruleset
Min. Subject Key Size $f(s) = \begin{cases} 1.0 & \textbf{if } s \geq 1024 \\ 0.0 & \textbf{otherwise} \end{cases}$	Min. Subject Key Size $f(s) = \begin{cases} 1.0 & \textbf{if } s \geq 2048 \\ 0.8 & \textbf{if } s \geq 1024 \\ 0.0 & \textbf{otherwise} \end{cases}$	CRL Publication Min. Frequency $f(p) = \begin{cases} 1.0 & \textbf{if } p \leq 14 \\ 0.0 & \textbf{if } p \geq 28 \\ \frac{p-14}{14} & \textbf{otherwise} \end{cases}$
Cert. Profile Version $f(v) = \begin{cases} 1.0 & \textbf{if } v = \texttt{X.509v3} \\ 0.0 & \textbf{otherwise} \end{cases}$	Cert. Profile Version $f(v) = \begin{cases} 1.0 & \textbf{if } v = \texttt{X.509v3} \\ 0.0 & \textbf{otherwise} \end{cases}$	Country ID $f(i) = \begin{cases} 1.0 & \textbf{if } i \in \{\texttt{ie}, \texttt{uk}\} \\ 0.0 & \textbf{otherwise} \end{cases}$
CRL Publication Min. Frequency $f(p) = \begin{cases} 1.0 & \textbf{if } p \leq 23 \\ 0.0 & \textbf{otherwise} \end{cases}$		

Applying these rulesets independently would give three sets of results. One simple approach to chaining with logical composition is to perform these evaluations in full and discard the results from lower precedence policies for which there are corresponding results in higher precedence policies.

A valuable enhancement is to evaluate the rulesets in a specified sequence keeping track of the partial results and replacing or adding individual rule results according to the specified precedence order. Two orders for sequential evaluation are from low-precedence to high-precedence, dubbed forward evaluation, and from high-precedence to low-precedence, dubbed reverse evaluation.

For forward evaluation, the ruleset of lowest precedence (the most ‘general’) is evaluated fully first. Then, for the higher-precedence (more ‘specific’) rulesets, each rule name in that ruleset is looked up in the partial result and, if found, the result will be replaced.

The worst case for this form of evaluation is when the rulesets of higher precedence are fully specified, i.e. all rules from previous rulesets must be overridden. If the high precedence rulesets correspond to local site policies that override just a small number of rules from the more general policies, as is expected to be the case, then this may not present a problem.

The data structure for the partial results should be chosen carefully to allow for efficient updates. A hash-table would be suitable. If a linked-list structure is chosen then it may be more efficient to simply add results from subsequent rulesets and use a second pass of the structure to get the final result.

For reverse evaluation, the highest precedence ruleset is evaluated first. As lower precedence rulesets are evaluated the rule names are looked up in the partial result and if they are not found the result values are added. This form of evaluation requires no replacement of results but it does require a large number of lookups.

Table 3.2: Ruleset Chain Evaluation

Default Ruleset	VO Ruleset	Site Ruleset
Forward: 1 →	2 →	3
{(Min-Subj-Key-Size, 1.0) **(Cert-Profile-Version, 1.0)** **(CRL-Pub-Min-Freq, 1.0)}**	**{(Min-Subj-Key-Size, 0.8)** **(Cert-Profile-Version, 1.0)** (CRL-Pub-Min-Freq, 1.0)}	{(Min-Subj-Key-Size, 0.8) (Cert-Profile-Version, 1.0) **(CRL-Pub-Min-Freq, 0.6)** **(Country-ID, 1.0)**}
Reverse: 3	← 2	← 1
{(Min-Subj-Key-Size, 0.8) (Cert-Profile-Version, 1.0) (CRL-Pub-Min-Freq, 0.6) (Country-ID, 1.0)}	**{(Min-Subj-Key-Size, 0.8)** **(Cert-Profile-Version, 1.0)** (CRL-Pub-Min-Freq, 0.6) (Country-ID, 1.0)}	**{(CRL-Pub-Min-Freq, 0.6)** **(Country-ID, 1.0)**}

In Table 3.2 the two ruleset chain evaluation orders are shown. Bold type indicates that a rule result has been added or replaced during the evaluation of that ruleset. Forward evaluation proceeds left-to-right from the most general ruleset to the most specific. Reverse evaluation proceeds right-to-left from the most specific ruleset to the most general.

3.5.2 Chaining by Logical Composition of Rules

In the above forms of chaining, results of the evaluation are passed along the chain, but it is also possible to consider the evaluation in terms of logical composition of rules. A single ruleset is built up from the component rules of the rulesets in the chain before the composite ruleset is applied. Such composite chained rulesets can be cached.

As for the logical composition of results, two sequences are considered. In the forward sequence, the ruleset of lowest precedence is taken as the starting point. Rules from the higher precedence rulesets replace lower precedence rules or are added to the intermediate ruleset.

In the reverse sequence, the process starts with the highest precedence ruleset and rules of lower precedence are added. Note that in this case there are no actual *replacements*.

Table 3.3: Logical Composition of Rules

Default Ruleset	VO Ruleset	Site Ruleset
Forward: 1 $\rightarrow$	2 $\rightarrow$	3
Min-Subj-Key-Size **Cert-Profile-Version** **CRL-Pub-Min-Freq**	**Min-Subj-Key-Size** **Cert-Profile-Version** CRL-Pub-Min-Freq	Min-Subj-Key-Size Cert-Profile-Version **CRL-Pub-Min-Freq** **Country-ID**
Reverse: 3	$\leftarrow$ 2	$\leftarrow$ 1
Min-Subj-Key-Size Cert-Profile-Version CRL-Pub-Min-Freq Country-ID	**Min-Subj-Key-Size** **Cert-Profile-Version** CRL-Pub-Min-Freq Country-ID	**CRL-Pub-Min-Freq** **Country-ID**

3.5.3 A Model for Chaining

In order to produce a model for chaining some refinement of the initial model for evaluation is required.

$Desc = FName \overset{m}{\rightarrow} Value$	Map of feature name to value
$Rule = Desc \rightarrow Result$	Function from description to result
$Ruleset = RName \overset{m}{\rightarrow} Rule$	Map of rule name to rule
$Resultset = RName \overset{m}{\rightarrow} Result$	Map of rule name to result value

Where a description was previously a simple set of features it is now a map from feature names to values. A *Ruleset* is now a map from rule names to rules. A *Resultset* type is introduced, which maps rule names to rule results.[3]

The Eval function must be modified to fit this new model. An Eval_1 function is introduced here. At this stage, aggregation has been ignored.

$$\begin{aligned}
\mathrm{Eval}_1 &: Desc \rightarrow Ruleset \rightarrow Resultset \\
\mathrm{Eval}_1(d, R) &\mathrel{\widehat{=}} (f \overset{m}{\rightarrow} g)R \\
&\textbf{where} \\
&\quad f : RName \rightarrow RName \\
&\quad f(n) \mathrel{\widehat{=}} n \\
&\quad g : Rule \rightarrow Result \\
&\quad g(r) = \mathrm{Eval1}_1(d)(r)
\end{aligned}$$

[3]It should be noted that the *-set* names were used to refer to these types before the model had been elaborated. The lesson is not to include *set*, *map*, etc. in type names, at least until a model is well developed.

The function for dealing with a single rule is:

$$\begin{aligned}&\mathrm{Eval1}_1 : \mathit{Desc} \rightarrow \mathit{Rule} \rightarrow \mathit{Result}\\ &\mathrm{Eval1}_1(d, r) \mathrel{\widehat{=}} r(d)\end{aligned}$$

A map functor is applied to the ruleset map to produce the resultset. This consists of a pair of functions which are applied to the domain and the range of the ruleset map. An identity function f is applied to the domain to preserve the rule names. A function g applies the curried function $\mathrm{Eval1}_1(d)$ to each rule r.

These refinements bring the model closer to an implementable solution and also make it possible to model chaining. A chain is a sequence of rulesets:

$$\mathit{Chain} = \mathit{Ruleset}^{\star}$$

And an evaluation function for a chain, ignoring aggregation, has the type

$$\mathrm{Eval}_c : \mathit{Desc} \rightarrow \mathit{Chain} \rightarrow \mathit{Resultset}$$

Assuming the rulesets in the sequence are ordered from lowest precedence to highest precedence, the following is a model for chaining by logical composition of rules:

$$\mathrm{Eval}_c(d, C) \mathrel{\widehat{=}} \mathrm{Eval}_1(d, {}^{\dagger}/C)$$

The chain C is reduced to a single ruleset using the map override operator, †. Eval_1 is applied to the resulting ruleset.

Chaining by logical composition of results is similarly defined:

$$\begin{aligned}&\mathrm{Eval}_c(d, C) \mathrel{\widehat{=}} {}^{\dagger}/(f^{\star}C))\\ &\quad\textbf{where}\\ &\qquad f : \mathit{Ruleset} \rightarrow \mathit{Resultset}\\ &\qquad f \mathrel{\widehat{=}} \mathrm{Eval}_1(d)\end{aligned}$$

In this case Eval_1 curried with the description d is applied to each ruleset in the chain C and the resultsets are reduced to a single resultset, again using the map override operator.

3.5.4 Functional Composition

The logical composition approaches presented above simply give the highest precedence ruleset priority over lower precedence rulesets. It is also possible to think of chaining by functional composition at the level of individual rule functions. Instead of one rule replacing another, the rules can themselves be functionally composed. This opens up the possibility of one ruleset acting as a filter for another.

The result from a rule in a previous ruleset must be passed explicitly to the next rule in the chain so that it can be used as another parameter of the rule function. Rule functions can only be composed if the output range from one rule function is in the input domain of the other rule function.

This form of rule composition can also be implemented in a manner similar to the approach of chaining by forward logical composition of rules mentioned above. In this case, the most general ruleset is evaluated in full first and the results are passed to the next ruleset. However, instead of simply replacing the result value with a new value, the old value is passed to the rule function where it can be factored in to the evaluation at the more specific level.

If a rule within a ruleset depends on the result of a corresponding rule from a previous ruleset in a chain, then the dependent ruleset could not appear as the first ruleset in a chain. To avoid this problem, a default value could be specified in the ruleset.

3.5.5 A Model for Functional Composition

The chaining model above requires further elaboration to support functional composition. First, an evaluation function is required which takes a *Resultset* E in addition to a description d and ruleset R:

$$\mathrm{Eval}_r : \mathit{Desc} \rightarrow \mathit{Resultset} \rightarrow \mathit{Ruleset} \rightarrow \mathit{Resultset}$$

$$\mathrm{Eval}_r(d, E, R) \mathrel{\widehat{=}} \{n \mapsto \mathrm{Eval1}_r(d, E(n), R(n)) \mid n \in \mathtt{dom}\ R\}$$

This creates a new *Resultset* (which is in fact a map) with maplets corresponding to every named rule in R.

The function for dealing with a single rule is:

$$\mathrm{Eval1}_r : \mathit{Desc} \rightarrow \mathit{Result} \rightarrow \mathit{Rule} \rightarrow \mathit{Result}$$

$$\mathrm{Eval1}_r(d, e, r) \mathrel{\widehat{=}} r(d, e)$$

The function for composition is defined recursively:

$$\begin{aligned}
\mathrm{Eval}_{rc} &: \mathit{Desc} \rightarrow \mathit{Resultset} \rightarrow \mathit{Chain} \rightarrow \mathit{Resultset} \\
\mathrm{Eval}_{rc}(d, E, C) \mathrel{\widehat{=}} &\quad C = \Lambda \ \rightarrow \ E \\
&\quad \textbf{otherwise} \ \rightarrow \ \mathrm{Eval}_{rc}(d, E \dagger \mathrm{Eval}_r(d, E, \texttt{hd}\ C), \texttt{tl}\ C)
\end{aligned}$$

If the chain C is empty, then the result is the *Resultset* E. Otherwise, an updated *Resultset* is calculated by calling Eval_r with the description d, the existing *Resultset* E, and the first element of the chain C. The previous *Resultset* is overridden with this value, and becomes the *Resultset* parameter of a recursive call to Eval_{rc}, along with the description d and the chain formed from the remaining elements of C.

3.6 Partial Evaluation

The distinction between static and dynamic evaluation becomes useful when considering caching and partial evaluation of CAs.[4] This can be modelled by splitting *Desc* into Desc_s and Desc_d, representing the static and dynamic parts of the description, and similarly for rulesets, where rules that involve only static values are considered static. The arguments are reordered to allow currying of the static parameters. A particular instance of a dynamic evaluation is defined with reference to particular static descriptions and rules.

$$\begin{aligned}
\mathrm{Eval} &: \mathit{Desc} \rightarrow \mathit{Ruleset} \rightarrow \mathit{Aggregate} \\
\mathrm{Eval}_{sd} &: (\mathit{Desc}_s \times \mathit{Desc}_d) \rightarrow (\mathit{Ruleset}_s \times \mathit{Ruleset}_d) \rightarrow \mathit{Aggregate} \\
&: \mathit{Desc}_s \rightarrow \mathit{Desc}_d \rightarrow \mathit{Ruleset}_s \rightarrow \mathit{Ruleset}_d \rightarrow \mathit{Aggregate} \\
\mathrm{Eval}'_{sd} &: \mathit{Desc}_s \rightarrow \mathit{Ruleset}_s \rightarrow (\mathit{Desc}_d \rightarrow \mathit{Ruleset}_d \rightarrow \mathit{Aggregate}) \\
\mathrm{Eval}_{d1} &: \mathit{Desc}_d \rightarrow \mathit{Ruleset}_d \rightarrow \mathit{Aggregate} \\
\mathrm{Eval}_{d1} &\mathrel{\widehat{=}} \mathrm{Eval}'_{sd}(d_{s1}, r_{s1}) \qquad d_{s1} \in \mathit{Desc}_s \quad r_{s1} \in \mathit{Ruleset}_s
\end{aligned}$$

3.7 Summary

This chapter introduced a semi-formal description of an approach to trust evaluation, including the specification of algebras for ruleset evaluation, and result aggregation and acceptance. Chaining of rulesets was described in some detail. A methodology for the design of rulesets was presented.

[4]The term *dynamic* is used here in a more general sense than in Section 3.3.2: it may include dynamic global and historical elements as well.

Clearly, there is great room in this methodology for flexibility, but also a need for clarity from the relying parties regarding the intent of a rule, what it measures, and the meaning of the result. The models presented here will, with further development, permit formal verification of evaluation software which implement them accurately. This can add to the trust in the overall system.

The model, algebras and chaining techniques can be implemented to provide trust evaluation services, and this is the subject of the Chapter 4.

Chapter 4

Trust Evaluation in Practice

4.1 Introduction

A POLICY-BASED network of trust can allow end-to-end authentication across multiple CAs, and hence the relative simplicity of the current IGTF accreditation process is appealing. However, it does not easily allow for changes to the accreditation requirements or to the CA policies.

The last chapter introduced and elaborated a model of trust evaluation for grid CAs. This model was created to provide the basis for a system for automatic trust evaluation. The potential applications of the model include partial automation of trust evaluation to provide computer-assisted assessment of CAs, and full automation of trust evaluation for software services.

This chapter documents the design and implementation of a system which embodies the model for trust evaluation. The aim of this work is to assess the feasibility and usefulness of automatic evaluation of trust in grid CAs. Some of the work in this chapter was first presented in [133].

4.1.1 Overview

This chapter has two major parts: the first part describes the potential applications (§4.2) and design requirements (§4.3) of a system for trust evaluation; the second part describes the implementation of the system (§4.4). The chapter concludes with further discussion and a summary (§4.5).

4.2 Potential Applications

THERE are a number of potential applications that emerge from the initial idea of automating trust evaluation of grid CAs. These include trust matrices, enhancements to the existing GSI and

SSL authentication mechanisms to make use of trust evaluation, and validation services based on the techniques.

4.2.1 Trust Matrices

Two forms of graphical trust matrix were introduced in [14]: the *CA feature matrix* and the *CA acceptance matrix*, as described in Section 2.4.2. The feature matrix allows an interested party to assess CAs by inspection of the published information and by comparison with other CAs in the matrix. The acceptance matrix shows the results of evaluating CAs against a policy.

The trust evaluation techniques described in the previous chapter can be applied to produce such matrices. The feature matrix will display the relevant features for a particular ruleset for a set of CAs. The acceptance matrix will display the results of CA evaluations against one or more rulesets.

4.2.2 GSI/SSL Authentication

GSI and SSL authentication typically rely on statically configured stores of trust anchors. Potentially these might be extended by using trust evaluation techniques to allow the authentication to be based on a defined policy that will use whichever CAs meet the policy at the time.

A client performing GSI/SSL verification would invoke the trust evaluation software with the certificate to be evaluated and the rulesets against which to evaluate it. This would extract the name of the CA and other details from the certificate, evaluate against the rulesets, and return a result.

4.2.3 Validation Service

Trust validation services validate credentials by checking certification paths and revocation status information. Such services reduce the need for clients to perform validation locally. A validation service which implements trust evaluation techniques could adapt to changes in trust policy and support addition or removal of CAs without the need to reconfigure clients.

Such a validation service could be queried during GSI/SSL verification to perform the necessary validation on behalf of the client. Relying parties would communicate with a service running on behalf of a PMA or VO rather than performing validations with multiple CAs. The public key of the validation service must be installed locally in a trusted store on all hosts that trust the service. From a client's point of view, trust in a small number of validation services could replace trust in a larger number of CAs.

4.2.4 Validated Credentials

A related application of trust evaluation is a service which validates credentials, as with the validation service, and then issues a token which indicates that the credentials have been validated.

A user's client would authenticate with the server, which would then evaluate trust in the CA that issued the user's certificate. The server would create and sign an attribute certificate or some other form of token specifying the rulesets for which evaluation passed. This would be returned to the client and could be added as an extension to a proxy certificate. A service which receives the proxy credential could check for the existence of a certificate extension containing such a token which has been signed by a trusted validation service.

This approach of caching the result in a proxy certificate extension is similar to the concept of OCSP *pre-validation* used by OGRO [110] and raises the issue of the lifetime of the validation result.

A validation service operated by a virtual organisation would use rulesets determined by the VO. The token could specify against which rulesets it has been successfully evaluated and the receiving party in the authentication could decide if it accepts those rulesets.

4.3 Design Requirements

The trust matrix software described in Section 2.4.2 implemented an acceptance matrix and a feature matrix. In that design the focus was on producing tables for visual inspection and the evaluation was closely tied to the display of the results. The variety of applications envisaged for the model of trust evaluation proposed in this thesis suggest that the core evaluation functionality should be an independent module. The *engine* is the component which will perform trust evaluation. It can be used to support services including the *trust matrices* and the *validation service.*

Figure 4.1 highlights the primary components of the trust evaluation architecture. Also shown are the inputs for the trust evaluation engine, and some of the proposed interfaces to the various services.

This section describes the design requirements for the engine, the trust matrices and the validation services. The requirements for file formats and their definition, programming model, and programming interface are also discussed.

4.3.1 Engine

The trust evaluation engine should be designed as an independent component with well-defined interfaces so that it can be used as a general-purpose library for trust evaluation. This library can then be used as the basis for applications such as trust matrices and trust validation services.

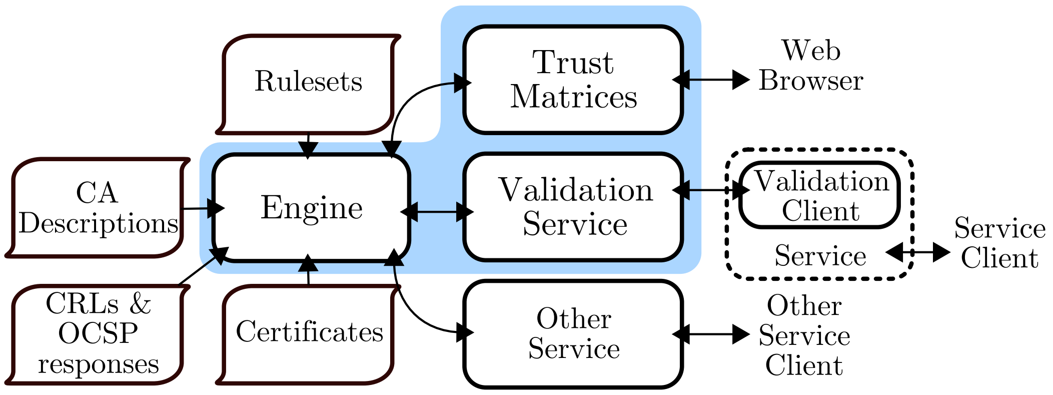


Figure 4.1: Trust Evaluation Architecture

Evaluation

The basic requirement for the engine is that it can interpret rulesets, CA descriptions and other forms of CA information, and carry out evaluations based on these by applying the rules to the CA features to generate results. That is, it must implement the evaluation model described in the previous chapter. It should support various ruleset algebras and chained ruleset evaluation. The result of evaluating a ruleset against a CA should be a map from rule names to results. This is the refined model of trust evaluation described in Section 3.5.3.

The engine should support updating or reloading rulesets and CA descriptions. One of the main motivations for this work is to support evaluations based on up-to-date relying party policies, in the form of rulesets, and CA information, from descriptions and other sources.

The engine should be able to make use of external dynamic information as well as the static information contained in the CA description. Third-party reports on CAs (by auditors, RPs, or other CAs) should be supported as sources of information.

In addition, the engine should support logging sufficient for auditing purposes. It would be desirable for the engine to be amenable to verification by formal analysis.

Performance

The trust evaluation engine will be used in a variety of applications. Some applications, such as the trust matrices, do not have strong performance requirements, while others, such as the online validation service, require that the engine will not significantly increase the response time for the service. To meet these requirements the engine should be designed to operate efficiently. However, given that the first implementation will be rather experimental, analysis of the software's performance is preferable to premature optimization.

4.3.2 Ruleset & Description Representation

The representation chosen for rulesets and CA descriptions has a strong influence on the design and implementation of the trust evaluation engine. The representation should be compatible with the models described in the previous chapter. The representation should have an associated file format to allow instances to be edited and distributed conveniently.

File Formats

As far as possible the file formats for CA descriptions and rulesets should be similar: this avoids the extra complexity of different parsers, etc. In general, the format should consist of a list or tree of keys and values. This corresponds to the models described in Section 3.5.3 of rulesets as maps of rule names to rules (which are themselves functions of descriptions to results) and descriptions as maps of feature names to feature values.

The file representations should be human-readable and writable. It would seem wise to use or extend an existing syntax such as XML [28], s-expressions (as in Lisp [115], Scheme [98], and SPKI/SDSI [147]), or even the simple format used for Java's *properties* files [164]. The format chosen should be extensible, i.e., it should be possible to insert new features that have not been previously defined without requiring the software to be rebuilt. It would be desirable to have tools to create, edit and validate these files. For creation and editing, a web-based system would be appropriate and would fit well with web-based trust matrices.

Rulesets

A ruleset must include an identifier so that it can be specified in an evaluation. There should be fields in the representation for descriptions of the ruleset and individual rules.

One desirable feature is that the rulesets could be formally analysed to some degree, in which case each rule function should be side-effect free. This has an influence on the choice of rule function language used. The ruleset format should allow a ruleset algebra to be specified.

Descriptions

A tree-like structure for the CA description language allows hierarchical organisation of the description and is compatible with the CP/CPS template of RFC 3647 [42]. The CA description language should not be tied to a restrictive pre-defined set of CA features as ultimately the set of features required depends on the evaluation rules. However, support for defining a schema for CA description features is desirable.

The CA descriptions should contain the information from the CA's CP/CPS documents in a canonical form. This requires some mapping between natural language expressions and numeric or symbolic values.

The CA description should include an identifier so that it can be specified in an evaluation. The Certification Authority's Distinguished Name (DN) is a suitable candidate as this must be unique within a PKI. If more than one CA appears with the same DN then the engine can use this information so that trust in all of the CAs bearing that DN should be substantially reduced.

Distribution

Secure distribution of rulesets and descriptions is important to the overall security of the trust evaluation system. CA descriptions and rulesets could be signed to allow for secure third-party use. Alternatively, unsigned CA descriptions and rulesets could be distributed from a trusted source. A mechanism should be provided to compare the downloaded files with the source. For example, cryptographic hashes of the files could be published along with the files themselves.

Audit Reports

It is desirable to allow information from an audit to be included in the trust evaluation system. This allows the evaluation to include the actual behaviour of the CA. This could be included as an overall audit result (e.g.'pass' or 'fail') or as a detailed audit report.

One approach is to use the CA description format as the format for an audit report: instead of the CA filling in the values for its features, the auditor gives the values that correspond to what has been observed in reality. This is a similar concept to the *audit certificates* introduced in [39].

Precedence rules could be established so that the audit report for a CA is given more weight than the CA-provided description. It is also desirable to be able to compare the audit report with the CA policy. In the description or audit report we need to identify the source: CA, auditor, or others.

This raises the issue of *namespaces* for otherwise overlapping sets of information from CA descriptions, audit reports, issued certificates and CRLs. Namespaces would allow unambiguous rules to be created.

4.3.3 Programming Model

The programming model for the trust evaluation software should fit with the overall aim of producing a system which can be formally analysed to some degree. The use of a programming language which supports a functional programming style may help in this respect.

As described above, the trust evaluation system should consist of several components. Within each component, the software should be written in a modular fashion to ease development and maintenance.

Application Programming Interface

The evaluation engine should provide a convenient application programming interface (API) for use by the trust matrices, validation service, and other applications. A functional interface will directly expose the core engine functions. An object-oriented interface can be provided for use with object-oriented systems.

Functional Interface

The programming interface for the evaluation engine must provide, at the most basic level, an Eval function, as specified in Chapter 3.

$$\text{Eval} : \mathit{Desc} \to \mathit{Ruleset} \to \mathit{Aggregate}$$

In practice the description and ruleset could be passed by reference. This requires the descriptions and rulesets to be available to the Eval function. These could be stored in some form of look-up table.

$$\text{Eval} : \mathit{DescTable} \times \mathit{RulesetTable} \times \mathit{DescName} \times \mathit{RulesetName} \to \mathit{Aggregate}$$

A similar EvalCert function is required to evaluate based on a certificate.

$$\text{EvalCert} : \mathit{DescTable} \times \mathit{RulesetTable} \times \mathit{Cert} \times \mathit{RulesetName} \to \mathit{Aggregate}$$

Additional functions could be provided that evaluate all known CAs against a single ruleset; a single CA against all known rulesets; and all known CAs against all known rulesets. The *Aggregate* type depends on the ruleset algebra of each ruleset, so these definitions are a slight abuse of the notation.

$$\begin{aligned}\text{EvalCA} &: \mathit{DescTable} \times \mathit{RulesetTable} \times \mathit{DescName} \to \mathit{RulesetName} \times \mathit{Aggregate}\\ \text{EvalRuleset} &: \mathit{DescTable} \times \mathit{RulesetTable} \times \mathit{RulesetName} \to \mathit{DescName} \times \mathit{Aggregate}\\ \text{EvalAll} &: \mathit{DescTable} \times \mathit{RulesetTable} \to \mathit{DescName} \times \mathit{RulesetName} \times \mathit{Aggregate}\end{aligned}$$

Support for chaining should allow multiple rulesets to be specified when evaluating a CA. The underlying chaining model is not exposed by this interface.

$$\text{EvalChain} : \mathit{DescTable} \times \mathit{RulesetTable} \times \mathit{DescName} \times \mathit{RulesetName}^{\star} \to \mathit{Aggregate}$$

Object-oriented Interface

An object-oriented interface for the evaluation engine should be provided for object-oriented systems wishing to use the engine. This interface can conceal data such as the tables mentioned above. The methods specified here have a parameter corresponding to the engine object itself.

$$\begin{aligned}
\text{Eval} &: \textit{Engine} \to \textit{DescName} \times \textit{RulesetName} \to \textit{Aggregate} \\
\text{EvalCert} &: \textit{Engine} \to \textit{Cert} \times \textit{Ruleset} \to \textit{Aggregate} \\
\text{EvalCA} &: \textit{Engine} \to \textit{DescName} \to \textit{RulesetName} \times \textit{Aggregate} \\
\text{EvalRuleset} &: \textit{Engine} \to \textit{RulesetName} \to \textit{DescName} \times \textit{Aggregate} \\
\text{EvalAll} &: \textit{Engine} \to \textit{DescName} \times \textit{RulesetName} \times \textit{Aggregate} \\
\text{EvalChain} &: \textit{Engine} \to \textit{DescName} \times \textit{RulesetName}^{\star} \to \textit{Aggregate}
\end{aligned}$$

4.3.4 Trust Matrices

It should be possible to display a variety of trust matrices that are useful for assessment of CAs by visual inspection. The trust matrices should be displayed via a web interface which uses standards-compliant markup for presentation. There should be a logical set of URI paths for the various matrix displays.

Feature matrices should be available for individual CAs and for multiple CAs to allow side-by-side comparison. Acceptance matrices should allow evaluation of individual CAs and multiple CAs against single, multiple or chained rulesets. It should be possible to drill down from a summary of results, via hyperlinks, to individual rule results in a particular CA evaluation.

Web Interface

The full feature matrix should display a table of all features for all CAs that are known to the service. The list of features may be determined by the features present in the CA descriptions, by the features specified in a particular ruleset, or by an external list of features. The individual CA matrix pages should show the same information for a single CA. The full feature matrix should link to the individual CA matrices, and perhaps also to the underlying description files. It should be possible to get some information about the meaning of the displayed features such as the type and range of acceptable values.

The acceptance matrices should include a summary table which displays the aggregated results for all CAs evaluated with all rulesets. A full acceptance matrix should show in detail the results of evaluation for each rule of each ruleset for all CAs. There should be matrices for the evaluation of one CA against all rulesets and one ruleset against all CAs. From the summary matrix, the results should link to detailed evaluation of a single CA against a single ruleset. These detailed evaluation tables should provide information about the rule descriptions, parameters and functions.

The matrices should make use of *web standards* so as to interoperate with a wide variety of web clients now and in the future. Figure 4.2 shows 'The Trinity of Web Standards' [179]. The trust matrices should be structured using appropriate HTML markup: the `table` element, and its child elements `thead`, `tbody`, `tr`, `th`, and `td` should be used for the matrices themselves (but not for general page layout). The visual presentation should be specified using CSS (Cascading Style Sheets). Dynamic behaviours should be controlled with JavaScript, although the basic display of the matrices should not depend on this.

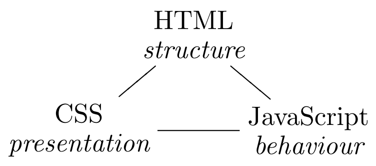


Figure 4.2: Web Standards

Adapted from [179].

URI Paths

The URI paths for the web interface are designed to be simple and human-readable. They should provide a REST-style interface to the trust matrices, i.e., the URIs identify resources on the server; the client makes requests for these resources; the server responds with a representation of the resource; and the client maintains the (minimal) state of the communication with the server [70].[1] The service should respond to GET requests shown in Figure 4.3. It may be possible to use the PUT method to support uploading new rulesets or CA descriptions.

/feature

- **/all/** – show features of all CAs; a full feature matrix
- **/ca/⟨*ca name*⟩/** – features of one CA

/acceptance

- **/all/** – evaluate all CAs against all rulesets; a full acceptance matrix
- **/all-summary/** – evaluate all CAs against all rulesets showing an overall result value for each CA/ruleset evaluation.
- **/ruleset/⟨*ruleset name*⟩/** – evaluate all CAs against a given ruleset
- **/ca/⟨*ca name*⟩/** – evaluate a CA against all rulesets
- **/ca-ruleset/⟨*ca name*⟩/⟨*ruleset name*⟩/** – evaluate a CA against a given ruleset

Figure 4.3: URI paths for Trust Matrices

[1]REST stands for **Re**presentational **S**tate **T**ransfer

4.3.5 Trust Validation Service

The trust validation service should provide remote access to a trust evaluation engine for the purpose of on-demand validation of certificates. The validation goes beyond the normal verification of the CA signature and revocation status, and evaluates the CA against the configured rulesets. A validation service may also capture dynamic information from certificates as an input for the engine.

Protocols

An online validation service should support standard protocols for communicating with clients. Possibilities include the Online Certificate Status Protocol (OCSP) [125] and the Server-based Certificate Validation Protocol (SCVP) [80] from IETF, and XML Key Information Service Specification (X-KISS) from the XML Key Management Specification (XKMS), a proposal to W3C [87].

OCSP, as described in Section 2.2, is a certificate status protocol. Using it to support trust evaluation via a validation service could be considered misuse of the protocol. However, OCSP has the advantage that it is quite widely supported in software libraries, such as the OpenSSL and Bouncy Castle [106] cryptographic libraries used by some grid middleware, and also some web browsers. This makes it an attractive standard to use.

OCSP messages support extensions, which can allow additional information (such as trust evaluation results) to be included in the message in a backwards-compatible fashion. To validate a certificate with the OCSP protocol the client sends the certificate's serial number and identifying information of the issuer. The server responds with a Good, Unknown or Revoked status. For the purposes of trust evaluation, a failed evaluation would correspond to the Unknown status, and the status for a successful evaluation would depend on subsequent validation steps, such as revocation checking. The proposed OCSP version 2 supports sending a full certificate in the status request [126]. This allows more flexibility but does increase the bandwidth requirements.

SCVP is a draft standard for a protocol for servers performing certification path construction and path validation. It supports the notion of validation policies which govern how the validation is performed, and the policy can be specified by a client in the request. This would be a suitable protocol for validation with trust evaluation but it is not as widely supported as OCSP. Preliminary support for OpenSSL has been developed [29] although it is not part of the OpenSSL library itself.

The X-KISS component of XKMS is intended to allow a client to offload processing of XML Signature `KeyInfo` elements [58]. It includes support for a Validate Service which can validate the binding between a public key and specified information. The trust evaluation engine could be used in an XKMS service to evaluate issuers. There are a number of XKMS client and server implementations available. XKMS has been proposed for use with grids and service-oriented architectures

[137] but it has not yet been widely adopted.

Although non-standard, an *ad hoc* web services protocol for trust validation may be useful for use in a web services environment. The WSDL (web services description language) description of the service can provide a basic level of documentation for integrating the service with existing systems.

Performance & Availability

Performance of an online validation service is important. If many clients are contacting the service for every authentication then a slow service will be unacceptable. Performance will partly be determined by the trust evaluation engine but also by the front-end service. It is not practical to give absolute requirements for performance for an experimental service such as this but it is certainly something that should be measured and evaluated.

Availability is another important quality for a validation service. Again, if many clients are relying on a central service for their validation needs, then any disruption to the service will effectively disable those services. Replication of the service may help. Rulesets, descriptions and other source information would need to be distributed and synchronised but the replicas can operate independently. Furthermore, if validation services are run by VOs and sites then the effects of failure are limited to the users of the local service.

4.4 Implementation

The principles described in Chapter 3, the potential applications described in Section 4.2, and the design requirements from Section 4.3 provide the basis for the implementations described in this section.

The contents of this section are presented in what is considered to be a logical order. However, it should be acknowledged that in the development of the software described there was a certain amount of interaction between the work on the different components: requirements from each part fed into the evolving design of other parts.

There is certainly room for improvement in the implementations to bring them more closely into line with the principles, application use-cases, and design requirements. These matters can be addressed in due course but do not materially affect the main points of this thesis.

This section describes the chosen representation for CA descriptions and rulesets; the implementation of the trust evaluation engine and the schema validation subsystem; and the implementation of the trust matrices and validation services.

4.4.1 CA Description & Ruleset Representation

An s-expression format was chosen to represent the CA descriptions and rulesets. This meets the requirements for extensibility and tool-support mentioned above, and fits well with the decision to use Scheme for the trust evaluation engine, described below.

The SXML (XML as S-Expressions) format [100] was used to provide a basic level of structure. This is an s-expression format that can be mapped to XML: it has the familiar tree structure of elements containing attributes and other elements. It encompasses only a small subset of all possible s-expressions. Section 4.4.3 below discusses a system for creating schemata for SXML documents.

CA Description

A CA description consists of a `ca-description` element which contains a sequence of elements that represent features of the CA. Each element has a name and may contain a value or further elements. Some elements such as `name` and `dn` are for identification purposes, while others are of interest for evaluation. The features are arranged in a hierarchy. For example, a top-level element relating to the CA's certificate may include elements for the certificate format, length of validity, and extensions.

Rulesets and CA descriptions must be semantically compatible with respect to the names, types and values of description features. The author of a CA description is expected to take care to make it compatible with the relevant rulesets. In effect, the writer of a ruleset defines what features must be described in a CA description which is evaluated with that ruleset. There is support for creating a schema for the CA description, described in Section 4.4.3 below. An example CA description schema is presented in Appendix E.

Figure 4.4 shows an example CA description with a set of features based on a ruleset for the IGTF profile for traditional X.509 CAs [67]. The example demonstrates the hierarchical structure and the types of feature value used. It is clear that this is a terse description of the CA but it captures the information necessary for evaluation with the specified ruleset.

Rulesets

A ruleset consists of a top-level `ruleset` element which contains a `name`, `description` and a sequence of `rules`. The name is used to identify the ruleset in evaluations. The description can provide more detail and, for example, is displayed by the trust matrices, described later.

```
(ca-description
  (name "CA42")
  (dn "/DC=ie/DC=grid/DC=te/DC=CA42/CN=CA42")
  (alias "CA42")
  (country "Ireland")
  (country-code "ie")
  (security-level low)
  (CA-email "ca42@te.grid.ie")
  (archive
    (available-to-external-auditors #t)
    (ca-system-logs #t)
    (cert-requests #t)
    (issued-certs #t)
    (issued-crls #t)
    (revocation-requests #t)
    (duration 3)) ;years
  (audit
    (allows-pma-audit #t)
    (internal-audits-per-year 1))
  (cert
    (basic-constraints
      (has-extension #t)
      (is-critical #t)
      (value "CA:true"))
    (key-usage
      (has-extension #t)
      (is-critical #t))
    (format
      (is-x509 #t)
      (version 3))
    (lifetime 3650)) ;days
  (cp-cps
    (has-cp-cps #t)
    (format rfc3647)
    (has-uniqiue-oid #t)
    (oid "1.3.6.1.1.1.1.42.1")
    (all-versions-available #t))
  ...)
```

Figure 4.4: Example CA Description

```
(ruleset
  (name "IGTF Traditional Ruleset")
  (description "Ruleset based on IGTF 'Profile for Traditional
               X.509 Public Key Certification Authorities
               with secured infrastructure' version 4.0")
  (rules ...))
```

Rules

Each `rule` contains a `name`, a `description`, parameters (`params`), and a rule function (`func`). For example, the following rule evaluates the minimum key size.

```
(rule
  (name "EE key size")
  (description "EE key greater than minimum size")
  (params ((ee-key-size (ee key size))))
  (func (if (>= ee-key-size 1024) 1.0 0.0)))
```

The name is used to refer to the rule in results and when performing ruleset chaining. The description can provide more information on the purpose of the rule. The `params` element contains a list of parameter declarations: the syntax is similar to a Scheme `let` expression. Each parameter declaration has two parts, a name for the parameter (in this case `ee-key-size`), and a *path* for the feature value (here `(ee key size)`). The parameter name is used in the rule function, and the path is used to find the value in the CA description. For example, the value for the parameter specified would be found in the CA description by traversing the tree to reach the `size` leaf node:

```
(ca-description ...
  (ee ...
    (key ...
      (size 2048))) ...)
```

For specifying top-level features in the description a simplified syntax is supported. Instead of writing `(name (name))`, for example, it is possible to write simply `name`. How these feature names are used to look up values is discussed in Section 4.4.2 below.

The rule function specified by the `func` element uses Scheme syntax to evaluate the parameters specified in the `params` element. The result value for the rule is calculated by the function and so can have any type supported by the Scheme language.

Aggregation

The ruleset may also contain a specification for the form of aggregation to use. In this implementation the aggregation is performed by a `fold` operation which combines results pairwise using a binary operator. For example, a ruleset could specify the following:

```
(ruleset ...
  (fold
    (null-value 1.0)
    (combiner min)) ...)
```

The engine can use the `combiner` function (`min`) and the `null-value` (more properly, the *identity* value for the operator) to aggregate the results, using `fold`:

```
(fold null-value combiner results)
```

This can be used to implement the simple fuzzy-set–based ruleset algebra described in Chapter 3. A more complex combiner function can be specified in the form of a Scheme lambda expression that takes two parameters, for example `(lambda (x y) (max 1.0 (+ x y)))`.

A ruleset for the IGTF profile for Traditional X.509 CAs [67] is presented in Appendix C.

Limitations

The above ruleset format does not allow full specification of a complex ruleset algebra within the ruleset itself (although such an algebra could be implemented externally based on the results of the ruleset evaluation with the current format). To remedy this an appropriate syntax for specifying the ruleset algebra would need to developed.

There is no explicit support for namespaces in the CA description or ruleset formats. The writer may choose to use some convention, such as a prefix, to identify elements from a particular namespace.

4.4.2 Engine Implementation

The trust evaluation engine is the central component of the trust evaluation system. The implementation presented here can evaluate CA descriptions and certificates against rulesets of the forms described above. The engine supports result aggregation and ruleset chaining.

Modules

The trust evaluation engine is divided into a number of modules. The `engine.scm` module is the core of the engine. An object-oriented interface to the engine is provided by `engine-class.scm`. Certificate-handling functionality is in `cert.scm` and some file-handling utilities are in `fileu-til.scm`.

Software Used

The Scheme language [98] was chosen to implement the trust evaluation system because of its support for a functional style of programming. This was considered important as it potentially allows the engine software to be formally verified, although this has not yet been done.[2] Scheme also has direct support for s-expressions: they are used to represent the language itself.

[2]In practice, however, flaws in the underlying operating system, libraries and compilers could undermine such verification.

There are many compilers and interpreters available for the Scheme language. The Kawa Scheme compiler [26] was chosen to implement the trust evaluation engine. Kawa compiles Scheme source code to Java Virtual Machine bytecode compatible with the Sun Microsystems Java Virtual Machine (and others). It can generate Java classes that interoperate seamlessly with other Java software. This makes Kawa attractive for writing software in Scheme that can interoperate with Java-based applications. Kawa can also produce native executables using the GNU Compiler for Java [25]. This could make it possible to package the engine in a library to be called from natively compiled applications using OpenSSL, for example.

The engine makes use of several Scheme features which are not part of the core language but have been standardised as Scheme Requests for Implementation (SRFIs). These include support for hash tables [97] and for processing of lists [155]. Kawa can make use of libraries written for Java. This provides access to a broad range of functionality, including cryptography support. The engine's `cert.scm` module uses the Bouncy Castle cryptography library and the gLite security utilities. The main `engine.scm` module is largely pure Scheme, but it does use a number of Kawa-specific features related to module loading and Java types. The use of these features would need to be removed if one wished to allow the engine to be built with other Scheme compilers.

Evaluation Core

Functions were created which implement the functional API specified in Section 4.3.3. These functions are implemented in terms of a more general function `eval-ca-list-against-ruleset-list` which takes a list of rulesets and a list of CA descriptions. The top-level functions take arguments for a table of rulesets and a table of CA descriptions and zero or more arguments for names of rulesets or CA descriptions. These arguments are used to create the lists required for the next level of processing.

The *table* arguments of the top-level functions are effectively objects which comprise a local hash-table of data and methods to add entries (from a data structure or from a named file) and to lookup entries given a key. These 'objects' are implemented as a lexical closure which contains the hash-table, the method functions, and a `dispatch` function, which dispatches the method functions based on the messages received, as shown in Figure 4.5.

To get the required lists of rulesets and CA descriptions the names can be looked up in the appropriate table, or when all entries are required the values from entire table are taken. When the lists have been created they are passed to `eval-ca-list-against-ruleset-list`. This function also takes an *evaluation function* parameter that determines how an individual CA should be evaluated against an individual ruleset. The function performs some checking to ensure that the lists and evaluation function are valid. If so, it evaluates each CA in its list against each ruleset, building up a result structure of the form described further below.

The evaluation function used for most of the API functions is `eval-ruleset-full`. This

```
(define (make-te-table)
  (let ((table (make-hash-table))) ; internal hash table
    ; Add an entry from a structure
    (define (add-entry entry)
      (hash-table-set! table (lookup-value entry 'name) entry))
    ; Get a entry by reference
    (define (entry-ref entry-name)
      (hash-table-ref table (as <String> entry-name) #f))
    (define (size)
      (hash-table-size table))
    ; Dispatch method
    (define (dispatch m)
      (cond ((eq? m 'add) add-entry)
            ((eq? m 'ref) entry-ref)
            ((eq? m 'table) table)
            ((eq? m 'size) (size))
            (else (error "Unknown method to te-table: " m))))
    dispatch))

(define t (make-te-table))
(t 'add '(item (name "a") (value 1)))
(t 'ref "a")  ; evaluates to (item (name "a") (value 1))
```

Figure 4.5: Table Object and Examples

extracts the list of rules from the supplied ruleset and passes the list and the CA description to `eval-rules-full`. This function evaluates the CA description against each rule in turn with the `eval-rule` function and returns a list of rule-name–result pairs. `eval-ruleset-full` aggregates the result and returns a result structure for that ruleset–CA evaluation.

Evaluation based on both a certificate and a CA description is similar: the certificate parameter is passed through each level, down to the `eval-rule-c` function.

Limitations

The current implementation supports a limited form of aggregation which *folds* the results together using a binary operator (e.g. $\times$, min, $+$, $\wedge$, etc.) which can be specified in the ruleset, as described in the previous section.

Rule Evaluation

The `eval-rule` function evaluates a CA description against a single rule. The function extracts the rule parameter names – corresponding to the CA features of interest – and the rule function from the rule. The feature values are looked up in the CA description. The alternative `eval-rule-c` function evaluates a certificate. The features are looked up in both the CA description and the provided certificate.

The rule function is not an executable function but merely a list data structure containing the function definition source code. The `make-lambda` function is used to create an anonymous executable function from the parameter names and the rule function. The definition of `make-lambda` is:

```
(define (make-lambda params function)
  (eval `(lambda ,params ,function)))
```

That is, the list of parameter names and the list containing function code are filled into a template for a lambda expression, and the expression is evaluated to produce an anonymous function. This function is applied to the feature values from the CA description and the result is returned by `eval-rule`. This is the very centre of the evaluation process.

Feature Lookup

The feature look-up, and most other such look-ups in s-expression tree structures in the trust evaluation system, are done with the `lookup` and `lookup-value` functions. `lookup` takes two parameters, a `tree`, which is a hierarchical list structure, and `terms` with which to search the tree. The `terms` specify a path in the hierarchy. The top element in the tree is excluded from the lookup. The lookup recursively searches for the terms in the hierarchy using Scheme's `assoc` function, which takes a term (or key) and compares it with the first element in each item in a list of pairs. For example, given a tree `(a (b (c 1) (d 2)))` and terms `(b d)`, `lookup` will return `(d 2)`. Note that `lookup`, like `assoc`, returns the matching pair, or `#f` (FALSE) if there is no match. `lookup-value` is also provided, which returns the second element of the result from `lookup`. The result may be ambiguous if the tree contains `#f` values. If `lookup` is called with a single term (rather than a list) the term is treated as if it were in a one-element list.

Look-up for certificates is handled by the `cert.scm` module. This provides a `lookup`-style interface to the attributes of a certificate using the Bouncy Castle cryptographic library. Some features can be taken directly from the certificate, such as the version, serial number, and public key. Others are processed by the `cert.scm` module. These include the certificate's validity period as a number of days; the RSA key length as a number of bits; the list of the certificate's extension OIDs (critical and non-critical); and the list of policy OIDs from the certificate's `certificatePolicies` extension.

Use of `eval`

The engine uses the Scheme `eval` procedure to convert the rule function body and parameters specified in the non-executable ruleset to a procedure at runtime. The engine obtains the ability to compile and run Scheme code at runtime 'for free.' However, this does introduce the risk that a badly written, or even malicious, ruleset could interfere with the operation of the engine. In addition, the Scheme language has a number of imperative programming features which could introduce side-effects into the rule functions. To overcome these problems a restricted `eval` could be written which only accepts a limited subset of the language.

Ruleset Chaining

All the forms of ruleset chaining by logical composition described in Section 3.5 were implemented: forward and reverse logical composition of results; and forward and reverse logical composition of rule.

Result Composition

For result composition, a top-level function was written which takes a partial result, a ruleset chain, a CA description, and a function which determines how the rule result is replaced, known as the *sequencing function.* The top-level function returns the list of rule-name–rule-result pairs.

This function is written using a common Scheme idiom for processing a list of items and accumulating the results in another list. A base case is defined for the empty list, in which case the results are returned. Otherwise if there are elements in the list the function is called recursively, with the processed first element of the list being added to the partial result and the rest of the list passed on for processing – see Figure 4.6.

```
(define (f a l)
 (cond
  ((null? l) a)
  (else
   (f (cons (g (car l)) a)
      (cdr l)))))
```

Figure 4.6: List Accumulation
f processes the elements of list *l* with function *g* and accumulates results in *a*. This example is equivalent to `(map g l)` but the pattern can support more complex accumulation.

In this case the elements of the list are rulesets in a chain. The processing of each ruleset recursively processes each rule with the sequencing function. The sequencing function for forward result composition evaluates the CA description against the rule with `eval-rule`, checks if there is already an entry for the rule in the partial result and, if so, replaces the existing result with the new result, or otherwise, adds the new result to the list of partial results. The function returns the partial result. This sequencing function is *destructive* since it modifies values in the partial result.

The sequencing function for reverse result composition first checks if there is already an entry for the rule in the partial result and, if so, does nothing and returns the partial result, or otherwise, evaluates the CA description against the rule and adds the result to the list of partial results. This function is *non-destructive* since it creates a new list based on the partial result, rather than modifying existing entries.

Wrapper functions are provided to call the top-level logical result composition function with the different sequencing functions. For the reverse composition, the order of rulesets in the chain is reversed before being passed to the generic logical result composition function. Effectively the two sequence functions embody two different orders of *precedence* for results: the forward function gives precedence to ‘new’ results while the reverse function gives precedence to ‘old’ results.

Rule Composition

With logical composition of rules, a combined ruleset is made by adding or replacing rules from the rulesets in the chain. The same pattern is used as for result composition above: a generic logical rule composition function accumulates a list of combined rules by processing each rule in each of a sequence of rulesets with a sequencing function.

The forward sequencing function checks if a specified rule already exists in the list of combined rules and, if so, destructively replaces the previous definition, or otherwise, adds to the list. The reverse sequencing function checks if a specified rule already exists in the list and, if not, adds it, or otherwise returns the list.

Again, wrapper functions are provided to call the generic rule composition function with the different sequencing functions. In this case, wrappers are also provided to call the `eval-rules-full` function with the combined ruleset.

Limitations

Functional composition (i.e., passing the result from a rule in one chain as an input to a rule in another chain) was not implemented. This would require a modified form of the `eval-rule` function.

The implementation of ruleset chaining is limited to static evaluation of CA descriptions. Evaluation of certificates is not supported. This would be simple to achieve: in the case of result composition the certificate should be passed through the chaining functions to the function which calls `eval-rule`. In the case of rule composition, the chained ruleset can simply be used by one of the evaluation functions that has a certificate parameter.

Result Format

The result of an evaluation is in the form of an SXML structure. The result contains a sequence of `ca` elements for which an evaluation has been made. Each `ca` element has a name and an element containing a sequence of rulesets. Each `ruleset` element has a name and contains a `result` element. The `result` element contains a `summary` element, which holds the aggregate result, and a `rules` element, which contains a list of individual rule results, each of which is a pair of the rule name and rule result. The same structure is used whether the evaluation is for a single CA against a single ruleset, or for multiple CAs against multiple rulesets. Figure 4.7 shows an example evaluation result.

The result format is used as the input for the acceptance matrices, described below. The engine can also generate a summarised version of the evaluation result, which omits the `rules` element in the result. This can be used by, for example, the validation services, which require only the

```
(eval-result
  (cas (ca (name "CA 1")
           (rulesets (ruleset (name "Ruleset 1")
                      (result
                       (rules (Rule1 1.0) (Rule2 1.0))
                       (summary 1.0)))
                     (ruleset (name "Ruleset 2")
                      (result
                       (rules (Rule1 0.9) (Rule2 1.0))
                       (summary 0.9)))))
       (ca (name "CA 2")
           (rulesets (ruleset (name "Ruleset 1")
                      (result
                       (rules (Rule1 1.0) (Rule2 1.0))
                       (summary 1.0)))
                     (ruleset (name "Ruleset 2")
                      (result
                       (rules (Rule1 0.9) (Rule2 1.0))
                       (summary 0.9)))))))
```

Figure 4.7: Example evaluation result

aggregate result. The engine can also return the aggregate result of a single evaluation as a plain value.

Interfaces

The evaluation engine allows direct access to its evaluation functions as well as through an object-oriented interface. The direct interface can be used from other Kawa Scheme programs, as regular functions, or from Java programs, in which case the engine is accessible through an object with a number of public static methods. The functions are as described in Section 4.3.3. The other methods used internally by the engine for handling descriptions, rulesets and results, and for other purposes, are also exposed through this interface.

An object-oriented wrapper for the trust evaluation engine was created in Kawa Scheme to provide an interface that Java-based programs can call. The engine class has member variables for the ruleset and CA description tables. The tables can be initialised from directories containing ruleset and description files.

The object-oriented interface provides methods to perform the types of evaluation specified in the evaluation API description in Section 4.3.3. It also provides methods to extract features from specified rulesets and CA descriptions, required for the feature matrices, described below.

Memoization

Some experimentation has been carried out with *memoization* of ruleset evaluation as a form of optimization. A *memoized* function caches its output for every set of input arguments to the function [2]. This requires that the function always gives the same output for a given set of inputs. One possibility for implementing such a system is to use a hash table and store the result value using the input arguments as the key. For large computations, a hash table look-up may be considerably more efficient than repeating the computation.

In a language such as Scheme, which supports lexical closures and functions as first-class objects, it is possible to define a generic `memoize` function which can be applied to any suitable function to produce a memoized version of that function. The `memoize` function called on function f will return a lexical closure containing a new anonymous function that, when called, looks up its arguments in a hash table in the closure and, if a result is found, returns the cached result or otherwise calls the function f, stores the result in the hash table, and returns the result. Outwardly, the memoized-f behaves the same as the original f.

This can provide effective optimization without introducing unnecessary complications to the program logic. Memoization has been used in the evaluation engine to cache the results of the evaluation of a particular CA description or certificate against a lists of rules in the `eval-rules-full` function. Caching at this level will not introduce consistency problems since a change to a ruleset or CA description will result in a new cache entry. However, it should be noted that there is no mechanism to automatically remove entries from the cache. The `memoize` function could be enhanced to support a method for clearing unused entires from the cache.

Native Compilation

The engine has been successfully compiled to native machine code with GCJ and it works as expected. GCJ's Compiled Native Interface (CNI) may be employed to use the natively-compiled engine from a C++ program [172, Chapter 12]. This requires some initialisation steps for Java to be added to the C++ program, and some conversion is required between Java types for strings and the equivalent C++ types, for example. A simple C++ test program was created. This program initialised the Java environment; then the Kawa environment; created and initialised a new trust evaluation engine object; performed an `eval-all` trust evaluation; and output the result. This represents a proof-of-concept. Interfacing the engine using this method to OpenSSL or similar libraries represents future work.

4.4.3 Ruleset & Description Validation

As mentioned in section 4.3.2 it is desirable to be able to validate the format of the SXML-based ruleset and description files. A system for validation was created, along with an initial set of tools for creating templates and editing files via the web. This software was written in Kawa Scheme and is essentially pure Scheme. The pregexp library [157] is used for regular expression pattern matching.

Schemata for S-Expressions

If an XML-based format had been used then an XML Schema for the two formats could have been produced and these could be validated using XML Schema tools [171, 20]. There does not appear to be a standard method for producing an *s-expression schema*, that is, an abstract description of an s-expression-based data structure. A common approach is to consider a Scheme or Lisp program which can read or validate the data structure to be the specification of the s-expression in question. Some advocates of s-expressions see this as a strength while XML supporters see it as a clear weakness of the format that precludes general use (for example, [142]).

Some opponents of XML Schema and similar ideas state that schemata limit the usefulness and extensibility of a language to the language that the designer originally imagined. Schema validation means that new elements and attributes cannot be added without invalidating the document.

Despite these reservations some XML-Schema-like validation for s-expressions, independent of the trust evaluation engine, appears to be a useful feature for a robust implementation. In addition to allowing an existing ruleset or description to be checked for well-formedness, such a schema could also be used as a template for producing new descriptions and rulesets.

The SXML format was chosen to represent descriptions and rulesets. Because SXML formats can be converted to an XML equivalent, one possibility for validating them is to convert and then validate against an XML Schema using appropriate tools. Equivalently, the files could be written as XML and validated in this manner before being converted to SXML s-expressions for use by the engine.

It was decided to implement the SXML Schema validation in Scheme to avoid such conversions between formats. The format for the SXML Schema was designed to closely mirror XML Schema but with the option of more compact syntax for some features, and items which are attributes in XML Schema descriptions may be elements in SXML Schema descriptions.

Elements can be defined to have a basic type (such as integer, string, Boolean, etc.) or a defined type. As with XML Schema, both *simple* and *complex* types can be defined. Simple types are derived from basic types and can have a restriction, such as a numeric range, a regular expression pattern or an enumeration of symbolic values. Other simple types include unions and lists of other basic or simple types. Complex types act as containers for other elements. A *sequence* is an ordered collection of elements. A minimum and maximum number of occurrences can be specified for each element in a sequence. A *choice* allows selection between different elements. The SXML Schema syntax is described in Appendix E.

Schema Validation

A schema consists of a collection of elements and types and one of the elements is considered to be the *top* element. The schema is read in by the validator and its top-level definitions are stored in look-up tables.

Validation of a document proceeds recursively depth-first from the top element. First, the element name is checked against the name of the element definition. If these do not match then validation halts at this point. Otherwise, the body of the element is compared against the type specified in the element definition. If the type is specified by name it is looked up in the type table. If found, the comparison function is called again with the expanded type definition. If the named type is not found in the type table it is assumed that it is a basic type and the element body is checked using built-in type predicates (`string?`,`integer?`, etc.).

If the type is specified using the simple or complex type definition structures then it is handled a little differently. A simple type must be a *restriction*, *list* or *union* of other simple or basic types. For restriction types the element body is compared against a set of restriction *facets*. Restriction facets include regular expression pattern matching for strings; *enumeration* of a restricted set of values; and minimum and maximum values for numbers. For simple lists every element of the list is compared against the base type and all must match. For a union type the element body is compared against the list of base types and one must match.

A complex type is either a *sequence* of elements or a *choice* between elements. For a sequence each sub-element in the content is recursively compared against the sub-element definition. The `min-occurs` and `max-occurs` attributes of elements in the sequence means that there may not be a one-to-one mapping from definition to content. The function to validate the sequence tracks the number of occurrences of content elements and element definitions, and consumes them as appropriate.

The *choice* complex type corresponds somewhat to the union simple type. The sub-element of the choice is compared recursively against all the sub-element definitions in the choice and one must match for the element to be validated.

The end-result of the validation is a Boolean value indicating success or failure. The validation can also give error messages to help track down errors.

Schema → Template

One of the aims of creating an SXML schema format was to allow the creation of template instances of the format. To do this, the schema is traversed in a similar fashion as for the validation process. For each element the name is output followed by a representation of the type or a suitable place-holder. In the case of basic types this is a simple value such as `"string"`, 1 or True. For a sequence, the inner elements are expanded and treated recursively. For a choice type just one of the inner elements is expanded. Currently this works for basic types and for sequences and choices of basic types.

Further work is required to support restrictions, unions, and lists. Other improvements could be made. If the element has a default value, this could be output. For derived simple types the name of the type might be an appropriate value. Ideally comments would be added to the output to fully document the format but this is non-trivial to achieve when generating the output as an s-expression rather than a raw string.

Schema → Form

It was felt that rulesets and descriptions could be created and edited most easily through a web interface. The ruleset and description formats are very flexible so it is important that the web interface is similarly flexible and not tied to a particular ruleset or description use. To provide this a web interface is generated dynamically from the schema. As for validation the Schema is traversed and form elements are output to correspond to elements in the schema.

Nested HTML `fieldset` elements are used to group Schema elements belonging to complex types. Appropriate input elements are provided for simple types. Enumerations have corresponding selection boxes. Client-side scripting does range checking and pattern matching of restricted types. Where repeated elements or values are allowed, client-side scripting allows extra form elements to be added or removed as necessary. For complex *choice* types and simple *union* types a selection box allows the type to be chosen and updates the input elements appropriately.

As far as possible, the HTML of the form contains only the necessary structure. The visual appearance of the form is controlled with CSS style sheets and the behaviour of the form is controlled with JavaScript scripts. Controls such as the add/remove buttons for repeated elements and the selection boxes for multiple-choice types are added on the client-side when the page is loaded. These make use of the jQuery library to provide convenient manipulation of the DOM elements of the form page [144].

Possible Improvements

Bag Types

As well as providing for sequence and choice it would be useful to provide a *bag* complex type that represents an unordered collection of elements and has some of the properties of both sequence and choice (in effect, it is a *choice* whose elements can appear an unbounded number of times).

Editing SXML Documents

The Schema → Form system described above can generate the appropriate forms for a given schema, but it does not yet have an appropriate back-end system to allow saving submitted forms as SXML documents, or to allow editing of previously saved documents.

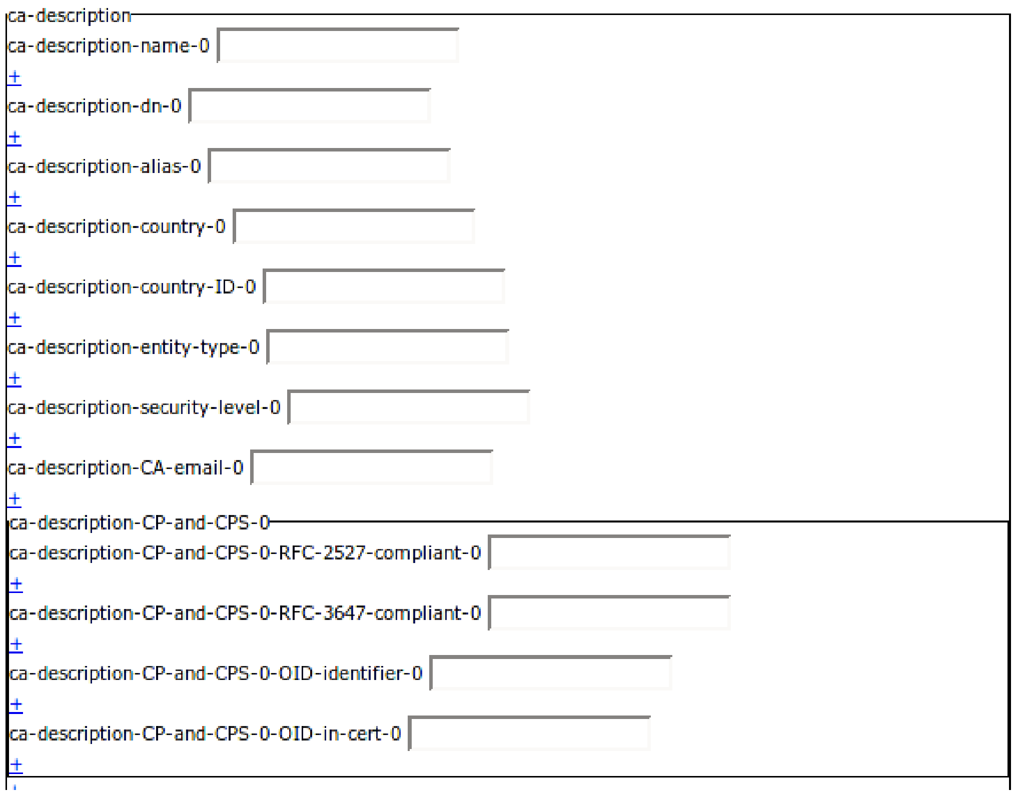

ca-description
ca-description-name-0
+
ca-description-dn-0
+
ca-description-alias-0
+
ca-description-country-0
+
ca-description-country-ID-0
+
ca-description-entity-type-0
+
ca-description-security-level-0
+
ca-description-CA-email-0
+
ca-description-CP-and-CPS-0
ca-description-CP-and-CPS-0-RFC-2527-compliant-0
+
ca-description-CP-and-CPS-0-RFC-3647-compliant-0
+
ca-description-CP-and-CPS-0-OID-identifier-0
+
ca-description-CP-and-CPS-0-OID-in-cert-0
+

Figure 4.8: CA Description Form

Ideally, to create a new SXML document the Schema → Form system would load the appropriate schema and present the form. The user could edit the form, adding and removing elements as necessary. When the user is finished the form could be submitted. The editing service would receive the submission and construct a corresponding SXML document from the returned data. The constructed document could then be validated against the schema as an additional check.

To edit an existing document the editing service would create the form from the schema but also populate the form with data from the description. This could either be done completely on the server side by populating the form structure as it is constructed, or partially on the client side, by having the browser request values for the form data.

When editing CA descriptions and rulesets the created document would need to be saved to a location accessible to the trust evaluation engine. The engine would then have to be signalled to update its static information. The editing service would have to be tightly controlled so that only authorized persons could update the files.

4.4.4 Trust Matrices

The trust matrices web application was developed to implement the requirements described in Section 4.3.4. The trust matrices are designed to provide HTML format presentations of output from the trust evaluation engine.

Modules

The trust matrices software is divided into a small number of distinct modules. The core module is `matrices.scm`, written in Kawa Scheme, which implements the output transformation and matrix formatting. The object-oriented interface is provided by `matrices-class.scm`, also written in Kawa Scheme. The acceptance matrix and feature matrix web servlets are written in Java and are called `AcceptanceServlet.java` and `FeatureServlet.java` respectively.

Software Used

The trust matrices make use of the trust evaluation engine module described above. Within the main trust matrices module the engine module is used mainly to provide functions for handling the result data structures. The `matrices` module does not directly call any of the evaluation functions. The `matrices-class` module implements a Java-compatible object-oriented interface to the trust matrices and this makes use of the trust evaluation engine's own object-oriented interface.

Like the evaluation engine, the trust matrices core module is developed in Kawa Scheme. This allows re-use of the trust evaluation engine functions and convenient handling of the output formats. As with the engine, it allows the trust matrices core to be conveniently integrated with Java software.

The web servlets for the acceptance and feature matrices are written in Java for use with the Tomcat servlet engine [11]. Tomcat is commonly used for a number of grid-related web-based services, such as VOMS Admin and R-GMA. The servlets use the object-oriented matrices interface. This provides a demonstration of the cross-language integration offered by Kawa. In fact, the servlets themselves could be written directly in Kawa, but this was not attempted.

The HTML formatting done by the trust matrices core is facilitated by the SSAX library [99]. SSAX provides a variety of tools for parsing, querying and converting XML documents using s-expressions and Scheme. In the trust matrices it is used to convert an SXML (XML as S-Expressions) representation of the matrix tables to HTML for use by the servlets [100].

A small amount of work was required to get SSAX (or to be more specific, its SXML-to-HTML converter) to work with Kawa. This was largely confined to a compiler-specific `myenv-kawa.scm` configuration module. SSAX provides such modules for several other Scheme compilers but did not have one from Kawa. This configuration module defines various useful constants, functions and macros, some of which may make use of compiler-specific functionality. This allows SSAX to have a standard interface to the various compilers. The changes required for Kawa were relatively minor. They included changes to the name of the temporary symbol generating function, the error handling function, and the standard error stream. There were also several macro definitions required, for which the implementation was taken from the configuration module for another compiler. Outside the configuration module, a number of the implementation modules for the SSAX library were altered to 'export' the relevant functions in a form recognised by Kawa. In addition, a Makefile was written to allow the SXML components to be compiled with Kawa and to produce a jar-file of the compiled bytecode.

Result Formatting

```
`(table
  (thead
   (tr (th "")
    ,(map ruleset-name->th
      (ruleset-names-of
        eval-result))))
  (tbody
   ,(map ca->tr
     (cas-of eval-result))))
```

Figure 4.9: Quasi-quoted SHTML Template

The initial ` indicates that the list is quasi-quoted, and that elements beginning with , should be evaluated.

The acceptance matrices are created from the evaluation results in two steps: transformation and SXML-to-HTML conversion. The `matrices` module contains several functions for formatting different forms of results but all share common techniques. The transformations turn the evaluation results into 'SXML HTML' or 'SHTML'.

The formatting functions use Scheme's *quasi-quoting* syntax to provide a template for the output, as shown in Figure 4.9. The template establishes the structure of the HTML and the values are filled in by functions manipulating the ruleset.

Within the templates, values from the rulesets are transformed. Individual functions perform simple transformations, such as the `(ruleset-name->th)` function which transforms a ruleset name string into an SHTML `th` (table heading) element containing the name and other formatting. Links can be included so that, for example, a CA name is linked to a page showing evaluations of that CA. The links use the URI paths described above.

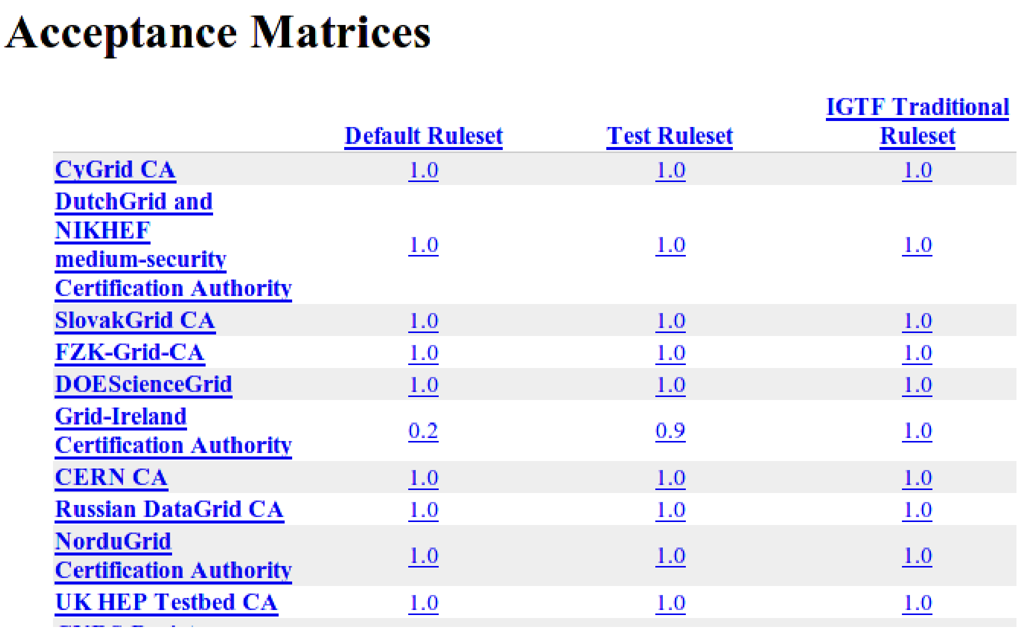
Some more complex transformations map a simple transformation over a list of values, as shown in Figure 4.9. The most complex transformations in the `matrices` module have nested calls which extract components from results, transpose rows and columns, and finally format as SHTML.

The transformation can be compared to those possible with XSLT [44]. In this case the language used to express the transformation – Scheme – is a general-purpose programming language. The input and output formats of the transformation are instances of the language's main representation format – s-expressions.

Once the transformation is complete, the SXML-to-HTML conversion is performed by the SSAX library. The `SXML->HTML` function takes an SXML structure and converts it to the equivalent HTML. The library performs some simple checking to ensure that text includes only the characters allowed in HTML. The `matrices-class` module, or another user of the `matrices` module, calls `SXML->HTML` with the output of one of the `matrices` formatting functions. The resulting HTML can then be used in the matrices web interface.

Web Interface

The web interface is provided by the `AcceptanceServlet` and `FeatureServlet` classes, which use the object-oriented interface of `matrices-class`. Instances of `matrices-class` contain a member variable for a trust evaluation `engine-class` instance. The engine class can be initialised, as described above, to load CA descriptions and rulesets. Methods are provided to create the various trust matrices each of which operates by calling the trust evaluation engine – either for evaluation or to extract the features of CAs; formatting with the `matrices` module; and then converting the result to HTML with SSAX. The servlets instantiate a `matrices-class` object to provide the matrices themselves. The servlet handles GET requests for the various forms of acceptance matrices and feature matrices. The supported requests are as described in Section 4.3.4 above.



Acceptance Matrices

	Default Ruleset	Test Ruleset	IGTF Traditional Ruleset
CyGrid CA	1.0	1.0	1.0
DutchGrid and NIKHEF medium-security Certification Authority	1.0	1.0	1.0
SlovakGrid CA	1.0	1.0	1.0
FZK-Grid-CA	1.0	1.0	1.0
DOEScienceGrid	1.0	1.0	1.0
Grid-Ireland Certification Authority	0.2	0.9	1.0
CERN CA	1.0	1.0	1.0
Russian DataGrid CA	1.0	1.0	1.0
NorduGrid Certification Authority	1.0	1.0	1.0
UK HEP Testbed CA	1.0	1.0	1.0

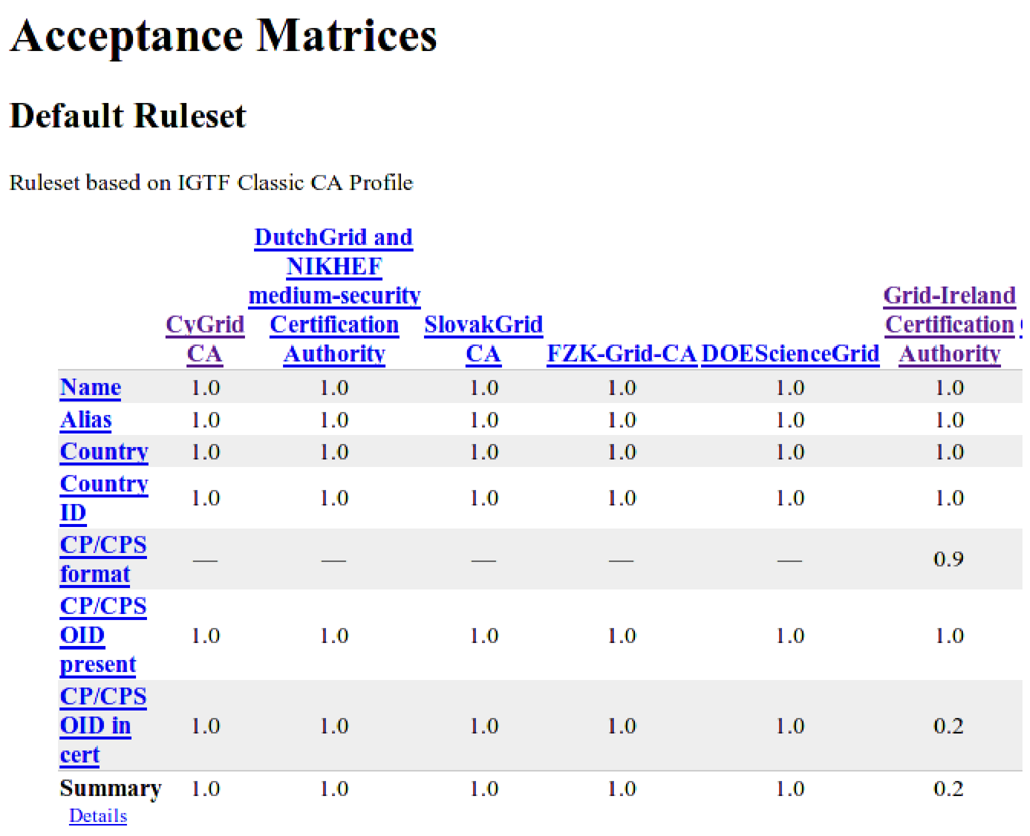
Figure 4.10: Summary Acceptance Matrix

4.4.5 Validation Service

Two prototype validation services have been implemented. The first is a simple web service using an *ad hoc* protocol. The second is an OCSP service. Both services use the trust evaluation engine to evaluate the issuer of credentials presented for validation.

Web Service

The validation web service is hosted on Tomcat and is implemented in Java using the Axis web services toolkit [10], with HTTPS authentication provided by the gLite trustmanager library. The service is implemented as a simple Java web service, meaning that the Java source file is deployed



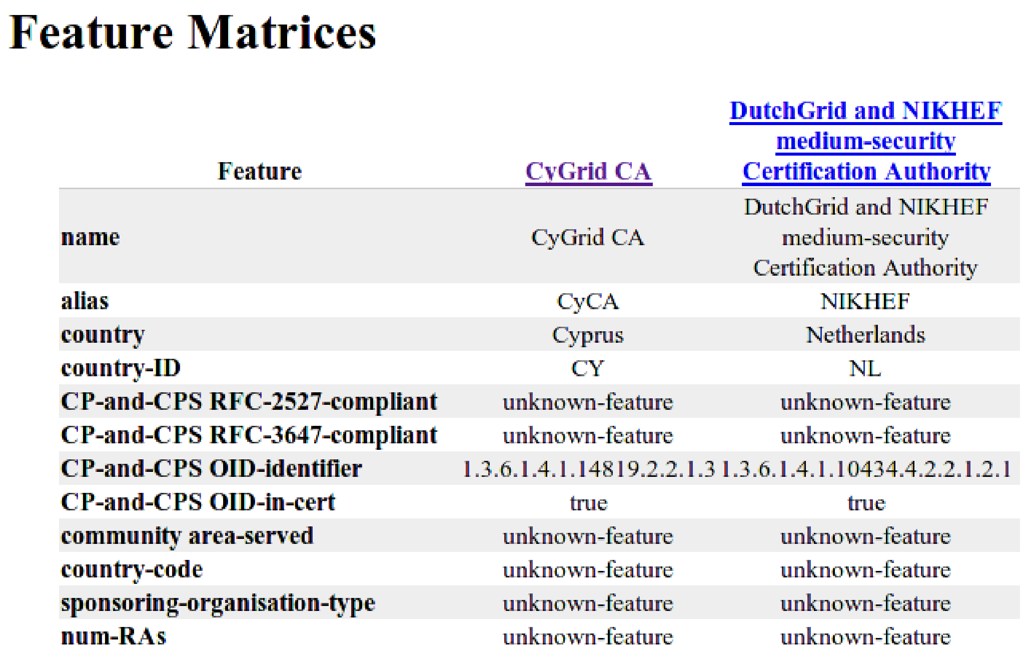
Acceptance Matrices

Default Ruleset

Ruleset based on IGTF Classic CA Profile

	CyGrid CA	DutchGrid and NIKHEF medium-security Certification Authority	SlovakGrid CA	FZK-Grid-CA	DOEScienceGrid	Grid-Ireland Certification Authority
Name	1.0	1.0	1.0	1.0	1.0	1.0
Alias	1.0	1.0	1.0	1.0	1.0	1.0
Country	1.0	1.0	1.0	1.0	1.0	1.0
Country ID	1.0	1.0	1.0	1.0	1.0	1.0
CP/CPS format	—	—	—	—	—	0.9
CP/CPS OID present	1.0	1.0	1.0	1.0	1.0	1.0
CP/CPS OID in cert	1.0	1.0	1.0	1.0	1.0	0.2
Summary Details	1.0	1.0	1.0	1.0	1.0	0.2

Figure 4.11: Acceptance Matrix for a Single Ruleset



Feature Matrices

Feature	CyGrid CA	DutchGrid and NIKHEF medium-security Certification Authority
name	CyGrid CA	DutchGrid and NIKHEF medium-security Certification Authority
alias	CyCA	NIKHEF
country	Cyprus	Netherlands
country-ID	CY	NL
CP-and-CPS RFC-2527-compliant	unknown-feature	unknown-feature
CP-and-CPS RFC-3647-compliant	unknown-feature	unknown-feature
CP-and-CPS OID-identifier	1.3.6.1.4.1.14819.2.2.1.3	1.3.6.1.4.1.10434.4.2.2.1.2.1
CP-and-CPS OID-in-cert	true	true
community area-served	unknown-feature	unknown-feature
country-code	unknown-feature	unknown-feature
sponsoring-organisation-type	unknown-feature	unknown-feature
num-RAs	unknown-feature	unknown-feature

Figure 4.12: Detailed Feature Matrix

on the Axis service, which compiles the source file; produces a WSDL description of the service based on the public interface of the class; and makes the service available. The service uses the Bouncy Castle cryptography library and the gLite security utilities.

When the service starts it instantiates a trust evaluation engine object from the `engine-class` module. The service exposes one public method, `validateCert`. This method takes a single parameter for a PEM-encoded certificate and returns a String value representing the evaluation result. The implementation of the method simply parses the certificate parameter and extracts the issuer's distinguished name. This is passed to the evaluation engine to be evaluated using a default ruleset. The result is formatted as a string and returned.

A simple client was developed for the service using the Axis client library. This makes use of the gLite trustmanager library to support secure connections to the service. The client requires a grid proxy credential in order to connect to the service. The validation client reads the certificate file to be validated into a string and makes a call to the `validateCert` method with the string value. The result string is printed to standard output.

This proof-of-concept validation service implementation does fully evaluate the description of the relevant CA but does not make use of the submitted certificate except to discover the issuer name. The service should call the evaluation engine with the certificate as a parameter. Furthermore, the service does not provide a full 'validation' of the certificate including checking the revocation status and validity period, and verification of the issuer's signature, which might be suggested by the method name. The service does not provide convenient configuration of supported rulesets and CAs: at present the source must be modified and redeployed to change these.

OCSP Service

The OCSP service is based on the Novosec OCSP software [163], written in Java, which in turn extends the Bouncy Castle cryptographic library. The OCSP service provides a backwards-compatible OCSP interface to the trust evaluation engine.

The Novosec OCSP service takes its configuration from an `OCSPServerParams` object accessed via Java remote method invocation (RMI). In the example configuration supplied with the software, the server is configured to handle requests for certificates from a single CA, and it must have access to all the valid certificates and to all the revoked certificates. The server was modified to support multiple CAs and to treat certificates from a known issuer as valid if they have not been revoked. The OCSP server configuration utility class, `RMISetter` was modified to allow multiple CAs to be configured, and to support configuration of the revocation information from CRLs rather than from non-standard files containing a collection of certificates.

The OCSP service was modified to use the trust evaluation engine to evaluate the issuers of certificates submitted for validation. The issuer name is derived from the request and this is passed to the engine, via the object-oriented interface, for evaluation. The result is handled in two ways.

It is used by the service to decide whether the result should be Good, for issuers which pass the evaluation, or Unknown, for those which fail. It is also packed into a custom extension to the OCSP response which is returned to the client.

The Novosec OCSP client was adapted to recognise the trust evaluation result extension in the response. It was also refactored to allow it to be called conveniently from within other Java (and Kawa) programs. The OpenSSL OCSP client also works with the modified Novosec OCSP service.

The service should be improved to support external configuration (via the RMI interface) of the rulesets and acceptance function used for trust evaluation. The service could also be modified to get its configuration for CAs and CRLs from the standard CA trust store location used by the LCG grid middleware.

4.5 Discussion

The discussion in this chapter and the last has focussed on trust evaluation for traditional X.509 certification authorities. The evaluation model and the engine implementation presented above should be applicable to evaluation of trust for, initially, the non-traditional IGTF profiles, which apply to short-lived credential services, for example. More generally, the evaluation engine could be used for evaluating trust in other security systems.

The validated credentials system described in Section 4.2 has not yet been implemented. If implemented, it would be interesting to compare its overall performance with that of the validation service. This would parallel the evaluations performed on pre-validation of OCSP status with OGRO [110].

The implemented trust evaluation system demonstrates the application of the principles from the previous chapter. However, for these services to be of practical use in realistic situations they must have adequate performance. In Chapter 5 an evaluation is conducted based on a number of experiments that test the performance and scalability of the trust evaluation engine and services in various configurations.

Chapter 5

Trust Evaluation Performance

5.1 Introduction

The trust evaluation system described in Chapter 4 – comprising the evaluation engine, trust matrices, and validation service – is intended as a proof-of-concept implementation of the models described in Chapter 3. As such, performance was not a major priority during the development process. However, for trust evaluation to be considered for practical use it is important to demonstrate that it can be used efficiently.

The preparations for the experiments are described below. Section 5.2 presents and evaluates the results of the experiments relating to the evaluation engine, whereas Section 5.3 covers those relating to the derived services. Section 5.4 concludes the chapter with a discussion of the results.

5.1.1 Preparations

A test-bed was created to allow the trust evaluation system to be tested in a controlled manner. Fifty test Certification Authorities were set up, which is approximately equal to the scale of the IGTF membership. Each CA has a description file customised with its details based on common practices for IGTF CAs and has issued three hundred certificates for personal and network entities. Approximately twenty certificates have been revoked for each CA. These test CAs were set up and managed with scripts based on the `CA.pl` script supplied with OpenSSL. This was modified to allow more convenient batch operations.

5.1.2 Experiment Software

A Kawa Scheme program was written to run the trust evaluation experiments. Functions were created to run a test for a given number of iterations and record the time taken; to run a test for a given amount of time and record the number of iterations; and to run a test for a given number of iterations or a given amount of time in a given number of concurrent threads.

The tests were written to generate output in a *space-separated value* text format suitable for use with the *Gnuplot* plotting software. Perl scripts were used to aggregate multiple test runs and calculate mean values.

5.1.3 Experiment Hardware

The experiments were carried out on a lightly-loaded desktop PC with 1GB of RAM and an Intel Pentium 4 processor with a clock speed of 2 GHz running a GNU/Linux operating system. The client used for OCSP network tests was a notebook PC with 768MB of RAM and a PowerPC processor with a clock speed of 1.3 GHz running Mac OS X.

5.2 Experiments on the Engine

5.2.1 Performance

These experiments test the performance of the trust evaluation engine. The engine was configured with a set of CAs and descriptions with a scale similar to the IGTF membership. Performance was measured both for sequential and concurrent requests.

1 to 100000 iterations

The first experiment was to assess how the engine would deal with an increasing number of iterations. The expected result was that the time taken would be directly proportional to the number of iterations. For this test memoization was enabled.

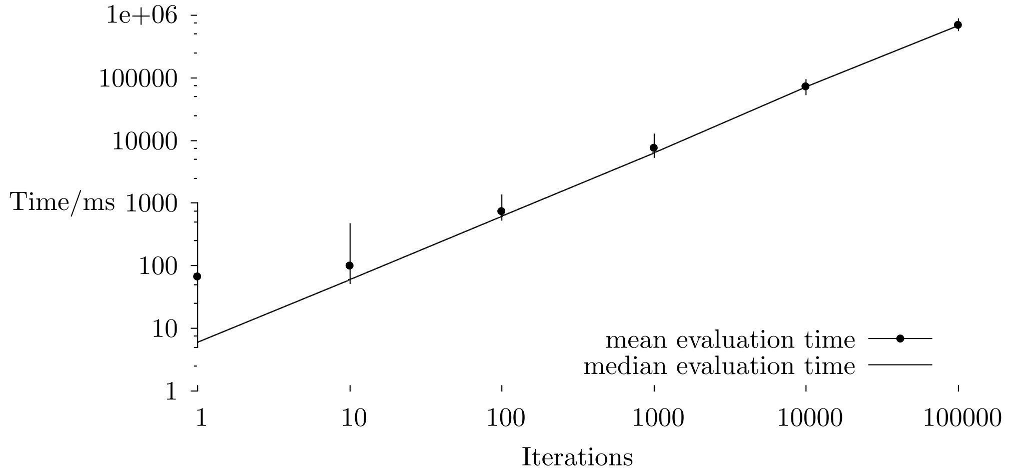


Figure 5.1: Evaluation time increases linearly with number of evaluations

Figure 5.1 shows a plot of the time taken for a logarithmically increasing number of evaluation iterations, over twenty runs. It is evident that the expectations for a linear increase in the time taken were confirmed. There is considerable variability in the values for one and ten iterations due to the large measured times for the earliest runs (1055ms for the first iteration and 488ms for the next 10 iterations). The mean time taken per iteration was 7.1 ± 0.9ms (based on the 100000-iteration measurements).

Concurrency

These concurrency tests measured how the number of concurrent threads affects the response time, i.e. the time taken to perform one thousand evaluations with between one and fifty concurrent threads. Twenty test runs were carried out. For this test memoization was enabled.

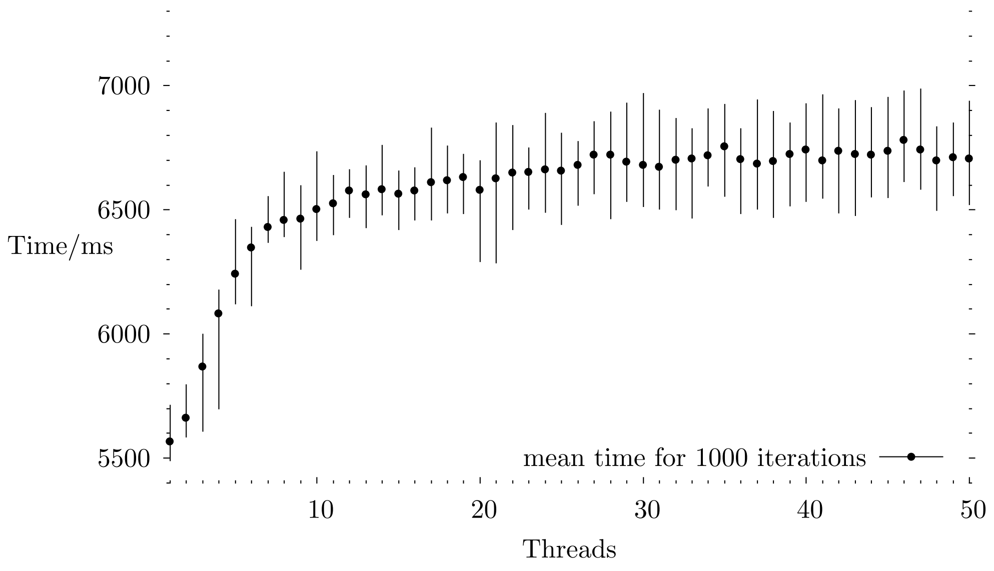


Figure 5.2: Evaluation time for 1000 iterations increases with number of threads

As shown in Figure 5.2, increasing the number of threads actually increases the time taken to perform a given number of iterations. Between 1 and 10 threads there is a relatively steep increase in time taken. Above 10 threads the increase is more gradual and the total time taken becomes more variable with increased concurrency. The time taken for 1000 iterations has a mean of 5.57 ± 0.05s with one thread and a mean of 6.71 ± 0.10s with 50 concurrent threads. This increase may be due to the overhead of the threading system. The 'levelling-off' visible above 10 concurrent threads may be due to the JVM using a maximum number of native threads and the operating system mapping multiple JVM threads onto a single kernel-level thread, thereby reducing the native threading overhead.

5.2.2 Memoization

Memoization, explained in Section 4.4.2, is a form of optimization where the results of a function are cached for each set of input values encountered. On subsequent calls to the function the cached values are returned. In the trust evaluation engine, the `eval-rules-full` function has been memoized. To measure the effect of memoization, trust evaluations were carried out with memoization on and off. A random sequence of fifty evaluations was made where each request was for one out of five possible CAs. The tests were repeated ten times with memoization and ten times without memoization, using the same random sequence. If memoization improves performance then the response time would be expected to decrease once the results have been cached.

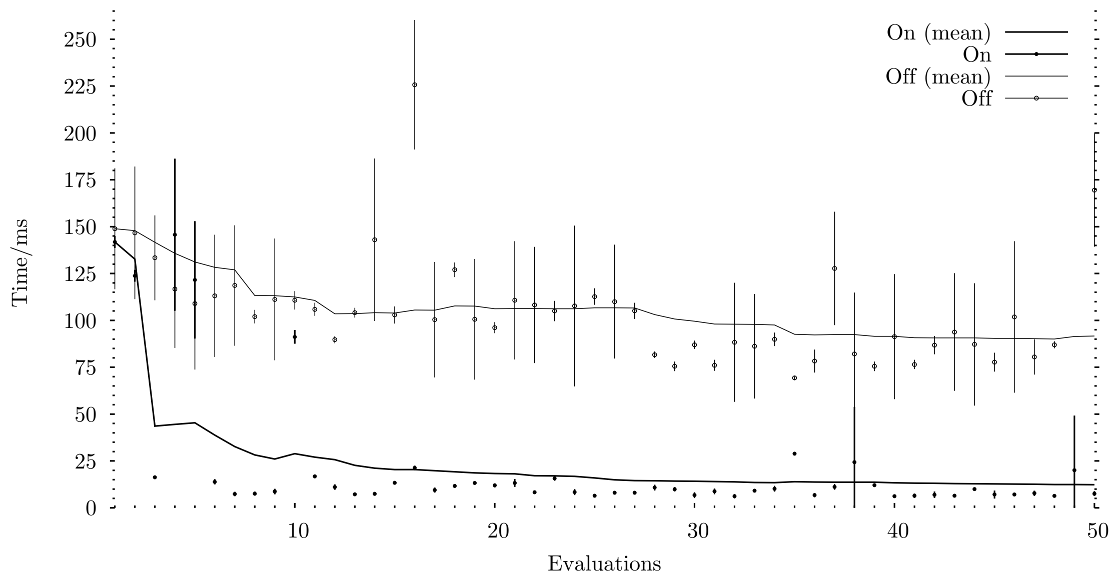


Figure 5.3: Evaluation time decreases due to memoization

Figure 5.3 shows the effect of memoization. The individual results for each scenario were averaged over the ten runs and are plotted as points (error bars show the sample standard deviation). The cumulative error-weighted means for each scenario are plotted as lines.[1] When memoization is used, the initial evaluation for each CA takes of the order of 100ms (124.8 ± 29.5ms) but thereafter drops to around 10ms in most cases (10.7 ± 7.8ms, including some outlying values[2]). Without memoization each evaluation takes of the order of 100ms (112.0 ± 61.5ms, including some outlying values[3]). Clearly memoization represents a significant optimisation in terms of computation time.

[1]The cumulative error-weighted mean for evaluation i is $\sum_{k=1}^{i} \frac{\bar{t}_k}{s_k} / \sum_{k=1}^{i} \frac{1}{s_k}$,
where $\bar{t}_i$ and s_i are respectively the sample mean time and standard deviation for evaluation i.

[2]Memoization on — $\bar{t}_{16} = 21.3 \pm 1.3$; $\bar{t}_{35} = 28.9 \pm 0.9$; $\bar{t}_{38} = 24.4 \pm 29.4$; $\bar{t}_{49} = 20.1 \pm 29.2$.

[3]Memoization off — $\bar{t}_{16} = 225.7 \pm 34.5$; $\bar{t}_{49} = 462.9 \pm 31.5$ (outside graph area); $\bar{t}_{50} = 169.5 \pm 30.2$.

5.3 Experiments on the Services

The next set of experiments were designed to test the performance of the network-based validation service and trust matrices, which are based on the trust evaluation engine. In all cases memoization was enabled.

5.3.1 Validation Service

The OCSP-based validation service was configured to support 50 CAs each with 300 certificates. Each CA has revoked approximately 20 certificates. This means that there are 15000 possible unique requests and responses. The Novosec OCSP responder on which the validation service is based uses response caching, and this was expected to have an effect on observed response time.

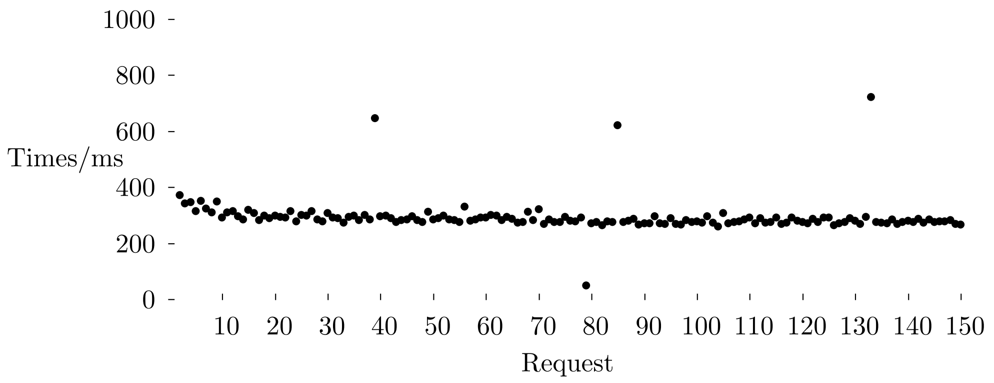


Figure 5.4: Response time for OCSP requests to a responder on the local host

Figure 5.4 shows a set of 150 requests made to the OCSP-based validation service on the local host. Most of the responses were not available in the cache and so the response time includes the time taken to evaluate trust and perform the OCSP validation.

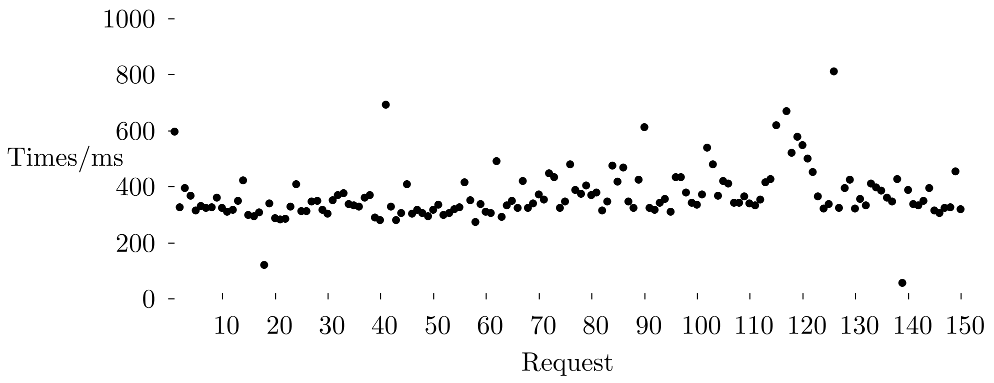


Figure 5.5: Response time for OCSP requests over a local area network

Figure 5.5 shows a set of 150 requests made to the OCSP-based validation service over a local area network (these were a different set of requests than in Figure 5.4). Again, most of the responses were not available in the cache. Note that there is considerable variation in response time introduced by the network.

To assess the performance of the OCSP validation service with a significant load, and to test the effect of caching on response time, a series of 30000 requests were made to the service after a restart. Figure 5.6 shows a 500-point moving average based on the response times of a series of 30000 requests. This plot shows that the response time decreases from around 290ms to 70ms once the cache fills with responses. A heavily used service would be expected to serve responses predominantly from its cache.

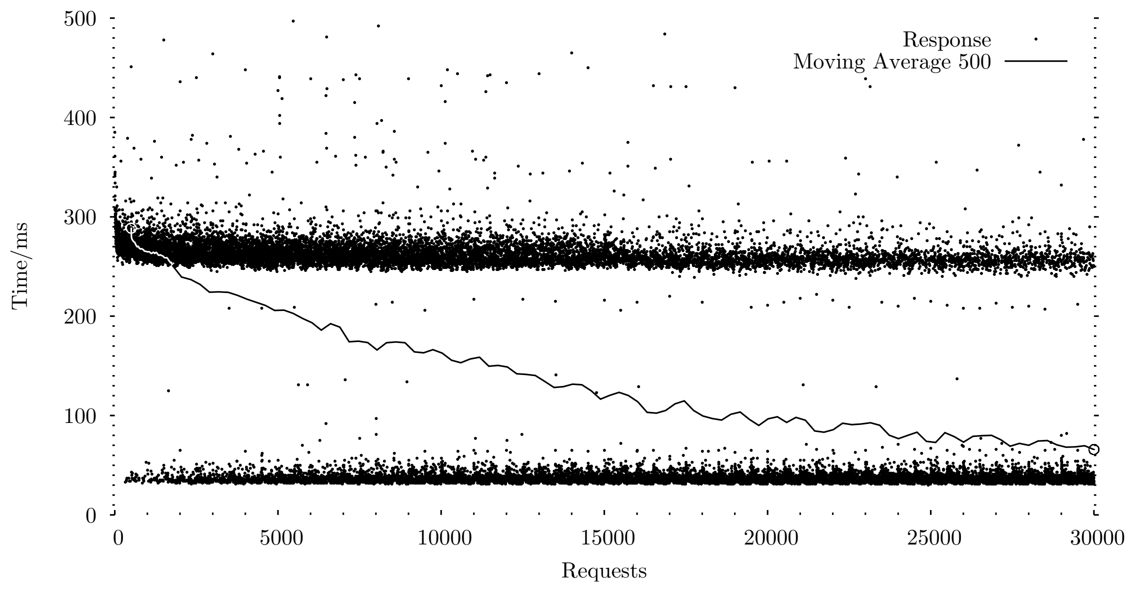


Figure 5.6: Response time for 30000 OCSP requests after service restart
The average response time decreases as responses are cached.

To give an impression of the response time pattern each request is plotted as a point. Bands are visible below 300ms, corresponding to uncached responses, and above 30ms, corresponding to cached responses. The band for uncached responses becomes less dense, and the band for cached responses becomes more dense, as the number of responses in the cache increases. There are also some higher response times visible, which may be due to variation in system load. A major outlier is not shown: the response time for the first request was 4.6s due to the initialisation that was performed on system startup.

5.3.2 Trust Matrices

The time taken to respond to a request for a feature matrix and for an acceptance matrix were measured using a command-line HTTP client. The response time includes the server processing time and the download time. The requests were made over a local area network. Figure 5.7 shows the response time of the acceptance matrices service for twenty requests after the service has been restarted. The measurements are averaged over five test runs. The response time on the first request of 3.0s decreases to 0.3s for subsequent requests. This is because the first request causes the input files to be read from disk and the trust evaluation to be performed, whereas subsequent requests read values, including cached evaluation results, directly from memory.

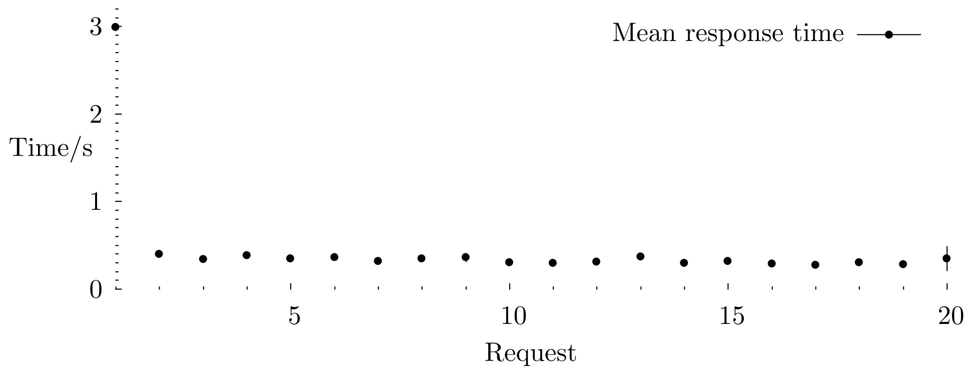


Figure 5.7: Response time of Acceptance Matrices for twenty requests after service restart

Figure 5.8 shows the response time of the feature matrices service for twenty requests after the service has been restarted. Again, the measurements are averaged over five test runs. The response time on the first request of 1.0s decreases to between 0.7s and 0.3s for subsequent requests. The cause of the variation in subsequent response times is unclear, but was present in all test runs.

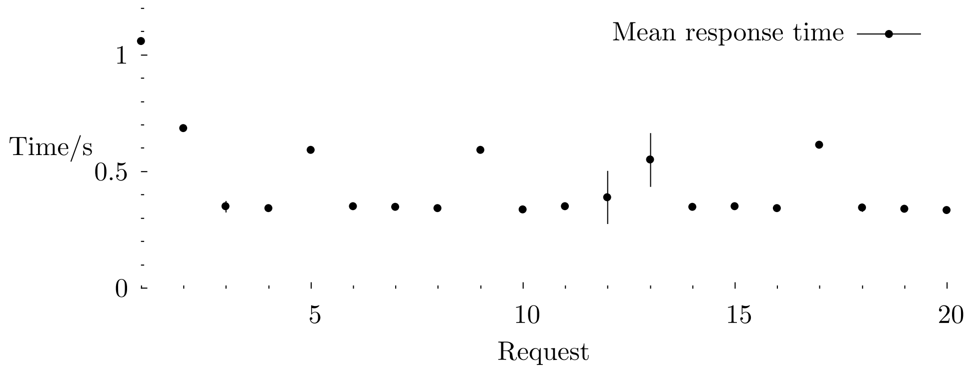


Figure 5.8: Response time of Feature Matrices for twenty requests after service restart

5.4 Discussion

For trust evaluation to be of more than academic interest it must perform adequately. The evaluation engine, with the memoization optimization enabled, has a response time of the order of 10ms, and this seems to be acceptable for many applications. The performance should, however, be assessed with reference to authentication and encryption overheads for secure communications [15].

The validation service operating over OCSP offers a response time to the client of around 300ms for non-cached responses and 70ms for cached responses. These values again seem suitable for many applications, especially if the response can be cached by the client. It would appear that the service implementation is the main factor in the response time and a more efficient OCSP responder implementation might reduce it. For the trust matrices the issue of performance is less important than for the on-demand services provided directly by the engine and remotely by the validation service. The response times of under one second are certainly acceptable for use in computer-assisted assessment of CAs.

Overall, the initial implementation of the trust evaluation model is acceptably efficient for its intended uses. This indicates that the model itself is of practical use and that the implementation presented in this thesis can form the basis for further development work.

Chapter 6

A Wider View of Security Interoperability

6.1 Introduction

A BASIS for trust is just the first step in interoperability between security domains. After trust has been established, there remains the need for interoperability between different authentication and authorisation systems used on various grids. Interoperability of grid middleware standards and implementations is currently a major research issue and security is a critical factor in any solution. The trust evaluation system described in the preceding chapters addresses the issue of interoperability between X.509 PKIs and this chapter describes further research and development in the remaining complex area of bridging security systems between incompatible grid middleware.

For example, if a user of one grid wishes to interoperate with another grid and has taken the first step of validating the trust in the PKI of the target grid using the trust validation service, then the user still has a number of other security steps to perform in order to use the target grid. These include acquiring the necessary credentials for authentication and authorisation. This chapter discusses the author's work on such a scenario.

Standardisation is desirable in many respects but the ability to harness the capabilities of specialised grids is also important. Systems that allow interoperability between existing (possibly non-standards-compliant) software may be more attractive than requiring that working software be rewritten to meet a new standard.

Within the WebCom-G project [123] a group of researchers that includes the author of this thesis have proposed such an approach to grid interoperability known as Metagrid [139]. *In this chapter the general Metagrid concepts should be attributed to all the authors of the cited papers, while the security architecture and components should be attributed to the author of this thesis.*

6.1.1 Metagrid

The Metagrid approach is to provide a bridge infrastructure which enables interoperation between different middleware and workflow components. This approach aims to achieve full $N \times N$ interoperability between a set of N supported middleware implementations through $N + M$ components: N middleware-specific interoperability components and M Metagrid services.

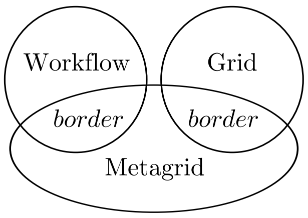


Figure 6.1: Metagrid Regions

The Metagrid architecture is based on the concept of service *regions*. Grid services exist in a *grid region*; workflow services in a *workflow region*; and Metagrid services in a *Metagrid region*. The interactions between the services occur in *border regions* where two of these service regions overlap. The non-overlapping portions of the regions are known as *internal regions* [140]. These concepts help when reasoning about security, amongst other concerns, as services in different regions have different permissions. For example, services in internal regions do not have access to Metagrid services except via the appropriate border region.

One of the aims of a Metagrid is to support access to the resources of multiple VOs on multiple grids. To this end, a Metagrid must support the authentication and authorisation mechanisms and credentials of the supported middleware. As a prerequisite for middleware to be supported in the Metagrid it must provide a basic level of security such that the confidentiality, integrity and availability of a user's work is not put at risk.

The initial targets for support are the pieces of middleware used in Grid-Ireland: the LHC Computing Grid, WebCom, and the Globus Toolkit 4, which were described in Chapter 2. Since LCG and Globus already have a lot in common in terms of security architecture this work has so far mainly focused on interoperability between these Globus-based systems and WebCom.

6.1.2 Overview

In the next section, a number of security-critical components of a Metagrid environment are discussed and a use case is presented to give an overall picture of the system. The role of the trust evaluation system in the Metagrid is mentioned.

The following two sections describe the requirements and prototype implementation of the KeyNote Credential Authority and the Delegation Service: two major components of the Metagrid security infrastructure. These are work in progress and have yet to be integrated with other Metagrid components. The chapter concludes with a discussion of future directions for Metagrid security.

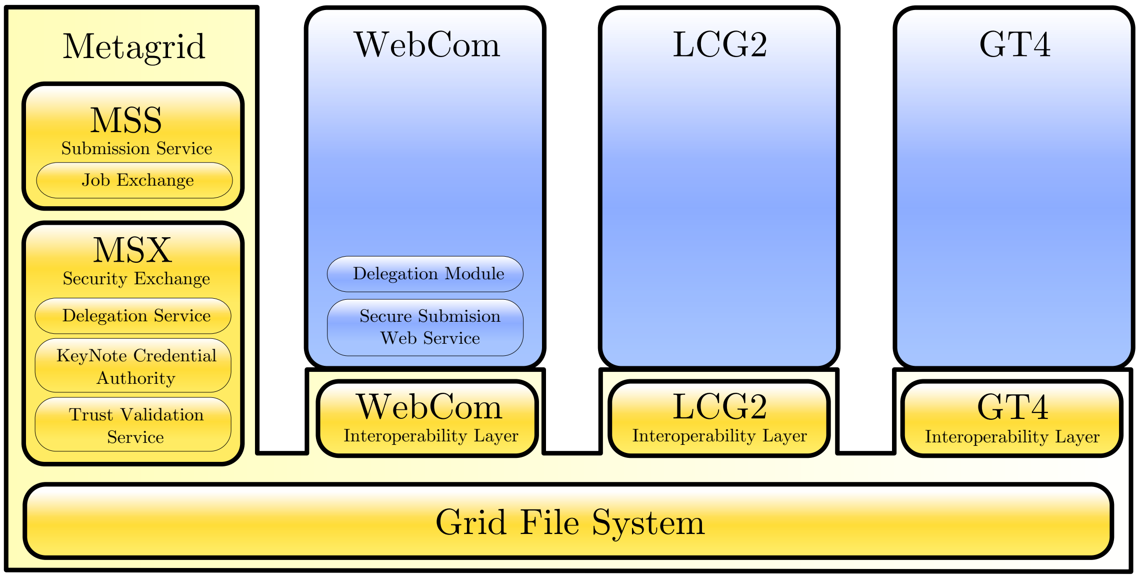


Figure 6.2: Overview

6.2 Metagrid Security

This chapter – and this thesis – focuses on authentication and authorisation but it should be noted that these are only part of any security architecture, which should also include support for accounting and auditing: these are to be handled by other Metagrid components.

The design of Metagrid uses wherever possible existing standards – *de jure* or *de facto* – for software and protocols. The primary and preferred method of authentication for the Metagrid is provided by X.509 certificates issued by suitably accredited CAs. These are widely used as part of the security infrastructure of the major European and international grid projects, as described in Chapter 2. Authorisation decisions in Metagrid services are based on successful authentication. A number of mechanisms are used for access control such as Globus GSI map-files, VOMS, and GACL [116].

Figure 6.2 gives an overview of Metagrid with an emphasis on the security components. The Metagrid *submission service* is the entry point for job submission to the Metagrid. The *security exchange* supports the use of various forms of credentials in the Metagrid.

6.2.1 Submission Service

The Metagrid Submission Service (MSS) takes requests for work from users. It provides a single consistent interface for submitting work to the various grid systems supported by the Metagrid. In addition to simple command-line tools it is expected that a graphical user interface based on the CrossGrid Migrating Desktop [104] will be provided.

Users must authenticate with the submission service. Depending on the target middleware, the user may have to delegate a proxy credential to the service so that it can act on behalf of the user and the submission service may need to contact a Metagrid Security Exchange service in order to retrieve appropriate credentials to access the resources requested by the user.

Successful authentication and authorisation with the submission service does not necessarily mean that a user's request for resources will succeed. If a user wishes to know at the time of submission whether their request will be accepted or refused this requires that the relevant authorisation information is available to the Metagrid submission service. Systems which makes use of authorisation credentials which the user obtains in advance, such as KeyNote, CAS or PERMIS, may be able to support this: if the user is not entitled to retrieve a suitable credential then they know in advance that the request will be refused.

6.2.2 Security Exchange

The Metagrid Security Exchange (MSX) provides services that allow mapping of security credentials from one middleware to those suitable for another. It is intended that the security exchange will become one of many *Social Grid Agents* and will potentially support a socio-economic model for security services [141].

The KeyNote Credential Authority and the Delegation Service, described below, are components of the security exchange that are particularly important for WebCom interoperability with any grid middleware which uses GSI-based security. Existing software such as MyProxy and CredEx can provide some of the necessary features for credential storage and exchange and these are under investigation as part of the delegation service work.

The trust evaluation engine and validation system described in the preceding chapters can facilitate interoperability between distinct PKIs and implement the different PKI policies of supported grids. It can be used by both the Metagrid services and by the border services.

6.2.3 Grid File System

The Grid File System is a distributed file system designed for use with grids [111]. It is based on the 'Filesystem in Userspace' (FUSE) module for the Linux kernel [166]; the Curl HTTP client library [161]; and the Gridsite library and server module [117] for the Apache web server, which allows file updates over HTTP and provides access control based on X.509 proxy credentials. The grid file system is designed to allow block level access to remote files, and modifications were made to the Gridsite module to allow this. Support is provided for file system discovery, virtual directory management and file system consistency.

Authentication is based around grid proxy credentials and is implemented using the Gridsite module. Authorisation makes use of Gridsite's GACL support [116]. In terms of authorisation, the grid file system should allow access to a file to be granted to single users, groups, and VOs. A user's job will access the file system with the appropriate delegated privileges.

6.2.4 VOMS Authorisation Credential Acquisition

For a WebCom workflow to submit an LCG grid job, a component is required to acquire an authorisation credential from the Virtual Organisation Membership Service (VOMS) for the appropriate VO. Ideally a user would be able to acquire all the necessary authorisation attributes from a service such as VOMS before invoking Metagrid. However, as the credential may pass through non-VOMS-aware components, the plain proxy must be delegated to Metagrid and then it can be used to acquire attributes as necessary.

This component contacts VOMS using a delegated credential on behalf of the user. The current prototype uses a WebCom operator to invoke the `voms-proxy-init` command line utility. Since it has access to a valid proxy credential for the user it does not need to receive the user's pass-phrase.

6.2.5 WebCom Enhancements

As mentioned above, each middleware implementation accessible from the Metagrid must provide a basic level of security so that a user's work is never unduly at risk. There are some users who would wish to use WebCom for volunteer computing in a totally unsecured environment. However, for WebCom to be used as a primary workflow component in the Metagrid some of its security features need to be enabled.

Users must be able to submit graphs to WebCom in a secure manner. Until very recently, the WebCom graph submission web service was an unsecured service. The most recent release has rectified this problem. Prior to this update, attempts were made – by the author – to create a secure submission method for WebCom by modifying the WebCom client script `submit.sh` to use appropriate Java security properties to connect to the WebCom submission service over SSL. The WebCom submission web service `WorkReceiverModule` was modified to use HTTPS instead of HTTP, and the WebCom invocation script `standalonewebcom.sh` was also modified to use appropriate Java security properties for SSL. The client and server were used successfully for tests of secure submission of simple graphs but since the submission service was implemented as a WebCom module based on the proprietary webMethods Glue web services engine it did not support grid proxy authentication or delegation.

The secure web service in the most recent release of WebCom will allow a user to submit a condensed graph and its parameters over HTTPS, with authentication using a personal X.509 certificate. Note that the certificate is not propagated within WebCom. The submission service has been updated to allow metadata to be passed securely with the submission and securely propagated within the WebCom network, and this will be used by the delegation service, described below.

Further work on WebCom and its submission service is being pursued by the WebCom developers as part of the WebCom-G project.

6.2.6 Use case: A Scientific Workflow

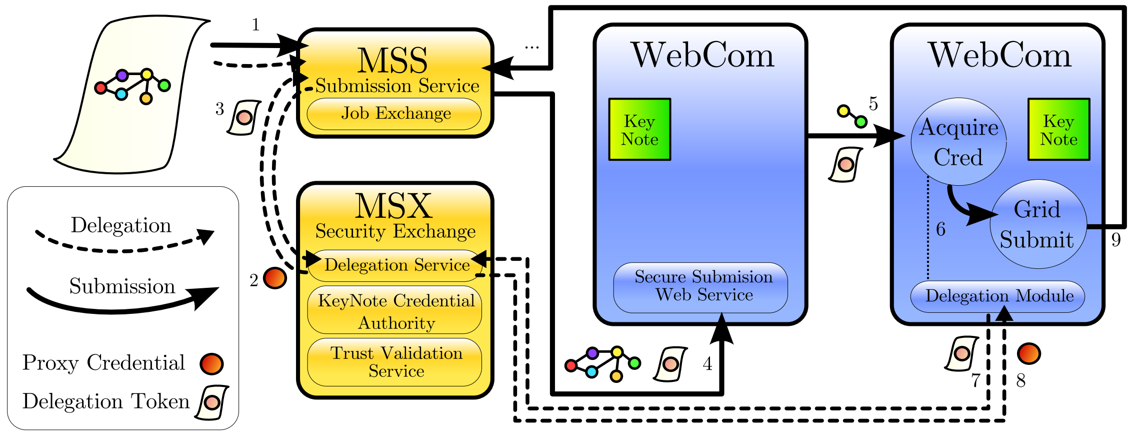


Figure 6.3: Metagrid security components involved in the workflow

The use case presented here demonstrates how the components of a Metagrid might be used in practice. The numbers in parentheses refer to Figure 6.3.

A scientist designs her experimental workflow as a Condensed Graph (see Section 2.3.3) using the WebCom integrated development environment (IDE), a graphical tool that allows creation of graphs from a palette of operations [122]. The graph contains some WebCom operations and also a number of grid jobs which will run on different grids using resources belonging to different virtual organisations. Input files are located on the Metagrid's grid file system, which appears as a network disk on her desktop computer.

The scientist first acquires credentials, then signs on using the Metagrid user interface and submits the workflow (1) and delegates a credential to the submission service. Her security credentials are delegated to the delegation service (2,3) and a delegation token is passed to WebCom along with the work (4) and securely propagated within the WebCom network. WebCom executes the graph over a distributed infrastructure (5). When a WebCom instance initiates a grid job or accesses the grid file system on behalf of the user it retrieves an appropriate credential from the delegation service via its delegation module (6). To do this, the token is passed back to the MSX (7), which validates it and delegates a grid proxy credential to the WebCom instance (8). The WebCom instance then uses this credential to submit the grid job to the Metagrid Submission Service (9). The work is passed to an appropriate border service which handles the middleware-specific details. The workflow continues execution with results or filenames from one grid job being passed to another and data being read from and written to the grid file system. Finally the results are returned through the Metagrid user interface. The output data files on the grid file system can be accessed via the network disk on the scientist's desktop.

The following sections discuss in detail the KeyNote Credential Authority and the Delegation Service. The Trust Validation Service can be considered part of the security exchange, but since it has been discussed in depth in previous chapters it is not further discussed below.

6.3 KeyNote Credential Authority

WebCom instances must have appropriate KeyNote credentials to authorise them to schedule work and return results to other instances. Each KeyNote credential must be signed by an authority that is trusted, directly or indirectly, by the party making an authorisation decision. WebCom does not specify an approach for credential creation or distribution. For the Metagrid it is proposed that there will be an online KeyNote Credential Authority (KNCA) that can be trusted by all participating WebCom instances.[1]

6.3.1 Credential Creation Authority

The KNCA builds on an earlier proposal for secure interoperability, which focused on direct EDG-to-WebCom job submission [132]. The Credential Creation Authority (CCA) was conceived as a mechanism for issuing KeyNote credentials to WebCom *users*.

KeyNote does not specify a mechanism for verifying the identity of users to whom credentials are issued. WebCom does not specify a means of distributing trusted KeyNote credentials to users. The CCA aimed to solve both these problems by issuing KeyNote credentials to users who have been authenticated using an X.509 certificate (or proxy certificate), which would be trusted by participating WebCom instances that accepted the authority of the CCA (see Figure 6.4). This would also allow grid jobs running on a user's behalf to submit work to WebCom.

The KeyNote credentials generated by the CCA would be short-lived (e.g. 12 or 24 hours) to force renewal and thereby allow changes to CRLs and the set of trusted CAs to take effect. It should have been possible to restrict further delegation of a CCA-issued credential.

Several alternatives for the authorisations to be granted by the CCA were proposed: a coarse-grained permission which simply allows full access to authenticated users is one; an attribute-based permission that tied the credential to particular roles, groups or capabilities was another. Figure 6.5 shows some of these attributes expressed in a KeyNote credential.

At the time this seemed like a promising proposal for allowing some interoperability for authorisation between European DataGrid (or Globus) and WebCom. However, support for user

[1]For wider deployment the Trust Evaluation engine could be used to support a bridge between multiple KNCAs.

```
Authorizer:  "POLICY"
licensees:  Key_CCA
Conditions:  App_Domain == "WebCom-G" && Operation == "GridSubmit"
```

Figure 6.4: Local policy trusting Credential Creation Authority to issue KeyNote credentials for a particular Applicatin Domain and Operation.

```
Authorizer:  Key_CCA
licensees:  Key_BobM
Conditions:  App_Domain == "WebCom-G" && Operation == "GridSubmit" &&
NotAfter == "2004-04-26 22:25:58" &&
VO == "ScienceGrid" && VOGroups == "Users" && VORoles == "Scientist"
Signature:  <signed by CCA>
```

Figure 6.5: Example credential issued by Credential Creation Authority authorising `BobM` to perform the `GridSubmit` operation, which has an expiration date, and contains VOMS VO, group and role attributes.

authorisation with KeyNote in WebCom did not come about as quickly as was expected and so the CCA as described was never implemented.

6.3.2 KNCA Requirements

The KNCA is similar in concept to the CCA but the main aim is to provide KeyNote credentials for WebCom *instances* rather than users. KeyNote credentials for WebCom instances can be fetched by authenticating using the appropriate host or service certificate. The KNCA-issued credentials would be short-lived for the same reasons as given for the CCA. WebCom instances therefore need to request KeyNote credentials when starting and renew them regularly thereafter.

The authorisations granted to WebCom instances should be determined by a policy stored on (or accessible from) the KNCA. Again, the authorisations could be quite broad, simply indicating that a particular instance exists on the list of approved WebCom instances, or it might grant only limited permissions to each instance. The authorisation decision could be made with the assistance of an appropriate service such as CAS, PERMIS or VOMS. One important permission that the KNCA should be able to grant to a WebCom instance is the permission to execute Metagrid or direct-to-grid operations: this permission effectively determines the WebCom instance's membership of the 'border' set, mentioned above. All the participating WebCom instances would have to include the KNCA (and its public key) as an authoriser in their local KeyNote policy.

6.3.3 KNCA Implementation

The KNCA prototypes were implemented in Java. The JKeyNote library is used for some KeyNote credential manipulation [93]. The first prototype ran as a local command line program with direct access to the signing key. This tool 'converted' an X.509 certificate in a specified file to a KeyNote credential, using a default set of permissions. In the current version the KNCA listens as an Axis web service hosted on a Tomcat server using the EGEE gLite Security components [124], which provide support for proxy authentication.

The core of the KNCA software is the `CredGenerator` class that generates a KeyNote credential based on a given X.509 certificate and supplied parameters for the KeyNote *conditions*. The public key is extracted from the supplied certificate and used as the identifying key for the credential. The software signs the generated credential with the signing key of the KNCA.

A prototype KNCA client has been developed that uses the Axis and EGEE gLite Security client libraries to connect to the KNCA service, authenticate, and retrieve a signed KeyNote credential. As indicated in Section 6.1, this is work in progress.

The client and server have been tested and work together. The generated KeyNote credentials validate successfully. However, the KNCA has not yet been tested with other Metagrid components as it has not been possible to configure an appropriate WebCom testbed.

The KNCA prototype does not yet support a server-side policy for generating credentials and does not perform any authorisation itself: currently any authenticated connecting client will be issued a KeyNote credential with a standard set of permissions. A simple list of allowed hosts is the planned first step in this direction.

It is still uncertain, but it appears that WebCom releases in the near future may support user authorisation and so the original CCA concepts may be revived. With this in mind, an authorisation scheme implementing a form of attribute- or role-based access control based on VOMS seems like an obvious choice for this purpose.

6.4 Delegation Service

Delegation of authority allows a service to operate on behalf of a user. Grid systems based on Globus use proxy credentials to delegate authority. Some middleware, such as WebCom, does not natively support proxy credential delegation, nor even the concept of user identity.

In the case of WebCom, it would be insecure to pass proxy credentials (including a private key) as WebCom graph parameters since they might be routed through untrusted WebCom instances or transmitted in clear-text. Ideally the credentials would be delegated – in the GSI sense – directly from the user's submission client to the WebCom instance that is accessing Metagrid services, but in general this is not possible as the client does not necessarily know from where this access will be made.

Instead an additional service must be provided to allow middleware such as WebCom to interoperate with GSI-based systems. The following process is proposed for submission to WebCom in conjunction with the delegation service:

1. the submission client contacts the delegation service and delegates a proxy credential to it;

2. the delegation service then generates a token which is signed and returned to the client, which submits the token along with the workflow graph to the WebCom instance;
3. the token is passed as metadata between WebCom instances executing the graph and it can later be used by a WebCom instance to request delegation from the delegation service if that WebCom instance is authorised to do so.

The use of the delegation service is dependent on a number of factors: that all communications between WebCom instances take place over SSL/TLS and are authenticated with valid CA-issued certificates; and that if any insecure (i.e. unauthenticated, non-SSL) WebCom instances are connected then secure WebCom instances must not accept work from them. These requirements aim to ensure that sensitive information, including credentials, remains confidential and that unauthenticated access to Metagrid resources is blocked.

This delegation service as proposed here is probably as secure as other similar systems which allow a third party to retrieve a user's credential. For example, for the LCG credential renewal mechanism to retrieve a credential from a MyProxy service it makes an authenticated connection from an authorised host and already be in possession of a valid proxy credential [103]. A preliminary security analysis of the delegation service has been started but more work is required to provide a clear view of its strengths and weaknesses.

6.4.1 Delegation Token

The delegation token contains a random, unguessable value generated by the delegation service: it is a form of ticket or magic cookie. It may also contain identifying information of the originating user and contact information (a URI) for the delegation service. It should be signed by the delegation service.

In a WebCom context, the token will be accessible to all WebCom instances involved in the execution of the submitted graph but they will not be able to request delegation unless they are in the delegation service's list of authorised nodes. Other hosts will not have access to the token because they are not taking part in the workflow execution, and they will not be able to guess or forge the package because of the random value and the signature.

Identifying Information

The aim of including identifying information for the user information is to confirm to the service that a request for delegation is being made with a particular credential in mind and not as part of a process of random guessing. The Subject Distinguished Name (DN) of the certificate (or a text representation thereof) could be used. Additionally, the Issuer DN (i.e. the identifier of the CA which issued the user's certificate) could be included. It would also be possible to include the certificate's serial number or even the entire certificate object in the token. However, including identifying information might allow an attacker to target the credentials of a particular user and actually reduces the entropy of the token as a whole.

Unguessable Value

The unguessable value could be purely random (subject to the limits of the underlying pseudo-random number generator) or it could be generated by encrypting a relevant value, such as the Subject DN, with the delegated private key. In the latter case, it can be decrypted by the service to verify that it matches the identity for which delegation is being requested. However, this does not appear to be more secure than simply looking up a randomly-generated value in a table. With either type of value, if it is altered (accidentally or deliberately) the lookup will fail and no proxy credential will be returned. Similarly, if extra information is included with the token such as a credential identifier, then if this does not match the unguessable value the request will fail. If either type of token is stolen and presented to the delegation service the request will succeed if and only if it is made from an authorised host.

The unguessable value is not a nonce. In a cryptographic protocol a true nonce is used only once: its purpose is purely to avoid replay attacks. In this proposal, the random value might be reused multiple times in a single WebCom graph, or even across multiple graphs submitted by the same user. However, the intention is that the token should have a relatively short validity period relative to the proxy credential it protects. It would be desirable to provide a mechanism for the token to be explicitly invalidated when it is no longer required, for example, at the completion of a job.

Symmetric-Key Encrypted Token

In order to protect the token value from being stolen, the token could be encrypted with a symmetric key shared between the delegation service and the delegation modules on authorised WebCom instances. The token can be decrypted and sent to the delegation service when making a request. The delegation service should not accept an encrypted token since it could have been presented by a client that does not have access to the shared key. If the token includes identifying information, encryption has the benefit that an entity intercepting the token will not be aware of what identity it corresponds to (i.e. who is the originating user).

There then arises the issue of how these shared keys are to be distributed. One possibility is for the shared key to be installed by a system administrator when setting up each WebCom instance. This makes replacement of the key (e.g. in the case where the key is exposed) rather complex. An alternative is for the authorised WebCom instances to authenticate with the delegation service and request the current shared key. This provides convenient pull-model distribution of the shared key and allows for periodic replacement of the key. However, this, in effect, brings us back to the situation we were in before shared keys were introduced: if any authorised WebCom instance can request the shared key, then any of these instances can make a delegation request with a token, stolen or otherwise. If the authentication and authorisation system is trusted to support secure distribution of symmetric keys then it should be trusted to support the delegation request process using unencrypted tokens.

Public-Key Encrypted Token

The above discussion of the token assumes that it is not feasible to know in advance on which border services it will be used. If the border services and their public keys are known to the delegation service then the token could be encrypted with each of their public keys (or equivalently, the token could be encrypted with a symmetric key that is in turn encrypted by each public key). This means that the token will only be readable by the specified border services. In such a system it is important to limit the delay between the time the token is issued and the time it is used, since the list of valid border services may change. The border service information could be retrieved from a secure information service or directly from an access control list on the delegation service.

6.4.2 Delegation Web Service

The delegation web-service should run over HTTPS (i.e. HTTP secured with SSL/TLS). The server should support GSI proxy authentication in addition to standard authentication with X.509 certificates.

A user's client software (will contact the delegation service, authenticating with the user's certificate or proxy certificate, to delegate a credential to the service. The service will send a delegation token in return.

To make a request for delegation, the requester (typically the delegation module, described below) contacts the delegation service, authenticating with the appropriate host/service certificate, and sends the token. The delegation service checks the requester's authorisation, validates the token and delegates an appropriate credential to the requester.

6.4.3 Delegation Module

WebCom supports plug-in modules which extend its functionality. A delegation module, which acts as a client of the delegation service, is described here. The delegation module will use the delegation token, including the unguessable value, to request a credential from the delegation service.

The delegation module is installed and configured on WebCom border instances that need to get access to delegated credentials. In addition to having the module installed, the instance must be authorised to make delegation requests at the delegation service.

When the WebCom engine queues a node for execution the engineQueueNode event is fired. The delegation module listens for this event and checks the event to determine what operator caused it. For certain operators the module makes a delegation request as outlined above and is delegated a proxy credential.

The detected operator should not need to do anything explicitly to get the delegated proxy. The request to the delegation service must be mutually authenticated using the credentials of the WebCom instance and those of the Delegation Service, and must contain information from the delegation token. The operator should not have access to either of these.

Parameter Replacement

One possible arrangement is for the module to detect operators which require the path to a proxy credential as a parameter. For example, there might `GridJobSubmitOp`. In the submitted graph, the proxy path parameter value could be set to a special value (e.g., `@DELEGATED_PROXY@`, or perhaps some unique identifier) which is recognised by the delegation module event listener. When it detects this operator with the special operand value it could make the delegation request, retrieve the proxy in a temporary file, and set the operand value to the temporary proxy file's path.

Delegation Method

An alternative would be for the operator class to have a public method which the delegation module could call to set the proxy path. WebCom's standard Operator class could be subclassed to create a DelegationOperator class and then all operators that need this feature could subclass the specialised class. This has the benefit of not making changes to operand values in the graph.

Single Delegation Operator

It might be desirable to create a single special operator that is the only one to interact with the delegation module, say, `ProxyDelegationOp`. The delegation module would look out for this operator and pass it the proxy path using one of the above methods. It would output this path as a result that could be passed to whatever other nodes require it. This operator might have operands that control the operation.

Modified WebCom Engine

Another alternative, proposed by the WebCom developers, would be to subclass the WebCom engine itself to include support for making delegation requests in a similar fashion to the above. This has the advantage that graph operators or operands would not be modified between submission and execution and so would be somewhat more consistent with the Condensed Graphs model of computation.

The single delegation operator using the object's delegation method appears preferable for its relative simplicity to implement and configure.

6.4.4 MyProxy

Instead of delegating a proxy credential to the delegation service, the user could use the existing credential delegation infrastructure provided by MyProxy servers. In this case, the Metagrid submission client would delegate a credential to the MyProxy server using a unique username and a strong automatically-generated password. The delegation should be made with rather strict settings for the credential lifetime and the authorised retrievers. The client passes the username and password during the initial WebCom submission as metadata, which is securely propagated to border nodes. When a WebCom border instance wishes to retrieve a credential it can use the username and password to contact the MyProxy service, which requires that the MyProxy service is configured to allow delegation from that host, as for the delegation service.

This approach has the advantage that an existing piece of infrastructure is used to support delegation rather than duplicating the functionality in a new delegation service. If a MyProxy username and password is stolen it may be usable from any host authorised to retrieve credentials from the affected MyProxy server, perhaps including interactive hosts such as grid user interface systems. This risk is minimized by setting a restricted list of authorised retrievers when the credential is first delegated. The MSX could provide such a list of authorised WebCom hosts on request, although if the list is set when the credential is first delegated this precludes the use of the credential by authorised WebCom instances added to the network after this time. Using MyProxy also allows for some support for renewing credentials which would not be possible in the case of the delegation service outlined above.

A hybrid approach is possible. Instead of passing the MyProxy username and password directly to the WebCom instance the client contacts the delegation service and sends the MyProxy username and password. The delegation service returns a delegation token. When a WebCom instance wishes to retrieve a credential it must first retrieve the MyProxy details from the delegation service, subject to successful authorisation, and then use the username and password to contact the MyProxy service. This adds another layer to the process but it does not necessarily make the system more secure since the token must be at least as secure as the generated MyProxy password.

6.4.5 Implementation

As for the KNCA, the Delegation Service is implemented in Java as an Axis web service hosted on Tomcat and using the EGEE gLite Security components.

To delegate a proxy credential to the service the client prototype authenticates with the service and transfers the proxy credential to the service as a parameter. The service stores the proxy credential, computes a random value for the token, records the pairing between these values and returns the token to the client. To retrieve a proxy credential the client authenticates with the service using the host certificate. The client passes the token to the service and the service checks if there is a match. The service returns the stored proxy credential as a return value. The prototype does not yet support a true proxy delegation protocol.

The delegation module which has been implemented can make delegation requests to the delegation service. In the prototype, it listens for the `engineQueueNode` event and will request delegation when it encounters an `EnterOp` operator. The request is made with a fixed token value for testing. The retrieved proxy is stored in a temporary file and the path is logged by the module. The permissions on the temporary file are set from Java with a call to the `chmod` command since Java does not provide functionality to set arbitrary file permissions. The proxy path is not passed to the operator in the prototype.

The service, client and module prototypes have been tested in isolation but have not yet been tested in conjunction with WebCom as the necessary agreed security features have only become available in the most recent release. Again, as indicated in Section 6.1, this is work in progress.

The Delegation Service and client must be improved to support a secure delegation protocol, as described in Section 2.3.2, for both uploading and downloading the proxy credential rather than the current mechanism of copying the credential over a TLS connection. In addition, requests for delegation should be authorised based on a list of valid hosts.

A possible direction for future work on the delegation service is to implement it as an exchange mechanism within the CredEx framework [55]. This would have the benefit of using the WS-Trust standards credential mapping. CredEx already supports X.509 and username/password authentication and the signed token format supported by the delegation service could be added.

6.5 Discussion

This chapter has analysed a real security architecture use case to highlight a sample of security issues (beyond establishing the roots of trust) that arise when considering interoperability between a number of grid middleware implementations. The Metagrid security architecture has been introduced, and prototypes of some components have been implemented and tested. In particular the delegation service and the KeyNote credential authority are key components for WebCom-to-grid interoperability. Arising out of joint discussion between the WebCom development team and the author, WebCom has recently been extended with several security features that make such interoperability practical. A secure submission service and support for handling graph metadata will allow the use of the delegation service and module as described here.

This is work in progress. Within the WebCom-G project, the security architecture will be further developed and the delegation service itself will be improved from a prototype to a production service, possibly based on CredEx. When this is complete, it will be possible to create a first secure Metagrid testbed. Successful testing will allow WebCom and the Metagrid services to be deployed securely on the Grid-Ireland infrastructure.

Chapter 7

Discussion

THIS thesis has introduced models and a proof-of-concept implementation for evaluation of trust in a certification authority's policies and practices against a set of rules based on the requirements of a relying party. In addition, the wider issue of security interoperability has been explored. This concluding chapter discusses the contributions made by the work presented in this thesis and the possibilities for future research which emerge from it.

The first major contribution of this thesis is the model of trust evaluation presented in Chapter 3. The model is based on the process of accreditation by a policy management authority, in which each CA is evaluated against a policy. The model has been useful for describing this trust evaluation approach. The trust evaluation model has a semi-formal mathematical basis, where the *VDM*♣ notation is used to describe the model. However, the associated formal methods have not been employed. It would be worthwhile to formally investigate and develop the evaluation model using these methods. This would entail adding the necessary preconditions and invariants to function definitions; refining the model to bring it closer to an implementable definition; and proving the correctness of the implementation.

Tools are available for working with *VDM*♣ expressions in the Haskell functional programming language that would make it possible to express the model directly in an executable form. This would both allow formal verification and provide a second implementation. A verified implementation of the trust evaluation model would provide a greater basis for trust in the system as a whole.

Part of the contribution made in relation to the trust evaluation model is the idea of ruleset algebras and the definition of several example algebras. The fact that the model is not specific to a particular algebra allows a great deal of flexibility. The existing algebras should themselves be formally verified: the algebra models could be refined from abstract mathematical models to implementable definitions. Comparison between the results of ruleset algebras should be carried out. It is clear that if a CA is evaluated against the same policy using several different rulesets then the results should, in some sense, show the same outcome.

There are a number of possibilities for further ruleset algebra developments. An interesting one to consider is a ruleset that returns function values. This would allow for parameterisation of the result. For example, each rule could return a result function which takes a single parameter representing an acceptance threshold. The function would compare the acceptance threshold against the result value it holds internally and return a Boolean result. The field of computational trust, introduced in Section 2.5, can provide possible approaches to calculating trust metrics. In addition, work on the concept of reputation may provide some useful directions for evaluating trust based on the observed behaviour of a CA.

A related contribution is in the description of a methodology for developing the rulesets and CA descriptions used with the model. This methodology has been used to develop a number of example rulesets and CA descriptions. It should be applied further to a larger number of policies and CAs in order to discover its limitations.

The methodology for CA descriptions does not fully deal with dynamic sources of information. Several types of dynamic CA features are described in Chapter 3, including *historical* and *global* features, which cannot be directly determined from a CA's policies. Historical features of a CA are those which can be measured over time. These include such properties as the actual frequency of CRL issuance, and re-issuance of certificates with the same serial number. The common factor is that these features cannot be determined directly from a single piece of dynamic information – a CRL or a certificate – but only from a record that has been accumulated over a period of time.

The model would need to be extended to deal with this form of dynamic input. It should support adding new pieces of information to a database and subsequently retrieving values, which may be aggregates for a particular class of data. The engine would need to be enhanced to implement this new model, which would require additional data storage and a means to retrieve useful feature values.

Global features are those which can only be assessed over a set of CAs. For example, the uniqueness of a CA's distinguished name or issuing namespace. The evaluation model assumes that each CA is evaluated independently but to support global features it would be necessary to introduce into the model the concept of a collection of CAs, as a map from CA name to CA description. The CA table in the current engine implementation would correspond to this definition. The evaluation functions, and indeed individual rules, would need to operate on the whole collection of CAs rather than just a single CA description.

The second major contribution of this thesis is the implementation of the trust evaluation software as a proof-of-concept of the trust evaluation model. The engine implements most of the features of the model and is written in a fashion that should enable future enhancements. The implementation has been successful as a prototype and as a test platform for experimentation with various aspects of the trust evaluation model, such as ruleset chaining.

The formal analysis of the existing trust evaluation implementation is worth pursuing. The software was written largely in a functional programming style in a version of the Scheme language. Any such analysis would need to take account of the imperative elements, the interfaces with Java libraries, and so on. Furthermore, it is common in high-integrity systems to increase trust by cross-checking results from execution of the same algorithm by multiple independent implementations. The existing implementation could operate in parallel with another implementation developed using formal methods, such as the Haskell approach suggested above.

The trust evaluation implementation performs acceptably well for its intended uses but there is still scope for optimization. The memoization technique provides a performance increase through caching. Some further experimentation could investigate the most suitable caching strategies for this kind of application. Speculative evaluations might be beneficial for heavily-used systems.

An interesting possibility is the use of partial evaluation. This is an approach to program optimization where a program $p(s, d)$ with *static* and *dynamic* inputs (s and d respectively) is evaluated with the respect to static input s to produce a residual program $p_s(d)$, which can be applied to the dynamic input d to give the same result as applying the original program to s and d. The aim is to optimize efficiency of the program. Typically, the static inputs are constant or constant-derived expressions in a program's source code. Some partial evaluation techniques require the source code to be annotated to indicate which expressions are considered static and which dynamic. Static inputs can also be more generally a set of inputs that would normally be supplied at runtime. For example, a general-purpose text searching program could be transformed into a specialised program that searches only for a specific regular expression.

The concepts of static and dynamic inputs in partial evaluation of programs can be carried over to trust evaluation. As stated in Section 3.3.2, the slowly-changing CA reports are considered static and information from individual certificates, CRLs, and so on are considered dynamic. A ruleset chain could be partially evaluated with respect to a given CA report and the residual ruleset chain could then be used to perform (presumably more efficient) trust evaluations using only dynamic inputs, for example, during peer authentication.

A related possibility is to specialise the evaluation program with respect to a particular ruleset or ruleset chain. In this case the partial evaluation would take the form of extracting and compiling the rule functions. The specialised evaluation could then be applied directly to CA descriptions.

An important factor for trusting the trust evaluation system itself is missing. A detailed security analysis of the software and the system as a whole should be performed, including attack trees and abuse case modeling, to uncover weaknesses in the design and the implementation. Formal development of the software directly from the model may help in some respects but issues related to how rulesets, CA descriptions and other inputs to the system are handled must still be considered.

The SXML Schema devised for validating SXML documents, such as the CA descriptions, is a useful contribution in the field of document processing in Scheme. XML Schema is heavily used to specify XML-based languages both as a form of documentation and for automatic validation. SXML Schema makes this possible for SXML and removes one of the main objections to s-expressions as a general-purpose representation format. The schema format might be more widely adopted if it was integrated with the SXML toolkit.

The CA description format could be used, with a suitable schema, as a template for certification polices and certification practice statements. The template would be used by CAs as the 'source code' for their policies and the natural language text of the policies would be generated from the template. A similar idea has been suggested in relation to the XML CP/CPS format developed by Ball et al.[16]. If it is possible to devise a system that reliably translates the terse features of the description to natural language then this will eliminate the need to make the reverse translation, from natural language to the restricted set of elements and values.

This could make localisation of CP and CPS documents to multiple languages semi-automatic. Each language would have its own set of statements, corresponding to the policy provisions, written by a professional human translator. Existing techniques used for localisation of software and technical documentation could be applied to substitute the appropriate localised text. Such a system would certainly be appealing in the European context where there are many official and working languages.

The trust matrices build upon the previous work by Coghlan. They are an improvement in terms of flexibility of the trust evaluation model and of performance, mostly due to the trust evaluation engine. The trust matrices have sufficient potential to be worthy of consideration by the EUGrid PMA, the other IGTF PMAs, and similar bodies, for use in assessing new and existing CAs.

To make the system more accessible to its potential users, the SXML-based editing software for CA descriptions and rulesets should be completed and should be integrated with the trust matrices. This could provide a suite of web-based tools for the configuration and use of the trust evaluation system. The graphical display of the trust matrices could be enhanced to show the evaluation results in a more visual way, using colour-codes and icons. The tabular displays could be supplemented with automatically generated diagrams that, for example, show which CAs meet each policy's requirements.

The validation services are a further contribution of this thesis. Previous work on trust evaluation, described in Chapter 2, was not integrated into online, on-demand services. Conversely, the previously developed validation services did not make use of trust evaluation techniques. The validation services presented in this thesis combine the trust evaluation engine with an online service to provide an on-demand trust evaluation service.

The OCSP-based validation service makes use of the OCSP protocol. This allows it to be used by a wide range of existing software, including some grid software. It has been tested with the OpenSSL OCSP client. The OCSP protocol supports non-critical extensions and the validation service makes use of this possibility to return an evaluation result in each response. A Java client based on the Novosec Bouncy Castle OCSP extensions has been modified to detect the presence of a trust evaluation result extension. The developers of the OGRO OCSP client and server are interested in including trust evaluation support in their tools, which may be included in the Globus toolkit in the future. The author intends to pursue inclusion of support for the trust evaluation system presented in this thesis in OGRO.

The use of OCSP for validation of a certificate's issuer could be considered a slight misuse of the protocol. In addition, the current version of OCSP does not support sending an entire certificate in the request. For these reasons support for other protocols such as SCVP and XKMS should be investigated. While these protocols do not yet have major support in grid software this is something that can be rectified, especially as there is a clear need for such services.

The trust evaluation engine could be used in further applications that require a set of CAs to be evaluated against a policy. When a system administrator is deciding which CAs to install in the trusted store of desktop web browsers or of grid servers he must make a decision as to which CAs to trust. There may be some guidelines in force in the organisation or an external policy may apply, perhaps from a PMA. Once the administrator has made the decision that is not, however, the end of the matter. Policies and CA practices change. A CA which may have been trustworthy in the past might now be considered less so (in fact one could argue this should be the default policy), or a new CA may have appeared and thereby created the need for its subscribers to be authenticated. The administrator must have a system for responding to changes. If the CA roots of trust are being distributed in software packages then the administrator needs a mechanism to automatically decide if a new package should be installed or not based on the current policy. This is a situation where the trust evaluation engine could be employed.

A related case arises in building cryptographic bridges. Current grid software based on Globus and OpenSSL does not fully support bridge CAs but this will not always be the case. Bridge CAs are appealing because of the cryptographic basis for trust but the means for establishing the initial trust is usually based on a policy. The trust evaluation techniques presented in this thesis could be used when founding a bridge. It might also be possible to create an online bridge CA that issues temporary bridge certificates for member CAs to relying parties. The relying party would need to regularly renew bridge certificates and the CA could take this opportunity to re-evaluate trust in the member CAs. The bridge CA could alternatively issue (and revoke) longer-term bridge certificates on a similar basis.

The TERENA Academic CA Repository (TACAR) maintains a collection of roots of trust for academic CAs from around the world. It is intended as an independent site which can be consulted to verify the authenticity of CA certificates downloaded from other sources. TERENA does not accredit the CAs that appear in its repository: this is outside the scope of its operations. However, it does classify the CAs according to accreditation. The trust evaluation engine, could be used to update these classifications when a CA or the accrediting body changes its policy. An alternative is to leave the classifications as they are but to provide an indication of the level of compliance with the accreditation policy.

Trust and security services can be valued within a socio-economic model along with other grid services. This is a major focus of the Metagrid work on interoperability and resource allocation through *social grid agents* [141]. Various social and economic models of cooperation and competition are under investigation. The trust evaluation system could be applied to calculate the utility or market value of CAs for such purposes. Should a CA with a poor evaluation result have a lower price than a more trustworthy CA – a discount for lower quality of service – or should it be given a higher cost to emphasise the risk and discourage its use? It will depend, amongst other factors, on whether the value is used for payment and who gets paid.

The trust evaluation engine could itself be embodied in a social grid agent and provide trust evaluation services to other agents in the society. The KeyNote credential authority and delegation service described in Chapter 6 could also become agents in the Metagrid society. However, the issue arises of how security agents could themselves be trusted in such a dynamic model.

As mentioned in Chapter 3, trust evaluation can be applied not only to traditional certification authorities but also to other credential issuers, such as short-lived credential services. The IGTF is developing an authentication profile for such services. These short-lived credential services make use of the local identity services at a particular institution, for example Kerberos, Windows Active Directory, or LDAP, to provide short-lived grid proxy credentials. Whereas most existing grid CAs serve an entire country or large region, these services serve a single site. This could lead to a massive proliferation of identity providers. For an individual relying party, manual evaluation on a regular basis of hundreds of services would be infeasible. The trust evaluation system, with appropriate rulesets and service descriptions, could automate such an evaluation.

The techniques used by the trust evaluation engine could also be applied more generally wherever a rule-based policy evaluation is appropriate and useful. Such evaluation could be used in authorization mechanisms to determine if a particular request should be granted. Another potential use is in establishing authentication federations.

The EU e-Infrastructures Reflection Group has indicated that federation of authentication providers for education and research users is an important area. European and international efforts such as the Géant2 eduGAIN project require a basis for establishment of trust between institutions at various levels [108]. A trust evaluation system such as the one described in this thesis could be used to apply the relevant policies to institutional PKI or other forms of identity services. Furthermore, this suggests that services should be built around the evaluation engine which support SAML authentication assertions in order to interoperate with federated identity systems.

The research of this thesis has dealt with the issue of trust in the context of grid computing. This chapter has discussed the contributions made in this thesis – the major two being a model for evaluation of trust in grid certification authorities, and a proof-of-concept implementation – and explored the future directions for research and development that they make possible. It is the belief of the author that this thesis has been successful in showing that automatic trust evaluation for the grid is desirable and feasible.

Bibliography

[1] A-Select. A-select 1.4.1 operational concept and system description, 2006. URL `http://a-select.surfnet.nl/doc/aselect1.4.1.ocd.html`.

[2] Harold Abelson, Gerald Jay Sussman, and Julie Sussman. *Structure and Interpretation of Computer Programs – Second Edition*. MIT Press, July 1996. URL `http://mitpress.mit.edu/sicp/`.

[3] WebTrust. *WebTrust Program for Certification Authorities*. AICPA–CICA, August 2000. URL `http://ftp.webtrust.org/webtrust_public/tpafile7-8-03fortheweb.doc`.

[4] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell Agnello, Á. Frohner, A. Gianoli, K. Lőrentey, and F. Spataro. VOMS, an authorization system for virtual organizations. In *1st European Across Grids Conference, Santiago de Compostela*, pages 33–40. Springer-Verlag LNCS 2970, 13–14 February 2003.

[5] W. Allcock, J. Bester, J. Bresnahan, S. Meder, P. Plaszczak, and S. Tuecke. *GridFTP: Protocol Extensions to FTP for the Grid*, April 2003. URL `http://globus.org/alliance/publications/papers/GFD-R.0201.pdf`.

[6] Jim Almond and Dave Snelling. UNICORE: uniform access to supercomputing as an element of electronic commerce. *Future Generation Computer Systems*, 15(5–6):539–548, October 1999. URL `http://www.sciencedirect.com/science/article/B6V06-405TF92-2/2/42d130ca710111ba90b912bae9da258c`.

[7] David P. Anderson. BOINC: A system for public-resource computing and storage. In *5th IEEE/ACM International Workshop on Grid Computing (Grid 2004)*, November 2004. URL `http://boinc.berkeley.edu/grid_paper_04.pdf`.

[8] ANSI. X9.79-1:2001 PKI practices and policy framework for the financial services industry, 2001. URL `http://www.x9.org/catalog2.cfm?item_no=%24%23%20%2F%217%20%210%0A&pub_item=%2334%2A%3B%0A`. Annex B, (Normative) Certification Authority Control Objectives.

[9] M. Antonioletti, M.P. Atkinson, R. Baxter, A. Borley, N.P. Chue Hong, B. Collins, N. Hardman, A. Hume, A. Knox, M. Jackson, A. Krause, S. Laws, J. Magowan, N.W. Paton, D. Pearson, T. Sugden, P. Watson, and M. Westhead. The design and implementation of grid database services in OGSA-DAI. *Concurrency and Computation: Practice and Experience*, 17(2-4):357–376, February 2005.

[10] Apache Software Foundation. *Apache Axis*, April 2006. URL `http://ws.apache.org/axis/`.

[11] Apache Software Foundation. *Apache Tomcat*, 2006. URL `http://tomcat.apache.org/`.

[12] APGridPMA. *Asia Pacific Grid Policy Management Authority*, 2006. URL `http://www.apgridpma.org/`.

[13] John Michael Ashley. *The GNU Privacy Handbook*. The Free Software Foundation, 1999. URL `http://www.gnupg.org/(en)/documentation/guides.html#gph`.

[14] J. Astalos, R. Cecchini, B.A. Coghlan, R.D. Cowles, U. Epting, T.J. Genovese, J. Gomes, D. Groep, M. Gug, A.B. Hanushevsky, M. Helm, J.G. Jensen, C. Kanellopoulos, D.P. Kelsey, R. Marco, I. Neilson, S. Nicoud, D. O'Callaghan, D. Quesnel, I. Schaeffner, L. Shamardin, D. Skow, M. Sova, A. Wäänänen, P. Wolniewicz, and W. Xing. International grid CA interworking, peer review and policy management through the European DataGrid certification authority coordination group. In Peter M.A. Sloot, Alfons G. Hoekstra, Thierry Priol, Alexander Reinefeld, and Marian Bubak, editors, *Advances in Grid Computing – EGC 2005*, LNCS3470, pages 275–285, Amsterdam, The Netherlands, February 2005. Springer. ISBN 3-540-26918-5.

[15] Bartosz Balis, Marian Bubak, Wojciech Rzasa, and Tomasz Szepieniec. Efficiency of the gsi secured network transmission. In *Computational Science – ICCS 2004*, Lecture Notes in Computer Science, pages 107–115. Springer-Verlag, 2004. URL `http://www.springerlink.com/content/auwxyn3f5lm4m7y5`. LNCS 3036.

[16] E. Ball, D. W. Chadwick, and A. Basden. *Trust in the Network Economy*, volume 2 of *Evolaris*, chapter The implementation of a system for evaluating trust in a PKI environment, pages 263–279. Springer, 2003. ISBN 3-211-06853-8.

[17] A. Baratloo, M. Karaul, Z. M. Kedem, and P. Wyckoff. Charlotte: Metacomputing on the web. In *Proc. of the 9th International Conference on Parallel and Distributed Computing Systems (PDCS-96)*, 1996. URL `citeseer.ist.psu.edu/baratloo96charlotte.html`.

[18] Tom Barton, Jim Basney, Tim Freeman, Tom Scavo, Frank Siebenlist, Von Welch, Rachana Ananthakrishnan, Bill Baker, Monte Goode, and Kate Keahey. Identity federation and

attribute-based authorization through the Globus Toolkit, Shibboleth, GridShib, and MyProxy. In *5th Annual PKI R&D Workshop*, 4–6 April 2006. URL `http://middleware.internet2.edu/pki06/proceedings/welch-idfederation.pdf`.

[19] William Bell, Diana Bosio, Wolfgang Hoschek, Peter Kunszt, Gavin McCance, and Mika Silander. Project Spitfire – towards grid web service databases. Technical report, July 2002.

[20] Paul V. Biron and Ashok Malhotra. *XML Schema Part 2: Datatypes (Second Edition)*, October 2004. URL `http://www.w3.org/TR/xmlschema-2/`. W3C Recommendation.

[21] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proc. IEEE Symposium on Security and Privacy*, 1996.

[22] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. *The KeyNote Trust-Management System Version 2*, September 1999. URL `ftp://ftp.isi.edu/in-notes/rfc2704.txt`. RFC 2704.

[23] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The role of trust management in distributed systems security. In Vitek and Jensen, editors, *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*. Springer-Verlag, 1999.

[24] Matt Blaze. Using the KeyNote trust management system, March 2001. URL `http://www.crypto.com/trustmgt/`.

[25] Per Bothner. A GCC-based Java implementation. In *IEEE Compcon 1997 Proceedings*, pages 174–178, February 1997. URL `http://citeseer.csail.mit.edu/bothner97gccbased.html`.

[26] Per Bothner. Kawa—compiling dynamic languages to the Java VM. In *Proceedings of the USENIX 1998 Technical Conference, FREENIX Track*, New Orleans, LA, June 1998. USENIX Association. URL `http://citeseer.csail.mit.edu/bothner98kawa.html`. See also: http://www.gnu.org/software/kawa/.

[27] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*, March 1997. URL `ftp://ftp.isi.edu/in-notes/rfc2119.txt`. RFC 2119.

[28] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and Fran cois Yergeau. *Extensible Markup Language (XML) 1.0 (Fourth Edition)*, August 2006. W3C Recommendation.

[29] Yann Busnel. Simple Certificate Validation Protocol : Etude du protocole et implémentation de la vérification dans OpenSSL. Technical report, ENST Bretagne, July 2003. URL `http://www-lor.int-evry.fr/˜maknavic/CADDISC/scvpDoc.pdf`.

[30] R. Butler and T.J. Genovese. *Global Grid Forum Certificate Policy Model*, June 2003. URL `http://forge.gridforum.org/projects/ggf-editor/document/GFD-C.16/en/1`.

[31] Andrew Butterfield et al. *VDM♣: Mathematical Structure for Formal Methods*, May 2000. URL `https://www.cs.tcd.ie/Andrew.Butterfield/IrishVDM/talks/Irish-School-Slides.pdf`.

[32] Vinny Cahill, Brian Shand, Elizabeth Gray, Ciarán Bryce, Nathan Dimmock, Andrew Twigg, Jean Bacon, Colin English, Waleed Wagealla, Sotirios Terzis, Paddy Nicon, Giovanna di Marzo Serugendo, Jean-Marc Seigneur, Marco Carbone, Karl Krukow, Christian Jensen, Yong Chen, and Mogens Nielsen. Using trust for secure collaboration in uncertain environments. *IEEE Pervasive Computing*, 2(3):52–61, Jul-Sep 2003.

[33] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. *OpenPGP Message Format*, November 1998. URL `ftp://ftp.isi.edu/in-notes/rfc2440.txt`. RFC 2440.

[34] Valentina Casola, Massimiliano Rak, Rosa Preziosi, and Luigi Troiano. Security level evaluation: Policy and fuzzy techniques. In *International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2*, page 752, 2004. URL `http://csdl.computer.org/comp/proceedings/itcc/2004/2108/02/210820752abs.htm`.

[35] Valentina Casola, Antonino Mazzeo, Nicola Mazzocca, and Massimiliano Rak. An innovative policy-based cross certification methodology for public key infrastructures. In *EuroPKI 2005*, pages 100–117, 2005. URL `http://dx.doi.org/10.1007/11533733_7`.

[36] Valentina Casola, Rosa Preziosi, Massimiliano Rak, and Luigi Troiano. A reference model for security level evaluation: Policy and fuzzy techniques. *Journal of Universal Computer Science*, 11(1):150–174, 2005. URL `http://www.jucs.org/jucs_11_1/a_reference_model_for`.

[37] CERN. LHC@home, 2006. URL `http://athome.web.cern.ch/`.

[38] CertiVeR. Certificate verification & revocation, 2006. URL `http://www.certiver.com/`.

[39] David W. Chadwick and Andrew Basden. Evaluating trust in a public key certification authority. *Computers and Security*, 20(7):592–611, November 2001.

[40] D.W. Chadwick and A. Otenko. The PERMIS X.509 role based privilege management infrastructure. *Future Generation Computer Systems*, 19(2):277–289, February 2003. URL `http://www.cs.kent.ac.uk/pubs/2003/2109`.

[41] Madhu Chetty and Rajkumar Buyya. Weaving computational grids: How analogous are they with electrical grids? *Computing in Science and Engineering*, 4(4):61–71, July 2002. URL `http://www.gridbus.org/papers/WeavingGrid.pdf`.

[42] S. Chokhani, W. Ford, R. Sabett, C. Merrill, and S. Wu. *Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework*, November 2003. URL `ftp://ftp.isi.edu/in-notes/rfc3647.txt`. RFC 3647.

[43] Bernd O. Christiansen, Peter R. Cappello, Mihai F. Ionescu, Michael O. Neary, Klaus E. Schauser, and Daniel Wu. Javelin: Internet-based parallel computing using java. *Concurrency – Practice and Experience*, 9(11):1139–1160, 1997.

[44] James Clark. *XSL Transformations (XSLT) Version 1.0*, November 1999. URL `http://www.w3.org/TR/xslt`. W3C Recommendation.

[45] ClimatePrediction.net. Climateprediction.net, 2006. URL `http://climateprediction.net/`.

[46] Brian Coghlan. CA trust matrices, July 2002. Talk presented at GGF5 – The Fifth Global Grid Forum.

[47] Andrew W. Cooke, Alasdair J. G. Gray, Werner Nutt, James Magowan, Manfred Oevers, Paul Taylor, Roney Cordenonsi, Rob Byrom, Linda Cornwall, Abdeslem Djaoui, Laurence Field, Steve Fisher, Steve Hicks, Jason Leake, Robin Middleton, Antony J. Wilson, Xiaomei Zhu, Norbert Podhorszki, Brian A. Coghlan, Stuart Kenny, David O'Callaghan, and John Ryan. The relational grid monitoring architecture: Mediating information about the grid. *Journal of Grid Computing*, 2:323–339, December 2004.

[48] L.A. Cornwall, J. Jensen, D.P. Kelsey, A. Frohner, D. Kouril, F. Bonnassieux, S. Nicoud, K. Lorentey, J. Hahkala, M. Silander, R. Cecchini, V. Ciaschini, L. dell'Agnello, F. Spataro, D. O'Callaghan, O. Mulmo, G.-L. Volpato, D. Groep, M. Steenbakkers, and A. McNab. Authentication and authorization mechanisms for multi-domain grid environments. *Journal of Grid Computing*, 2:301–311, December 2004.

[49] Linda Cornwall, David Kelsey, et al. DataGrid security requirements and Testbed-1 security implementation, D7.5, 2002. URL `https://edms.cern.ch/document/340234`.

[50] Mark J. Cox, Ralf S. Engelschall, Stephen Henson, Ben Laurie, Eric A. Young, and Tim J. Hudson. *OpenSSL: The Open Source toolkit for SSL/TLS*, 2006. URL `http://www.openssl.org/`.

[51] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A resource management architecture for metacomputing systems. In *The 4th Workshop on Job Scheduling Strategies for Parallel Processing*, pages 62–82. Springer-Verlag LNCS 1459, 1998.

[52] DataTAG. Research & technological development for a data trans-Atlantic grid, 2004. URL `http://datatag.web.cern.ch/`.

[53] T. de Fanti, I. Foster, M. Papka, R. Stevens, and T.Kuhfuss. Overview of the I-WAY: Wide area visual supercomputing. *International Journal of Supercomputer Applications*, 10(2): 123–130, 1996.

[54] DEISA. Distributed European Infrastructure for Supercomputing Applications, 2004. URL `http://www.deisa.org/`.

[55] David del Vecchio, Marty Humphrey, Jim Basney, and Nataraj Nagaratnam. CredEx: User-centric credential management for grid and web services. In *Proc. International Conference on Web Services (ICWS)*. IEEE Computer Society, July 2005.

[56] T. Dierks and C. Allen. *The TLS Protocol Version 1.0*, January 1999. URL `ftp://ftp.isi.edu/in-notes/rfc2246.txt`. RFC 2246.

[57] distributed.net. Project RC5, 2005. URL `http://distributed.net/rc5/`.

[58] Donald Eastlake, Joseph Reagle, David Solo, Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, and Ed Simon. *XML-Signature Syntax and Processing*, February 2002. URL `http://www.w3.org/TR/xmldsig-core/`. W3C Recommendation.

[59] Ebay. Evaluating a member's reputation, October 2006. URL `http://pages.ebay.co.uk/help/feedback/evaluating-feedback.html`.

[60] EDG. European DataGrid Java security, 2003. URL `http://edg-wp2.web.cern.ch/edg-wp2/security/edg-java-security.html`.

[61] EGEE. Enabling Grids for E-science, April 2006. URL `http://www.eu-egee.org/`.

[62] eIRG. White paper, version 5.51, April 2004. URL `http://www.e-irg.org/publ/2004-Dublin-eIRG-whitepaper.pdf`.

[63] Taher ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.

[64] ETSI. ETSI TS 102 042 v1.1.1 (2002-04), Policy requirements for certification authorities issuing public key certificates, April 2002. URL `http://portal.etsi.org/docbox/EC_Files/EC_Files/ts_102042v010101p.pdf`. Clause 7, Requirements on CA practice.

[65] ETSI. ETSI TS 101 456 v1.2.1 (2002-04), Policy requirements for certification authorities issuing qualified certificates, April 2002. URL

http://webapp.etsi.org/action/PU/20020402/ts_101456v010201p.pdf. Clause 7, Requirements on CA practice.

[66] EUGridPMA. *Accreditation Procedures.* EU Policy Management Authority for Grid Authentication in e-Science, April 2004. URL http://eugridpma.org/guidelines/EUGridPMA-accreditation-20040402-1-0.pdf.

[67] EUGridPMA. *Profile for Traditional X.509 Public Key Certification Authorities with secured infrastructure, version 4.0.* European Policy Management Authority for Grid Authentication in e-Science, October 2005. URL http://www.eugridpma.org/guidelines/IGTF-AP-classic-20050930-4-0.pdf.

[68] S. Farrell and R. Housley. *An Internet Attribute Certificate Profile for Authorization*, April 2002. URL ftp://ftp.isi.edu/in-notes/rfc3281.txt. RFC 3281.

[69] Adam Ferrari, Frederick Knabe, Marty Humphrey, Steve Chapin, and Andrew Grimshaw. A flexible security system for metacomputing environments. In *Proc. 7th International Conference on High-Performance Computing and Networking Europe (HPCN '99)*, pages 370–380. Springer-Verlag LNCS 1593, April 1999. URL http://legion.virginia.edu/papers/CS-94-20.pdf.

[70] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures.* PhD thesis, University of California, Irvine, 2000.

[71] S. Fisher. Relational model for information and monitoring, 2001.

[72] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke. A directory service for configuring high-performance distributed computations. In *Proc. 6th IEEE Symp. on High Performance Distributed Computing*, pages 365–375. IEEE Computer Society Press, 1997.

[73] S.N. Foley, T.B. Quillinan, J.P. Morrison, D.A. Power, and J.J. Kennedy. Exploiting KeyNote in WebCom: Architecture neutral glue for trust management. In *Proc. Fifth Nordic Workshop on Secure IT Systems*, pages 101–119, October 2000.

[74] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997.

[75] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In *ACM Conference on Computers and Security*, pages 83–91. ACM-P, 1998.

[76] Ian Foster. What is the Grid? A three point checklist, July 2002. URL http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf.

[77] Ian Foster. Globus toolkit version 4: Software for service-oriented systems. In *IFIP International Conference on Network and Parallel Computing*, pages 2–13. Springer-Verlag LNCS 3779, 2005.

[78] Ian Foster, Carl Kesselman, and Steven Tuecke. The Anatomy of the Grid – enabling scalable virtual organizations. *International Journal Supercomputer Applications*, 15(3): 200–222, 2001.

[79] Fraunhofer-Gesellschaft. *New Security Infrastructure*, 2002. URL `http://sit.sit.fraunhofer.de/_SIT-Projekte/NSI/index-en.html`.

[80] T. Freeman, R. Housley, A. Malpani, D. Cooper, and T. Polk. *Standard Certificate Validation Protocol (SCVP)*, March 2006. URL `http://tools.ietf.org/wg/pkix/draft-ietf-pkix-scvp/`.

[81] James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steve Tuecke. Condor-G: A computation management agent for multi-institutional grids. *Cluster Computing*, 5: 237–246, 2002.

[82] Fabrizio Gagliardi, Bob Jones, Mario Reale, and Stephen Burke. European DataGrid project: Experiences of deploying a large scale testbed for e-science applications. In *Performance Evaluation of Complex Systems: Techniques and Tools. Performance 2002, Tutorial Lectures*. Springer-Verlag LNCS 2459, 2002. ISBN 3-540-44252-9.

[83] GGF. Open Grid Services Architecture glossary of terms, January 2005.

[84] J. Gomes, M. David, J. Martins, L. Bernardo, A. García, M. Hardt, H. Kornmayer, J. Marco, R. Marco, D. Rodríguez, I. Diaz, D. Cano, J. Salt, S. Gonzalez, J. Sánchez, F. Fassi, V. Lara, P. Nyczyk, P. Lason, A. Ozieblo, P. Wolniewicz, M. Bluj, K. Nawrocki, A. Padee, W. Wislicki, C. Fernández, J. Fontán, Y. Cotronis, E. Floros, G. Tsouloupas, W. Xing, M. Dikaiakos, J. Astalos, B. Coghlan, E. Heymann, M. Senar, C. Kanellopoulos, A. Ramos, and D. Groen. Experience with the international testbed in the CrossGrid project. In Peter M.A. Sloot, Alfons G. Hoekstra, Thierry Priol, Alexander Reinefeld, and Marian Bubak, editors, *Advances in Grid Computing – EGC 2005*, LNCS 3470, pages 98–110, Amsterdam, The Netherlands, February 2005. Springer. ISBN 3-540-26918-5. URL `http://www.springerlink.com/content/cq2n0pg3ukm0vv6v`.

[85] Andrew S. Grimshaw, William A. Wulf, James C. French, Alfred C. Weaver, and Paul F. Reynolds, Jr. A synopsis of the legion project. Technical Report CS-94-20, University of Virginia, June 1994. URL `http://legion.virginia.edu/papers/CS-94-20.pdf`.

[86] Andrew S. Grimshaw, William A. Wulf, et al. Legion - a view from 50, 000 feet. In *5th IEEE International Symposium on High Performance Distributed Computing (HPDC-5 '96)*, pages 89–99, 1996. URL http://legion.virginia.edu/copy-hpdc5.html.

[87] Phillip Hallam-Baker and Shivaram H. Mysore. *XML Key Management Specification (XKMS 2.0)*, 28 June 2005. URL http://www.w3.org/TR/xkms2/.

[88] A. Hanushevsky and R. Cowles. Virtual smart card, 2002. URL http://www.slac.stanford.edu/~abh/vsc/.

[89] Frank Hecker. *Mozilla CA Certificate Policy (Version 1.0).* Mozilla Foundation, November 2005. URL http://www.mozilla.org/projects/security/pki/nss/ca-certificates/policy.html.

[90] Michael Helm. Validation service, January 2006. URL http://www.eugridpma.org/agenda/askArchive.php?base=agenda&categ=a054&id=a054s3t8/transparencies.

[91] Shelley Henderson and James A. Jokl. *Grids and Bridges.* University of Southern California, University of Virginia, July 2005. URL http://www.educause.edu/ir/library/powerpoint/PKI0509.pps.

[92] R. Housley, W. Polk, W. Ford, and D. Solo. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, April 2002. URL ftp://ftp.isi.edu/in-notes/rfc3280.txt. RFC 3280.

[93] Eoin Huggard and Thomas Quillinan. Jkeynote: a java implementation of the keynote trust management system, 2003. URL http://kargellan.ucc.ie/JKeyNote/.

[94] IGTF. *Profile for Traditional X.509 Public Key Certification Authorities with secured infrastructure (Version 4.0).* International Grid Trust Federation, October 2005. URL http://www.eugridpma.org/guidelines/IGTF-AP-classic-20050930-4-0.html.

[95] IGTF. *Trust on the Grid Goes Global.* International Grid Trust Federation, October 2005. URL http://www.gridpma.org/docs/igtf-newsrelease-20051005.pdf.

[96] ITU. Information technology – open systems interconnection – the directory: Public-key and attribute certificate frameworks, August 2005. URL http://www.itu.int/rec/T-REC-X.509-200508-I. International Telecommunication Union Recommendation X.509 (08/05).

[97] Panu Kalliokoski. *SRFI 69: Basic Hash Tables*, September 2005. URL http://srfi.schemers.org/srfi-69/srfi-69.html.

[98] Richard Kelsey, William Clinger, and Jonathan Rees. Revised^5 report on the algorithmic language Scheme. *ACM SIGPLAN Notices*, 33(9):26–76, 1998. URL `http://www.schemers.org/Documents/Standards/R5RS/HTML/`.

[99] Oleg Kiselyov. *S-exp-based XML parsing/query/conversion*, September 2004. URL `http://ssax.sourceforge.net/`.

[100] Oleg Kiselyov. *SXML Specification Version 3.0*, March 2004. URL `http://okmij.org/ftp/Scheme/xml.html#SXML-spec`.

[101] Olga Kornievskaia, Peter Honeyman, Bill Doster, and Kevin Coffman. Kerberized credential translation: A solution to web access control. In *Proceedings of the 10th USENIX Security Symposium*, pages 235–250, 2001. URL `http://www.citi.umich.edu/techreports/reports/citi-tr-01-5.pdf`.

[102] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Leboisky. SETI@home – massively distributed computing for SETI. *Computing in Science & Engineering*, 3(1): 78–83, January–February 2001.

[103] Daniel Kouřil and Jim Basney. A credential renewal service for long-running jobs. In *6th IEEE/ACM International Workshop on Grid Computing (Grid 2005)*, November 2005. URL `http://www.ncsa.uiuc.edu/~jbasney/proxy-renewal.pdf`.

[104] Miroslaw Kupczyk, Rafal Lichwala, Norbert Meyer, Bartek Palak, Marcin Plciennik, and Pawel Wolniewicz. Mobile work environment for grid users. In *1st European Across Grids Conference, Santiago de Compostela*, pages 132–138. Springer-Verlag LNCS 2970, 13–14 February 2003.

[105] LCG. *Large Hadron Collider Computing Grid Project*, 2006. URL `http://lcg.web.cern.ch/LCG/`.

[106] Legion of the Bouncy Castle. *Bouncy Castle Cryptography Libraries*, 2006. URL `http://bouncycastle.org/`.

[107] Michael Litzkow, Miron Livny, and Matthew Mutka. Condor – a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, June 1988.

[108] D. R. Lopez, R. Castro, B. Kerver, T. Lenggenhager, I. Melve, M. Milinovic, J. Rauschenbach, K. Wierenga, S. Winter, and H. Ziemek. Deliverable DJ5.2.2: GÉANT2 authorisation and authentication infrastructure (AAI) architecture, October 2005. GN2-05-192v6.

[109] Jesús Luna, June 2006. Private email correspondence with the author.

[110] Jesús Luna, Manel Medina, and Oscar Manso. Using OGRO and CertiVeR to improve OCSP validation for grids. In Yeh-Ching Chung and José E. Moreira, editors, *Advances in Grid and Pervasive Computing, First International Conference, GPC 2006, Taichung, Taiwan, May 3–5, 2006, Proceedings*, volume 3947 of *Lecture Notes in Computer Science*, pages 12–21. Springer, 2006. ISBN 3-540-33809-8. URL `http://dx.doi.org/10.1007/11745693_2`.

[111] Soha Maad, Brian Coghlan, Geoff Quigley, John Ryan, Eamonn Kenny, and David O'Callaghan. Towards a complete grid filesystem functionality. *Future Generation Computer Systems*, submitted to special issue on Data Analysis, Access and Management on Grids, 2006.

[112] Mícheál Mac an Airchinnigh. *Conceptual Models and Computing.* PhD thesis, University of Dublin, Trinity College, October 1990. URL `https://www.cs.tcd.ie/Andrew.Butterfield/IrishVDM/reading/PhD-MMA.pdf`.

[113] Stephen Paul Marsh. *Formalising Trust as a Computational Concept.* PhD thesis, Department of Mathematics and Computer Science, University of Stirling, April 1994. URL `citeseer.ist.psu.edu/marsh94formalising.html`.

[114] Justin Mason. Filtering spam with SpamAssassin. In *HEANet Annual Conference 2002*, November 2002. URL `http://spamassassin.apache.org/presentations/HEANet_2002/`.

[115] John McCarthy. Recursive functions of symbolic expressions and their computation by machine, part I. *Communications of the ACM*, 3(4):184–195, 1960.

[116] Andrew McNab. Testbed security issues and GACL, May 2002. URL `http://www.hep.man.ac.uk/u/mcnab/grid/security17may02/`.

[117] Andrew McNab. The gridsite web/grid security system. *Software: Practice and Experience*, 35(9):827–834, 2005.

[118] Andrew McNab. Pool accounts patch for Globus, 2002. URL `http://www.gridpp.ac.uk/gridsite/gridmapdir/`.

[119] Microsoft. *Microsoft Root Certificate Program*, 2006. URL `http://www.microsoft.com/technet/archive/security/news/rootcert.mspx`.

[120] J. Morrison, D. Power, and J. Kennedy. A condensed graphs engine to drive metacomputing. In *Proc. International conference on parallel and distributed processing techniques and applications (PDPTA 1999)*, Las Vegas, Nevada, June 28 – July 1, 1999.

[121] John P. Morrison. *Condensed Graphs: Unifying Availability-Driven, Coercion-Driven, and Control-Driven Computing*. PhD thesis, Technische Universiteit Eindhoven, October 1996.

[122] John P. Morrison, Sunil John, David A. Power, Neil Cafferkey, and Adarsh Patil. A grid application development platform for WebCom-G. In *Cluster Computing and Grid (CCGrid) 2005*, May 2005.

[123] J.P. Morrison, B. Coghlan, A. Shearer, S. Foley, D. Power, and R. Perrot. WebCom-G: A condidate middleware for Grid-Ireland. *High Performance Computing Applications and Parallel Processing*, 2005.

[124] Olle Mulmo et al. EGEE global security architecture, 2004. URL `https://edms.cern.ch/document/487004/`.

[125] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*, June 1999. URL `ftp://ftp.isi.edu/in-notes/rfc2560.txt`. RFC 2560.

[126] Michael Myers, Rich Ankney, Carlisle Adams, Stephen Farrell, and Carlin Covey. *Online Certificate Status Protocol, version 2*, March 2001. URL `http://tools.ietf.org/wg/pkix/draft-ietf-pkix-ocspv2/`. Expired Internet Draft.

[127] NIST. *Advanced Encryption Standard (AES)*, November 2001. URL `http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf`. FIPS 197.

[128] NIST. *Data Encryption Standard (DES)*, October 1999. URL `http://www.csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`. FIPS 46-3.

[129] NIST. *Digital Signature Standard (DSS)*, October 2001. URL `http://www.csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf`. FIPS 186-2.

[130] NIST. *Secure Hash Standard (SHS)*, February 2004. URL `http://www.csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf`. FIPS 180-2.

[131] Jason Novotny, Steven Tuecke, and Von Welch. An online credential repository for the grid: Myproxy. In *10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10 '01)*, page 104, 2001.

[132] David O'Callaghan and Brian Coghlan. Bridging secure WebCom and European DataGrid security for multiple VOs over multiple grids. In *Proc. ISPDC 2004*, pages 225–231, Cork, Ireland, July 2004. IEEE Computer Society.

[133] David O'Callaghan and Brian Coghlan. On-demand trust evaluation. In *7th IEEE/ACM International Conference on Grid Computing (Grid 2006)*, 2006.

[134] OGF. CA Operations working group, 2006. URL `https://forge.gridforum.org/sf/projects/caops-wg`.

[135] Jon Ølnes. DNV VA white paper: PKI interoperability by an independent, trusted validation authority. Technical Report 2005-0673, Det Norske Veritas, 6 June 2005. URL `http://www.dnv.com/binaries/Report-2005-0673_tcm4-159646.pdf`.

[136] Jon Ølnes. PKI interoperability by an independent, trusted validation authority. In *5th Annual PKI R&D Workshop*, 4–6 April 2006. URL `http://middleware.internet2.edu/pki06/proceedings/olnes-interoperability.pdf`.

[137] Namje Park, Kiyoung Moon, Sungwon Sohn, and Cheehang Park. Certificate validation scheme of open grid service usage XKMS. In *Grid and Cooperative Computing, Second International Workshop, GCC 2003, Shanghai, China, December 7-10, 2003, Revised Papers, Part I*, pages 849–858, 2003. URL `http://springerlink.metapress.com/openurl.asp?genre=article\&issn=0302-9743\&volume=3032\&spage=849`.

[138] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. A Community Authorization Service for group collaboration, 2002. URL `http://globus.org/alliance/publications/papers/CAS_2002_Revised.pdf`.

[139] G. Pierantoni, O. Lyttleton, D. O'Callaghan, G. Quigley, E. Kenny, and B. Coghlan. Multi-grid and multi-vo job submission based on a unified computational model. In *Cracow Grid Workshop (CGW'05)*, Cracow, Poland, November 2005.

[140] G. Pierantoni, O. Lyttleton, D. O'Callaghan, G. Quigley, and E. Kenny. Interoperability using a Metagrid architecture. In *Proc. ExpGrid workshop at The 15th IEEE International Symposium on High Performance Distributed Computing (HPDC2006)*, Paris, France, June 2006.

[141] Gabriele Pierantoni, Brian Coghlan, and Eamonn Kenny. Agent-based societies for the sharing, brokerage and allocation of grid resources. In *Workshop on state-of-the-art in scientific and parallel computing. Umeøa, Sweden, June 18–21, 2006*, 2006.

[142] Paul Prescod. XML is not s-expressions, February 2003. URL `http://www.prescod.net/xml/sexprs.html`.

[143] Rajesh Raman, Miron Livny, and Marvin H. Solomon. Matchmaking: Distributed resource management for high throughput computing. In *7th IEEE International Symposium on High Performance Distributed Computing (HPDC-7 '98)*, pages 140–146, 1998.

[144] John Resig. *jQuery: New Wave Javascript*, 2006. URL http://jquery.com/.

[145] R. Rivest. *The MD5 Message-Digest Algorithm*, April 1992. URL ftp://ftp.isi.edu/in-notes/rfc1321.txt. RFC 1321.

[146] R. L. Rivest, A. Shamir, and L. A. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. URL http://citeseer.ist.psu.edu/rivest78method.html.

[147] Ron Rivest. *S-Expressions*, May 1997. URL http://theory.lcs.mit.edu/~rivest/sexp.txt. Expired Internet Draft.

[148] Judith Rossebo and Jon Ølnes. Single sign-on secure service access, March 2003. URL http://www.freepatentsonline.com/20050144463.pdf. US Patent Application: US 2005/0144463, Filing Date: 2003-03-18; Publication Date: 2005-06-30.

[149] Luis F. G. Sarmenta. Bayanihan: Web-based volunteer computing using java. In *Worldwide Computing and Its Applications – WWCA '98, 2nd International Conference*, pages 444–461. Springer-Verlag LNCS 1368, 1998. URL citeseer.ist.psu.edu/sarmenta98bayanihan.html.

[150] Bruce Schneier. Description of a new variable-length key, 64-bit block cipher (Blowfish). In *Fast Software Encryption, Cambridge Security Workshop Proceedings (December 1993)*, pages 191–204. Springer-Verlag, 1994. URL http://www.schneier.com/paper-blowfish-fse.html.

[151] SEEGRID. South Eastern European Grid-enabled eInfrastructure Development, 2004. URL http://www.see-grid.org/.

[152] E. Seidel, G. Allen, A. Merzky, and J. Nabrzyski. Gridlab: A grid application toolkit and testbed. In *Future Generation Computer Systems*, pages 1143– 1153, 2002.

[153] Shibboleth. Shibboleth architecture technical overview, June 2005. URL http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-tech-overview-latest.pdf.

[154] Michael Shirts and Vijay S. Pande. Screen savers of the world unite!, 2000. URL http://folding.stanford.edu/papers/SPScience2000.pdf.

[155] Olin Shivers. *SRFI 1: List Library*, October 1999. URL http://srfi.schemers.org/srfi-1/srfi-1.html.

[156] Frank Siebenlist, Von Welch, Steven Tuecke, Ian Foster, Nataraj Nagaratnam, Philippe Janson, John Dayka, and Anthony Nadalin. Ogsa security roadmap, July 2002. URL https://forge.gridforum.org/sf/go/doc6883?nav=1.

[157] Dorai Sitaram. *pregexp: Portable Regular Expressions for Scheme and Common Lisp*, May 2005. URL `http://www.ccs.neu.edu/home/dorai/pregexp/pregexp.html`.

[158] David F. Snelling, Sven van den Berghe, and Vivian Qian Li. Explicit trust delegation: Security for dynamic grids. *Fujitsu Scientific and Technical Journal*, 40(2):282–294, December 2004.

[159] Martijn Steenbakkers. Guide to lcas version 1.1.16, September 2003. URL `http://www.dutchgrid.nl/DataGrid/wp4/lcas/edg-lcas-1.1/`.

[160] Martijn Steenbakkers. Guide to lcmaps version 0.0.23, September 2003. URL `http://www.dutchgrid.nl/DataGrid/wp4/lcmaps/edg-lcmaps_gcc3_2_2-0.0.23/`.

[161] Daniel Stenberg et al. *The LibCurl Library*, 2006. URL `http://curl.haxx.se/libcurl/`.

[162] Heinz Stockinger et al. Defining the grid: A snapshot on the current view. To appear in Journal of Supercomputing, 2006.

[163] Maik Stohn, Marco Juliano, and Johannes Nicolai. *Novosec Bouncy Castle Extensions*. Novosec AG, October 2004. URL `http://novosec-bc-ext.sourceforge.net/`.

[164] Sun Microsystems. *Java 2 Platform, Standard Edition 5.0 API Specification*, 2004. URL `http://java.sun.com/j2se/1.5.0/docs/api/index.html`.

[165] SWITCH. Swiss education and research network authentication and authorization infrastructure, 2006. URL `http://www.switch.ch/aai/`.

[166] Miklos Szeredi. FUSE – filesystem in userspace, 2005. URL `http://fuse.sourceforge.net`.

[167] TAGPMA. *The Americas Grid Policy Management Authority*, 2006. URL `http://www.tagpma.org/`.

[168] Yoshio Tanaka. *APGridPMA CA Audit Checklist*, June 2005. URL `http://forge.gridforum.org/projects/caops-wg/document/GGF14-Audit-Checklist/`.

[169] J. A. Templon and D. A. Groep. VO server information, January 2002. URL `http://marianne.in2p3.fr/datagrid/documentation/ldap-doc.pdf`.

[170] Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice: the Condor experience. *Concurrency – Practice and Experience*, 17(2-4):323–356, 2005.

[171] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn. *XML Schema Part 1: Structures (Second Edition)*, October 2004. URL `http://www.w3.org/TR/xmlschema-1/`. W3C Recommendation.

[172] Tom Tromey. *Guide to GNU gcj*, 2005. URL `http://gcc.gnu.org/onlinedocs/gcj.pdf`.

[173] TrustCoM. Trustcom, 2005.

[174] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. *Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile*, June 2004. URL `ftp://ftp.isi.edu/in-notes/rfc3820.txt`. RFC 3820.

[175] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence. *AAA Authorization Framework*, August 2000. URL `ftp://ftp.isi.edu/in-notes/rfc2904.txt`. RFC 2904.

[176] X. Wang and H. Yu. How to break md5 and other hash functions. In *Advances in Cryptology – EUROCRYPT 2005*, Lecture Notes in Computer Science, 2005. LNCS 3494.

[177] William A. Wulf, Chenxi Wang, and Darrell Kienzle. A new model of security for distributed systems. Technical Report CS-95-34, University of Virginia, August 1995. URL `http://legion.virginia.edu/papers/CS-95-34.pdf`.

[178] Lotfi Asker Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, June 1965.

[179] Jeffrey Zeldman. *Designing with Web Standards*. New Riders, 2003.

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

Author's Publications

[1] J. Astalos, R. Cecchini, B.A. Coghlan, R.D. Cowles, U. Epting, T.J. Genovese, J. Gomes, D. Groep, M. Gug, A.B. Hanushevsky, M. Helm, J.G. Jensen, C. Kanellopoulos, D.P. Kelsey, R. Marco, I. Neilson, S. Nicoud, D. O'Callaghan, D. Quesnel, I. Schaeffner, L. Shamardin, D. Skow, M. Sova, A. Wäänänen, P. Wolniewicz, and W. Xing. International grid CA interworking, peer review and policy management through the European DataGrid certification authority coordination group. In Peter M.A. Sloot, Alfons G. Hoekstra, Thierry Priol, Alexander Reinefeld, and Marian Bubak, editors, *Advances in Grid Computing – EGC 2005*, LNCS3470, pages 275–285, Amsterdam, The Netherlands, February 2005. Springer. ISBN 3-540-26918-5. *The author of this thesis coordinated the editing process for this paper, and presented it at the European Grid Conference in 2005.*

[2] R. Byrom, B. Coghlan, A. Cooke, R. Cordenonsi, L. Cornwall, M. Craig, A. Djaoui, S. Fisher, A. Gray, S. Hicks, S. Kenny, J. Leake, O. Lyttleton, J. Magowan, R. Middleton, W. Nutt, D. O'Callaghan, N. Podhorszki, P. Taylor, J. Walk, and A. Wilson. Production services for information and monitoring in the grid. In *Allhands Meeting, Nottingham, September 2004*, 2004. Precis.

[3] R. Byrom, B. Coghlan, A. Cooke, R. Cordenonsi, L. Cornwall, A. Datta, A. Djaoui, L. Field, S. Fisher, S. Hicks, S. Kenny, J. Magowan, W. Nutt, D. O'Callaghan, M. Oever, N. Podhorszki, J. Ryan, M. Soni, P. Taylor, A. Wilson, and X. Zhu. The canonical producer: An instrument monitoring component of the relational grid monitoring architecture (R-GMA). In *Proc. ISPDC 2004*, pages 232–237. IEEE Computer Society, July 2004.

[4] R. Byrom, B. Coghlan, A. Cooke, R. Cordenonsi, L. Cornwall, M. Craig, A. Djaoui, A. Duncan, S. Fisher, A. Gray, S. Hicks, S. Kenny, J. Leake, O. Lyttleton, J. Magowan, R. Middleton, W. Nutt, D. O'Callaghan, N. Podhorszki, P. Taylor, J. Walk, and A. Wilson. Fault tolerance in the R-GMA information and monitoring system. In Peter M.A. Sloot, Alfons G. Hoekstra, Thierry Priol, Alexander Reinefeld, and Marian Bubak, editors, *Advances in Grid Computing - EGC 2005*, LNCS3470, Amsterdam, The Netherlands, February 2005. Springer. ISBN 3-540-26918-5.

[5] R. Byrom, B. Coghlan, A. Cooke, R. Cordenonsi, L. Cornwall, A. Datta, A. Djaoui, L. Field, S. Fisher, S. Hicks, S. Kenny, J. Magowan, W. Nutt, D. O'Callaghan, M. Oever, N. Podhorszki, J. Ryan, M. Soni, P. Taylor, A. Wilson, and X. Zhu. The canonical producer: an instrument monitoring component of the relational grid monitoring architecture (R-GMA). *Scientific Programming*, 13(2):151–158, October 2005.

[6] Rob Byrom, Brian Coghlan, Andrew Cooke, Roney Cordenonsi, Linda Cornwall, Ari Datta, Abdeslem Djaoui, Laurence Field, Steve Fisher, Steve Hicks, Stuart Kenny, James Magowan, Werner Nutt, David O'Callaghan, Manfred Oevers, Norbert Podhorszki, John Ryan Manish Soni, Paul Taylor, Antony Wilson, and Xiaomei Zhu. R-GMA: A relational grid information and monitoring system. In *Proc. 2nd Grid Workshop, Cracow, December 2002*, December 2002.

[7] Rob Byrom, Brian Coghlan, Andrew Cooke, Roney Cordenonsi, Linda Cornwall, Ari Datta, Abdeslem Djaoui, Laurence Field, Steve Fisher, Steve Hicks, Stuart Kenny, James Magowan, Werner Nutt, David O'Callaghan, Manfred Oever, Norbert Podhorszki, John Ryan, Manish Soni, Paul Taylor, Antony Wilson, and Xiaomei Zhu. *SANTA-G: and instrument monitoring framework using the Relational Grid Monitoring Architecture (R-GMA)*, chapter CrossGrid. LNCS. Springer-Verlag, 2005.

[8] S. Childs, B.A. Coghlan, D. O'Callaghan, G. Quigley, and J. Walsh. Grid testing using virtual machines. In Marian Bubak, Michal Turala, and Kazimierz Wiatr, editors, *Proc. Cracow Grid Workshop (CGW'04)*, pages 371–378, Cracow, Poland, December 2004. Academic Computer Centre CYFRONET AGH. ISBN 83-915141-4-5.

[9] S. Childs, B. Coghlan, D. O'Callaghan, G. Quigley, and J. Walsh. A single-computer grid gateway using virtual machines. In *Proc. AINA 2005*, pages 761–770, Taiwan, March 2005. IEEE Computer Society.

[10] S. Childs, B. Coghlan, D. O'Callaghan, G. Quigley, and J. Walsh. Centralised fabric management for a national grid infrastructure. In *Cracow Grid Workshop (CGW'05)*, Cracow, Poland, November 2005.

[11] S. Childs, B.A. Coghlan, D. O'Callaghan, G. Quigley, and J. Walsh. Deployment of grid gateways using virtual machines. In Peter M.A. Sloot, Alfons G. Hoekstra, Thierry Priol, Alexander Reinefeld, and Marian Bubak, editors, *Advances in Grid Computing - EGC 2005*, LNCS3470, Amsterdam, The Netherlands, February 2005. Springer. ISBN 3-540-26918-5.

[12] S. Childs, B. Coghlan, J. Walsh, D. O'Callaghan, G. Quigley, and E. Kenny. virtual testgrid, or how to replicate a national grid. In *Proc. ExpGrid workshop at The 15th IEEE*

International Symposium on High Performance Distributed Computing (HPDC2006), Paris, France, June 2006.

[13] B. Coghlan, G. Quigley, S. Maad, J. Ryan, E. Kenny, and D. O.Callaghan. A transparent grid filesystem. In *Workshop on state-of-the-art in scientific and parallel computing. Umea, Sweden, June 18–21, 2006*, 2006.

[14] B.A. Coghlan, J. Walsh, G. Quigley, D. O.Callaghan, S. Childs, and E. Kenny. Transactional grid deployment. In Marian Bubak, Michal Turala, and Kazimierz Wiatr, editors, *Proc. Crakow Grid Workshop (CGW'04)*, pages 363–370, Cracow, Poland, December 2004. Academic Computer Centre CYFRONET AGH. ISBN 83-915141-4-5.

[15] B.A. Coghlan, J. Walsh, and D. O'Callaghan. Grid-ireland deployment architecture. In Peter M.A. Sloot, Alfons G. Hoekstra, Thierry Priol, Alexander Reinefeld, and Marian Bubak, editors, *Advances in Grid Computing - EGC 2005*, LNCS3470, Amsterdam, The Netherlands, February 2005. Springer. ISBN 3-540-26918-5.

[16] B.A. Coghlan, J. Walsh, G. Quigley, D. O'Callaghan, S. Childs, and E. Kenny. Principles of transactional grid deployment. In Peter M.A. Sloot, Alfons G. Hoekstra, Thierry Priol, Alexander Reinefeld, and Marian Bubak, editors, *Advances in Grid Computing - EGC 2005*, LNCS3470, pages 88–97, Amsterdam, The Netherlands, February 2005. Springer. ISBN 3-540-26918-5.

[17] A. Cooke, A. Gray, L. Ma, W. Nutt, J. Magowan, M. Oevers, P. Taylor, R. Byrom, L. Field, S. Hicks, J. Leake, M. Soni, A. Wilson, R. Cordenonsi, L. Cornwall, A. Djaoui, S. Fisher, N. Podhorszki, B. Coghlan, S. Kenny, and D. O'Callaghan. R-GMA: An information integration system for grid monitoring. In *Proc.Int.Conf. Cooperative Information Systems (CoopIS'03)*, Catania, Sicily, November 2003.

[18] A. Cooke, W. Nutt, J. Magowan, P. Taylor, J. Leake, R. Byrom, L. Field, S. Hicks, M. Soni, A. Wilson, R. Cordenonsi, L. Cornwall, A. Djaoui, S. Fisher, N. Podhorszki, B. Coghlan, S. Kenny, D. O.Callaghan, and J. Ryan. Relational grid monitoring architecture (R-GMA). In *Allhands Meeting, Sheffield, September 2003*, 2003.

[19] Andrew Cooke, Werner Nutt, James Magowan, Manfred Oevers, Paul Taylor, Ari Datta, Roney Cordenonsi, Rob Byrom, Laurence Field, Steve Hicks, Manish Soni, Antony Wilson, Xiaomei Zhu, Linda Cornwall, Abdeslem Djaoui, Steve Fisher, Norbert Podhorszki, Brian Coghlan, Stuart Kenny, David O'Callaghan, and John Ryan. R-GMA: First results after deployment. In *Proc. CHEP'03*, La Jolla, California, March 2003.

[20] Andrew W. Cooke, Alasdair J. G. Gray, Werner Nutt, James Magowan, Manfred Oevers, Paul Taylor, Roney Cordenonsi, Rob Byrom, Linda Cornwall, Abdeslem Djaoui, Laurence

Field, Steve Fisher, Steve Hicks, Jason Leake, Robin Middleton, Antony J. Wilson, Xiaomei Zhu, Norbert Podhorszki, Brian A. Coghlan, Stuart Kenny, David O'Callaghan, and John Ryan. The relational grid monitoring architecture: Mediating information about the grid. *Journal of Grid Computing*, 2(4):323–339, December 2004.

[21] L.A. Cornwall, J. Jensen, D.P. Kelsey, A. Frohner, D. Kouril, F. Bonnassieux, S. Nicoud, K. Lorentey, J. Hahkala, M. Silander, R. Cecchini, V. Ciaschini, L. dell'Agnello, F. Spataro, D. O'Callaghan, O. Mulmo, G.-L. Volpato, D. Groep, M. Steenbakkers, and A. McNab. Authentication and authorization mechanisms for multi-domain grid environments. *Journal of Grid Computing*, 2:301–311, December 2004.

[22] S. Fisher, A. Cooke, A. Djaoui, S. Hicks, R. Middleton, J. Walk, A. Wilson, N. Podhorszki, B. Coghlan, S. Kenny, O. Lyttleton, D. O'Callaghan, A. Gray, J. Leake, W. Nutt, J. Magowan, P. Taylor, R. Cordenonsi, R. Byrom, L. Cornwall, and M. Craig. Information and monitoring services within a grid environment. In *Proc. CHEP'04*, Interlaken, Switzerland, March 2004.

[23] E. Kenny, B. Coghlan, G. Tsouloupas, M. Dikaiakos, J. Walsh, S. Childs, D. O'Callaghan, and G. Quigley. Heterogeneous grid computing: Issues and early benchmarks. In *Proc. ICCS 2005*, volume Part III of *LNCS3516*, pages 870–874, Atlanta, USA, May 2005.

[24] E. Kenny, B. Coghlan, J. Walsh, S. Childs, D. O'Callaghan, and G. Quigley. Autobuilding multiple ports of computing nodes for grid computing. In *Cracow Grid Workshop (CGW'05)*, Cracow, Poland, November 2005.

[25] E. Kenny, B.A. Coghlan, J. Walsh, S. Childs, D. O'Callaghan, and G. Quigley. Heterogeneity of computing nodes for grid computing. In Peter M.A. Sloot, Alfons G. Hoekstra, Thierry Priol, Alexander Reinefeld, and Marian Bubak, editors, *Advances in Grid Computing - EGC 2005*, LNCS3470, pages 401–413, Amsterdam, The Netherlands, February 2005. Springer. ISBN 3-540-26918-5.

[26] Eamonn Kenny, Brian A. Coghlan, John Walsh, Stephen Childs, David O'Callaghan, and Geoff Quigley. Testing heterogeneous computing nodes for grid computing. In Marian Bubak, Michal Turala, and Kazimierz Wiatr, editors, *Proc. Cracow Grid Workshop (CGW04)*, Cracow, Poland, December 2004. Academic Computer Centre CYFRONET AGH. ISBN 83-915141-4-5.

[27] Soha Maad, Brian Coghlan, Geoff Quigley, John Ryan, Eamonn Kenny, and David O'Callaghan. Towards a complete grid filesystem functionality. *Future Generation Computer Systems*, Special issue on Data Analysis, Access and Management on Grids, 2006. To appear.

[28] David O'Callaghan and Brian Coghlan. Bridging secure WebCom and European DataGrid security for multiple VOs over multiple grids. In *Proc. ISPDC 2004*, pages 225–231, Cork, Ireland, July 2004. IEEE Computer Society.

[29] David O'Callaghan and Brian Coghlan. On-demand trust evaluation. In *7th IEEE/ACM International Conference on Grid Computing (Grid 2006)*, September 2006.

[30] G. Pierantoni, O. Lyttleton, D. O'Callaghan, G. Quigley, E. Kenny, and B. Coghlan. Multi-grid and multi-vo job submission based on a unified computational model. In *Cracow Grid Workshop (CGW'05)*, Cracow, Poland, November 2005.

[31] G. Pierantoni, O. Lyttleton, D. O'Callaghan, G. Quigley, and E. Kenny. Interoperability using a Metagrid architecture. In *Proc. ExpGrid workshop at The 15th IEEE International Symposium on High Performance Distributed Computing (HPDC2006)*, Paris, France, June 2006.

Appendix A

Glossary

ABAC	Attribute-Based Access Control
AES	Advanced Encryption Standard
AJO	Active Job Object
APGrid PMA	Asia-Pacific Grid Policy Management Authority
CA	Certification Authority
CACG	Certification Authority Coordination Group; group within **EDG**
CAS	Community Authorization Service
CCA	Credential Creation Authority
CG	Condensed Graph
CP	Certificate Policy
CP/CPS	Combined **CP** and **CPS**
CPS	Certification Practice Statement
CRL	Certification Revocation List
DAC	Discretionary Access Control
DAG	Directed Acyclic Graph
DEISA	Distributed European Infrastructure for Supercomputing Applications
DES	Data Encryption Standard
DN	Distinguished Name
DNV	Det Norske Veritas
DSA	Digital Signature Algorithm
EDG	European DataGrid; the project and its software
EE	End Entity: a person, host, etc. that is the subject of a **CA** certificate
EGEE	Enabling Grids for E-science
ETSI	European Telecommunications Standards Institute

EUGridPMA	European Policy Management Authority for Grid Authentication in e-Science
FUSE	Filesystem in Userspace
GACL	Grid Access Control List
GGF	Global Grid Forum; now **OGF**
GIIS	Grid Index Information Service
GNU	**GNU**'s Not Unix
GRAM	Globus Resource Allocation Manager
GRIS	Grid Resource Information Service
GSI	Grid Security Infrastructure
HPC	High-Performance Computing
HSM	Hardware Security Module
HTML	Hypertext Markup Language
IETF	Internet Engineering Task Force
IGTF	International Grid Trust Federation: a federation of grid **PMA**s
KCA	Kerberized Certification Authority
KNCA	KeyNote Credential Authority
LCAS	Local Centre Authorisation Service
LCG	**LHC** Computing Grid
LCMAPS	Local Credential Mapping Service
LDAP	Lightweight Directory Access Protocol
LHC	Large Hadron Collider
LOID	Legion Object Identifier
MAC	Mandatory Access Control
MD5	Message Digest 5
MDS	Monitoring and Discovery Service
MSS	Metagrid Submission Service
MSX	Metagrid Security Exchange
NSI	New Security Infrastructure
OCSP	Online Certificate Status Protocol
OGF	Open Grid Forum; formerly **GGF**
OGRO	Open Grid OCSP library
OGSA	Open Grid Service Architecture
OGSA-DAI	**OGSA** Data Access and Integration
OID	Object Identifier
PDP	Policy Decision Point
PEP	Policy Enforcement Point

PERMIS	Privilege and Role Management Infrastructure Standards
PKI	Public Key Infrastructure
PKIS	**PKI** server in **NSI**
PMA	Policy Management Authority (for grid authentication)
RA	Registration Authority (of a **CA**, etc.)
RBAC	Role-Based Access Control
R-GMA	Relational Grid Monitoring Architecture
RLS	Replica Location Service
RP	Relying Party (of a **CA**, etc.)
RSA	Rivest, Shamir & Adleman, the inventors of this encryption algorithm
SAX	Simple **API** for **XML**
SHA-1	Secure Hash Algorithm – 1
SHTML	**HTML** in **SXML**
SRFI	Scheme Request For Implementation
SSAX	An **SXML SAX** parser
SSL	Secure Sockets Layer
SXML	**XML** as S-expressions
TACAR	**TERENA** Academic **CA** Repository
TAGPMA	The Americas Grid Policy Management Authority
TERENA	Trans-European Research and Education Networking Association
TLS	Transport Layer Security
UNICORE	Uniform Interface to Computing Resources
VA	Validation Authority
VDM	Vienna Development Method
VDM♣	The Irish School of the **VDM**
VO	Virtual Organisation
VOMS	Virtual Organisation Membership Service
VSC	Virtual Smart Card
WSDL	Web Services Description Language
X.509	an ITU-T standard for **PKI**
X-KISS	**XML** Key Information Service Specification
X-KMS	**XML** Key Management Specification
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

Appendix B

VDM♣ Notation

Presented here is a brief informal description of some of the notation of the Irish School of the Vienna Development Method used in models of the trust evaluation system. The notation was introduced in [112] and later presented in [31]. Further information is available from `https://www.cs.tcd.ie/Andrew.Butterfield/IrishVDM/`.

Domain	a domain set
$\mathbb{P}\mathit{Domain}$	The powerset of Domain The set of all subsets of Domain, written $\mathcal{P}$ in other notations
$\mathit{Domain}^\star$	Sequence of Domain The type of all sequences of elements of Domain
$\mathbb{B}$	The boolean set
$\mathbb{R}$	The set of real numbers
$A \rightarrow B$	A function from A to B
$A \rightarrow B \rightarrow C$	A function from A to B to C this is equivalent to $A \times B \rightarrow C$ but emphasises the possibility of currying
Op	an operator
Op : $A \rightarrow B$	Op is a function $A \rightarrow B$
Op(a) $\widehat{=} \ldots$	Op is defined as …. the type of parameter a is indicated in the signature of Op as A

$A \overset{m}{\to} B$	a map from A to B
$A \dagger \{x_i \mapsto y_i\}$	map override The maplet in A with domain x_i is overridden with a range y_i
$\mathbb{P} f S$	Applying a function f to a set S with the set functor $= \{\, f(s_1), \ldots, f(s_n) \,\}$ The function f is applied to each element of S
$f^{\star} S$	Applying a function f to a sequence S $= \langle f(s_1), \ldots, f(s_n) \rangle$
$(f \overset{m}{\to} g)A$	Applying a pair of functions f and g to a map A with the map functor
$^{\star}/S$	Reducing a set with binary operation $^{\star}/\emptyset = \mathbb{I}_{\star} \quad {}^{\star}/s_1 \sqcup S = s_1 \star {}^{\star}/S$
π_i	projecting the ith element from a tuple

Projection

The projection function π_i takes the ith element from a tuple.

$$\pi_i : X_1 \times \ldots \times X_i \times \ldots \times X_n \to X_i$$
$$\pi_i(x_1, \ldots, x_i, \ldots, x_n) \mathrel{\widehat{=}} x_i$$

This function can be mapped over a set of tuples with the set functor, $\mathbb{P}$, to gather all ith values.

$$\begin{aligned} \mathbb{P}\pi_i(\emptyset) &\mathrel{\widehat{=}} \emptyset \\ \mathbb{P}\pi_i(\{\, (x_1, \ldots, x_i, \ldots, x_n) \,\} \sqcup X) &\mathrel{\widehat{=}} \{\, \pi_i(x_1, \ldots, x_i, \ldots, x_n) \,\} \sqcup \mathbb{P}\pi_i X \\ &\mathrel{\widehat{=}} \{\, x_i \,\} \sqcup \mathbb{P}\pi_i X \end{aligned}$$

Appendix C

Rulesets

C.1 IGTF Traditional CA

The ruleset below is based on the IGTF profile for *Traditional X.509 Public Key Certification Authorities with secured infrastructure*. The rules in this ruleset return Boolean values. The rules include a number of extra elements for **weight**, based on the RFC 2119 definitions of Must, Should, etc. [27], **refs**, which are references to source documents, **type**, which specifies the type (e.g., global, historical) of certain rules, and **reason** which gives the reasoning behind the rule. These elements are not currently used by the evaluation engine.

```
(ruleset
  (name "IGTF Traditional Ruleset")
  (description "Ruleset based on IGTF 'Profile for Traditional X.509 Public Key
      Certification Authorities with secured infrastructure' version 4.0")
  (fold
    (null-value #f)
    (combiner (lambda (a b) (and a b))))

  (rules
    (rule
      (name "Community Served")
      (description "The CA should serve a community covering a country, large
          region, or international organization.")
      (params ((area (community area-served))))
      (func (member area '(country large-region international-organization)))
      (weight should)
      (refs (trad 2) (doc 1)))

    (rule
      (name "Sponsoring Organization Type")
      (description "Is sponsoring organisation of a suitable type?")
    (params (sponsoring-organization-type))
    (func (member sponsoring-organization-type '(university research-institute
        nren)))
    (weight should)
    (reason "It is expected that the CAs will be operated as a long-term
        commitment by institutions or organisations rather than being bound to
        specific projects.")
    (refs (trad 2) (doc 3 f)))

  (rule
    (name "Unique End-Entity")
    (description "Is every subject DN linked to one and only one entity during
        the CA's lifetime?")
    (params ((unique (ee cert subject-dn unique-linking))))
    (func (equal? unique #t))
    (weight must)
    (refs (trad 3)  (doc 5 r)))

  (rule
    (name "RA Verification Methods")
    (description "Is RA using acceptable verification methods")
    (params ((methods (ras verification-methods))))
    (func (forall
            (lambda (m)(member m '(check-ID compare-ID-with-logs check-other-IdP)
                ))
            methods))
    (weight must)
    (refs (trad 3) (doc 6 f)))

  (rule
    (name "CA Verification Methods")
    (description "Is CA using acceptable verification methods")
    (params (verification-methods))
    (func (forall
            (lambda (m) (member m '(compare-request-with-logs)))
            verification-methods ))
    (weight must)
    (refs (trad 3) (doc 7 f)))

  (rule
    (name "Certificate Sharing")
    (description "Certificates must not be shared between end-entities")
    (params ((sharing (ee cert is-sharing-allowed))))
    (func (eq? sharing #f))
    (weight must-not)
    (refs (trad 3) (doc 8 r)))
```

```
(rule
  (name "RA Personal Validation")
  (description "How does RA validate personal requests?")
  (params ((val (ras personal-validation))))
  (func (equal? val 'face-to-face))
  (weight should)
  (refs (trad 3.1) (doc 9 r)))

(rule
  (name "Identification Accepted")
  (description "What forms of personal identification/entitlement are accepted
      by the RA?")
  (params ((id (ras acceptable-id)))
  (func (member id '(government-photo-id institute-photo-id
      institute-official-document)))
  (weight should)
  (refs (trad 3.1) (doc 10 f)))

(rule
  (name "RA Server Validation")
  (description "How does RA validate server (host/service) requests?")
  (params ((val (ras server-validation)))
  (func (member val
          '(personal-validation check-requestor-cert whois dns-lookup)) )
  (weight should)
  (refs (trad 3.1) (doc 11 r)))

(rule
  (name "Association of CSR")
  (description "Does RA validate association of CSR?")
  (params ((val (ras csr-association-validation)))
  (func (equal? val #t))
  (weight should)
  (refs (trad 3.1) (doc 12 r)))

(rule
  (name "RA Archival")
  (description "What does RA archive?")
  (params ((ra-archive (ras archive)))
  (func (equal? ra-archive '(requests confirmations)))
  (weight must)
  (refs (trad 3.1) (doc 13 r)))

(rule
    (name "RA/CA Secure Comms")
    (description "How does RA communicate securely with CA?")
    (params ((methods (ras secure-comms-methods))))
    (func (member-powerset methods '(signed-emails ssl-web)))
    (weight must)
    (refs (trad 3.1) (doc 15 f)))

  (rule
    (name "CA dedicated")
    (description "Is CA a dedicated machine?")
    (params ((dedicated (system is-dedicated))))
    (func (equal? dedicated #t))
    (weight must)
    (refs (trad 4) (doc 16 r)))

  (rule
    (name "CA other services")
    (description "Is CA running other services?")
    (params ((other (system runs-other-services))))
    (func (equal? other #f))
    (weight must)
    (refs (trad 4) (doc 16 r) ))

  (rule
    (name "CA Security")
    (description "Is CA machine in a secure environment, with controlled access
        restricted to trained personnel?")
    (params ((sec (system secure-environment))
             (ctl (system ca-controlled-access))
             (per (system trained-personnel)))
    (func (and sec ctl per))
    (weight must)
    (refs (trad 4) (doc 16 r))
    (comment "Is secure environment documented/auditable?"))

  (rule
    (name "CA Network")
    (description "If CA does not have HSM it must be disconnected")
    (params ((net (system connected-to-network))
             (hsm (system has-hsm))))
    (func (or (not new) hsm))
    (weight must)
    (refs (trad 4) (doc 17 r))
    (comment "Consider network protection w/ HSM"))

  (rule
    (name "CA Key Length")
    (description "Minimum CA key length")
    (params ((size (key size))))
    (func (>= size 2048))
    (weight must)
    (refs (trad 4) (doc 18 r)))

  (rule
    (name "CA Lifetime")
    (description "CA Lifetime ")
    (params (issues-ee-certs
              (lifetime (cert lifetime))
              (ee-max (ee cert max-lifetime))))
    (func (and issues-ee-certs (>= lifetime (* 2 ee-max)) (<= lifetime 20)))
    (weight must)
    (refs (trad 4) (doc 19 r)))

  (rule
    (name "CA private key protection")
    (description "How is the CA private key protected")
    (params ((hsm (system has-hsm))
              (minl (key passphrase min-length))))
    (func (or hsm (>= minl 15)))
    (weight must)
    (refs (trad 4) (doc 20 r)))

  (rule
    (name "CA private key backup")
    (description "CA backs up private key")
    (params ((backup (key has-backup))))
    (func (equal? backup #t))
    (weight must)
    (refs (trad 4) (doc 21 r)))

  (rule
    (name "CA has CP/CPS")
    (description "CA has CP/CPS documents")
    (params ((has (cp-cps has-cp-cps))))
    (func (equal? has #t))
    (weight must)
    (refs (trad "4.1") (doc 22 r)))

  (rule
    (name "CA CP/CPS has unique oid")
    (description "CA CP/CPS documents have unique OID")
    (params ((unique (cp-cps has-unique-oid))))
  (func (equal? unique #t))
  (weight must)
  (refs (trad "4.1") (doc 22 r)))

(rule
  (name "CA CP/CPS format")
  (description "CA CP/CPS is in a suitable format")
  (params ((format (cp-cps format))))
  (func (equal? format 'rfc3647))
  (weight should)
  (refs (trad "4.1") (doc 23 r)))

(rule
  (name "CA CP/CPS web versions")
  (description "All versions of CA's CP/CPS under which valid certs exist must
      be available on the web")
  (params ((available (cp-cps all-versions-available))))
  (func (equal? available #t))
  (weight must)
  (refs (trad "4.1" )(doc 25 r)))

(rule
  (name "CA publishes cert")
  (description "CA publishes X.509 certificate.")
  (params ((x509 (cert format is-x509))))
  (func (equal? x509 #t))
  (weight must)
  (refs (trad "4.2") (doc 26 r)))

(rule
  (name "CA cert critical extensions")
  (description "CA cert requires certain critical extensions")
  (params ((ku (cert key-usage has-extension))
           (ku-is-crit (cert key-usage is-critical))
           (bc (cert basic-constraints has-extension))
           (bc-is-crit (cert basic-constraints is-critical))))
  (func (and ku ku-is-crit bc bc-is-crit))
  (weight must)
  (type rule cert)
  (refs (trad "4.2") (doc 27 r)))

(rule
  (name "EE crypto data generation")
  (description "How is EE cryptographic material generated?")
  (params ((gen (ee crypto-data-generation))))
  (func (map (lambda (g) (member g '(applicant on-token))) gen))
```

```
    (weight shall)
    (refs (trad "4.2") (doc 28 r)))

  (rule
    (name "EE key length")
    (description "EE key greater than minimum length")
    (params (size (ee key size))
    (func (>= size 1024))
    (weight must)
    (refs (trad "4.2") (doc 30 r)))

  (rule
    (name "EE cert lifetime")
    (description "EE cert less than max lifetime")
    (params (life (ee cert max-lifetime))
    (func (<= life (+ 365 31)))
    (weight must)
    (refs (trad "4.2") (doc 31 r)))

  (rule
    (name "EE cert format")
    (description "EE cert in X.509v3 format")
    (params ((is-x509 (ee cert format is-x509))
             (version (ee cert format version))))
    (func (and is-x509 (equal? version 3)))
    (type cert)
    (weight must)
    (refs (trad "4.2") (doc 32 r)))

  (rule
    (name "EE cert rfc3280")
    (description "EE cert compliant with RFC3280")
    (params ((rfc3280 (ee cert format rfc3280-compliant))))
    (func (equal? rfc3280 #t))
    (weight must)
    (refs (trad "4.2") (doc 33 r)))

  (rule
    (name "EE cert policy OID")
    (description "EE cert use of policy OID")
    (params ((has (ee cert policy-oid has-policy-oid))
            (only (ee cert policy-oid policy-oid-only))))
    (func (and has only))
    (weight must)
    (refs (trad "4.2") (doc 34 r)))
```

```
(rule
  (name "EE cert CRL distribution points")
  (description "EE cert includes CRL distribution points")
  (params ((has (ee cert crl-distribution-points has-extension))
           (value (ee cert crl-distribution-points value))))
  (func (and has (any (map is-http-url value))))
  (weight must)
  (refs (trad "4.2") (doc 35 r)))

(rule
  (name "EE Cert Key Usage")
  (description "EE cert key usage extension")
  (params ((has (ee cert key-usage has-extension))
           (crit (ee cert key-usage is-critical))))
  (func (and has crit))
  (weight must)
  (type rule cert)
  (refs (trad "4.2") (doc 36 r)))

(rule
  (name "EE Cert Basic Constraints")
  (description "EE cert basic constraints extension")
  (params ((has-ext (ee cert basic-constraints has-extension))
           (is-crit (ee cert basic-constraints is-critical))
           (value   (ee cert basic-constraints value))))
  (func (and (equal? has-ext #t)
             (if has-ext
               (and is-crit (equal? value CA:false))
              #t)))
  (weight should)
  (type rule cert)
  (refs (trad "4.2") (doc 37 r)))

(rule
  (name "EE Cert AuthorityInfoAccess")
  (description "EE Cert AuthorityInfoAccess conditions")
  (params ((ca-ocsp (ocsp-service is-production-service))
           (has-ext (ee cert authority-info-access has-ext))
           (value   (ee cert authority-info-access value))))
  (func (if ca-ocsp
          (and (equal? has-ext #t)
               (any (map is-uri value)))
          #t))
  (weight must)
  (type rule cert)
  (refs (trad "4.2") (doc 38 r)))
```

```
(rule
  (name "EE Cert Network Entity")
  (description "EE Cert for Network Entity contains FQDN")
  (params ((has-fqdn (ee cert network-entity has-fqdn))))
  (func (equal? has-fqdn #t))
  (weight shall)
  (type rule cert)
  (refs (trad "4.2") (doc 39 r)))

(rule
  (name "EE Cert common name")
  (description "If a commonName component is used as part of the subject DN, it
       should contain an appropriate presentation of the actual name of the
      end-entity")
  (params ((has-cn (ee cert subject-dn has-cn))
           (is-apt (ee cert subject-dn cn-is-appropriate-presentation-of-name))
              ))
  (func (if has-cn is-apt #t))
  (weight should)
  (refs (trad "4.2") (doc 40 r)))

(rule
  (name "CRL format")
  (description "CRL format and version")
  (params ((rfc3280 (crl rfc3280-compliant))
           (version (crl version))))
  (func (and rfc3280
             (member version '(1 2))))
  (weight must)
  (refs (trad "4.2") (doc 42 r)))

(rule
  (name "CA Message Digest")
  (description "CA uses secure/supported message digests")
  (params ((md (message-digest))))
  (func (and (not (equal? md md5))
             (member md '(sha1))))
  (weight must)
  (refs (trad "4.2") (doc 43 r)))

(rule
  (name "CA publishes CRL")
  (description "CA must publish a CRL")
  (params ((pub (crl is-published))))
  (func (equal? pub #t))
```

```
    (weight must)
    (refs (trad "4.3") (doc 44 r)))

  (rule
    (name "CA revocation reaction time")
    (description "CA revocation reaction time")
    (params ((time (crl max-reaction-time))))
    (func (<= time 1))
    (weight must)
    (refs (trad "4.3") (doc 45 r)))

  (rule
    (name "CA revocation CRL publication")
    (description "CRL must be issued immediately for a valid revocation request")
    (params ((del (crl lifetime-after-revocation))))
    (func (equal? del 0))
    (weight must)
    (refs (trad "4.3") (doc 45 r)))

  (rule
    (name "CA publication frequency")
    (description "CA publication frequency")
    (params ((del (crl lifetime-after-revocation))
             (maxl (crl lifetime))
             (minl (crl min-before-expiration))
             (ee (issues-ee-certs))))
    (func (if ee
            (and (<= maxl 30)
                 (>= minl 7)
                 (= del 0))
           #t))
    (weight must)
    (refs (trad "4.3") (doc 46 r)))

  (rule
    (name "CA publishes CRL to web")
    (description "CA must publish CRLs in repository accessible on web when
        issued")
    (params ((how (crl how-published))))
    (func (any (map is-http) how))
    (weight must)
    (refs (trad "4.3") (doc 47 r)))

  (rule
    (name "Revocation Requestors")
    (description "Who can request revocation?")
```

```
   (params ((reqs (revocation−requestors))))
   (func (equal? reqs '(ee ra ca)))
   (weight must)
   (comment "Also, others who ‘can sufficiently prove compromise or exposure of
       the associate private key.’")
   (refs (trad "4.3") (doc 49 r)))

 (rule
   (name "EE revocation duty")
   (description "If private key is lost or compromised, or data is no longer
       valid EE must request revocation.")
   (params ((r (ee revocation reasons))))
   (func (equal? r '(key−lost key−compromised data−invalid)))
   (weight must)
   (refs (trad "4.3") (doc 50 r)))

 (rule
   (name "CA passphrase backup")
   (description "CA passphrase must be securely backed up or equivalent")
   (params ((off (key passphrase backup offline))
            (sep (key passphrase backup seperate))
            (safe (key passphrase backup safe))))
   (func (and off sep safe))
   (weight must)
   (comment "Or equivalent documented secure procedure")
   (refs (trad "5") (doc 54 r)))

 (rule
   (name "CA publishes root")
   (description "CA must publish root cert or set of root certs leading to
       self-signed root")
   (params ((self−pub (publication  self−signed−root is−published))
            (path−pub (publication  root−path is−published))))
   (func (or self−pub path−pub))
   (weight must)
   (refs (trad "6") (doc 55 r)))

 (rule
   (name "CA publishes cert PEM URL")
   (description "CA must publish http or https URL of PEM-formatted CA
       certificate")
   (params ((http (publication pem−cert http−url))
            (https (publication pem−cert https−url))))
   (func (or http https))
   (weight must)
   (refs (trad "6") (doc 56 r)))
```

```
(rule
  (name "CA publishes CRL URL")
  (description "CA must publish a http URL of the PEM or DER formatted CRL")
  (params ((http-pem (publication pem-crl http-url))
           (http-der (publication der-crl http-url))))
  (func (or http-pem http-der))
  (weight must)
  (refs (trad "6") (doc 57 r)))

(rule
  (name "CA publishes information")
  (description "CA must publish a http or https URL of the web page of the CA
      for general information.")
  (params ((http (publication general-info http-url))
           (https (publication general-info https-url))))
  (func (or http https))
  (weight must)
  (refs (trad "6") (doc 58 r)))

(rule
  (name "CA publishes CP/CPS")
  (description "CA must publish the CP and/or CPS documents")
  (params ((cp (publication cp is-published))
           (cps (publication cps is-published))))
  (func (or cp cps))
  (weight must)
  (refs (trad "6") (doc 59 r)))

(rule
  (name "CA publishes email address")
  (description "CA must publish an official contact email address for inquiries
       and fault reporting.")
  (params ((email (publication contact-mechanism email-address))))
  (func (or email #f))
  (weight must)
  (refs (trad "6") (doc 60 r)))

(rule
  (name "CA publishes postal address")
  (description "CA must publish a physical or postal address")
  (params ((postal (publication contact-mechanism postal-address))))
  (func (or postal #f))
  (weight must)
  (refs (trad "6") (doc 61 r)))
```

```
(rule
  (name "CA trust validation")
  (description "CA should provide a means to validate the integrity of their
      root of trust")
  (params ((valid (publication integrity-validation means))))
  (func (or valid #f))
  (comment "To-do: hash/fingerprint?")
  (weight should)
  (refs (trad "6") (doc 62 r)))

(rule
  (name "CA root in repository")
  (description "CA shall provide their trust anchor to a trust anchor
      repository, specified by the accrediting PMA, via the method specified in
       the policy of the trust anchor repository.")
  (params ((in-repo (publication repository is-in-repository))
           (repo (ca-publication repository name))))
  (func (and in-repo
             (equal? repo 'tacar)))
  (weight shall)
  (refs (trad "6") (doc 63 r)))

(rule
  (name "CA allows redistribution")
  (description "CA must grant to the PMA and the Federation --- by virtue of
      its accreditation --- the right of unlimited re-distribution of this
      information")
  (params ((redis (publication allows-redistribution))))
  (func (or allows-redistribution #f))
  (weight must)
  (refs (trad "6") (doc 64 r)))

(rule
  (name "CA archives")
  (description "CA must record and archive information for auditing")
  (params ((reqs  (archive cert-requests))
           (certs (archive issued-certs))
           (revrs (archive revocation-requests))
           (crls  (archive issued-crls))
           (logs  (archive ca-system-logs))))
  (func (and reqs certs revrs crls logs))
  (weight must)
  (refs (trad "7") (doc 65 r)))

(rule
  (name "CA archive duration")
  (description "CA must keep records for at least three years")
  (params ((dur (archive duration))))
  (func (>= dur 3))
  (weight must)
  (refs (trad "7") (doc 66 r)))

(rule
  (name "CA auditable")
  (description "CA must make records available to external auditors")
  (params ((ava (archive available-to-external-auditors))))
  (func (or ava #f))
  (weight must)
  (refs (trad "7") (doc 67 r)))

(rule
  (name "CA accepts audits")
  (description "CA must accept being audited by other accredited CAs to verify
      compliance with its CP/CPS")
  (params ((allows (audit allows-pma-audit))))
  (func (or allows #f))
  (weight must)
  (refs (trad "7") (doc 68 r)))

(rule
  (name "CA operational audits")
  (description "CA should perform operational audits of the CA/RA staff at
      least once per year.")
  (params ((freq (audit internal-audits-per-year))))
  (func (>= freq 1))
  (weight should)
  (refs (trad "7") (doc 69 r)))

(rule
  (name "CA privacy policy")
  (description "CA must define a privacy and data release policy compliant with
       the relevant national legislation.")
  (params ((pol (privacy has-compliant-policy))))
  (func (or pol #f))
  (weight must)
  (refs (trad "8") (doc 70 r)))

(rule
  (name "CA subscriber identification")
  (description "CA is responsible for recording, at the time of validation,
      sufficient information regarding the subscribers to identify the
      subscribers.")
      (params ((id (privacy is−identification−recorded))))
      (func (or id #f))
      (weight "‘‘is responsible’’")
      (refs (trad "8") (doc 71 r)))

    (rule
      (name "CA information release")
      (description "CA is not required to release such information unless provided
          by a valid legal request according to national laws applicable to that CA
          .")
      (params ((req (privacy information−release))))
      (func (equal? req ’on−legal−request))
      (weight "‘‘not required ... unless’’")
      (refs (trad "8") (doc 72 r)))

    (rule
      (name "CA disaster recovery")
      (description "CA must have an adequate compromise and disaster recovery
          procedure.")
      (params ((com−proc (recovery has−compromise−recovery−procedure))
               (dis−proc (recovery has−disaster−recovery−procedure))))
      (func (and com−proc dis−proc))
      (weight must)
      (comment "Disclosure of policy: public (in CPS), release to PMA, discuss in
          PMA")
      (refs (trad "9")(doc 73 r)))

    (rule
      (name "EE key protection")
      (description "CA should make a reasonable effort to make sure that
          end-entities realize the importance of properly protecting their private
          data.")
      (params ((is−informed (ee key protection is−informed))
               (min−length (ee cert natural−person min−passphrase−length))))
      (func (and is−informed
                 (>= min−length 12)))
      (weight should)
      (refs (trad "9.1")(doc 75 r)(doc 76 r)))

    )
  )
```

C.2 IGTF Charter

This ruleset is based on the IGTF Charter. It overlaps somewhat with the ruleset for traditional CAs.

```
(ruleset
  (name "IGTF Charter Ruleset")
  (description "Ruleset based on International Grid Trust Federation charter
      document")

  (rules
    (rule
      (name "Unique End-Entity")
      (description "every identifier issued by an accredited authority, under any
          authentication profile, by any PMA, is associated with one and only one
          identity, within the scope of the federation.")
      (params ((unique (ee cert subject-DN unique-linking))))
      (func (= unique #t))
      (weight must)
      (ref (igtf 3) (doc 77 r)))

    (rule
      (name "Responsible for Non-natural Entity")
      (description "the entity must have an assigned natural person named as
          responsible for this entity.")
      (params ((responsible (ee cert network-entity responsible-person is-assigned)
          )))
      (func (= responsible #t))
      (weight must)
      (ref (igtf 3) (doc 78 r)))

    (rule
      (name "Transfer of responsibility")
      (description "responsible may assign or transfer responsibility for the
          entity to another natural person.")
      (params ((transfer (ee cert network-entity responsible-person may-transfer)))
          )
      (func (= transfer #t))
      (weight may)
      (ref (igtf 3) (doc 79 r)))

    (rule
      (name "Failure of responsibility")
      (description "every accredited authority must specify a mechanism for dealing
           with entities whose responsible person fails in their responsibilites
          for an entity without assigning a new responsible for this entity.")
  (params ((mechanism (ee cert network-entity responsible-person
      has-failure-mechanism))))
  (func (= mechanism #t))
  (weight must)
  (ref (igtf 3) (doc 80 r)))

(rule
  (name "Unique namespaces")
  (description "A specific subject namespace or set of subject namespaces is
      allocated to each authority. This name space must not overlap with any
      existing name space already assigned.")
  (params ((namespaces (namespaces))))
  (func (are-all-unique namespaces))
  (weight must)
  (type global)
  (ref (igtf "3.1") (doc 83 g)))

(rule
  (name "Namespace relationship")
  (description "The assignment of a name space to an authority will be
      according tocurrent best practices and the namespace will have a
      reasonable relationship to the scope of the authority.")
  (params ((is-reasonable (namespaces reasonable-relationship))))
  (func (= is-reasonable #t))
  (weight shall) ;will?
  (ref (igtf "3.1") (doc 83 g)))

(rule
  (name "Identity Vetting")
  (description "Each accredited authority must document its identity vetting
      rules and this document must be publicly available")
  (params ((is-documented (identity-vetting is-documented))
          (is-public (identity-vetting documentation is-public))))
  (func (and is-documented is-public))
  (weight must)
  (ref (igtf "3.2") (doc 84 r) (doc 85 r)))

(rule
  (name "Identity Vetting Records")
  (description "The issuing authority must ensure access to the records of the
      identity vetting process for at least three years")
  (params ((duration (identity-vetting records archive-duration))))
  (func (>= duration 3))
  (weight must)
  (ref (igtf "3.2") (doc 86 r)))
```

```
(rule
  (name "CA publishes contact information")
  (description "Each authority within the federation shall maintain at least
      one contact mechanism")
  (params ((has-contact (publication has-contact-mechanism))))
  (func (= has-contact #t))
  (weight shall)
  (ref (igtf "4.3") (doc 87 r)))

(rule
  (name "CA public contact")
  (description "Contact mechanism must allow for un-moderated access to report
      problems and faults regarding the authority by relying parties and the
      general public.")
  (params ((public (publication contact-mechanism is-public))))
  (func (= public #t))
  (weight must)
  (refs (igtf "4.3") (doc 87 r)))

(rule
  (name "CA publishes CP/CPS")
  (description "The accredited authority must disclose to the accrediting PMA
      and to the general public its documented policies and practices")
  (params ((cp (publication cp is-published))
           (cps (publication cps is-published))))
  (func (and cp cps))
  (weight must)
  (refs (igtf "4.3") (doc 88 r)))

(rule
  (name "CA documents security")
  (description "Each accredited authority will document its security mechanisms
       and this document must be available to all members of the IGTF
      federation (i.e. all members of all PMAs). This document will contain at
      least the software, network, server and physical security at the site. It
       must also describe the procedural controls, personnel security controls,
       and the life cycle management for security controls.")
  (params ((is-documented  (security-mechanisms is-documented))
           (is-ava-to-fed  (security-mechanisms documentation
               is-available-to-federation))
           (system-details (security-mechanisms documentation
               describes-system-details))
           (procedure-details (security-mechanisms documentation
               describes-procedure-details))))
  (func (and is-documented
             is-ava-to-fed
```

```
                system−details
                procedure−details))
    (weight must)
    (refs (igtf 5) (doc 89 r)))

  (rule
    (name "CA Auditable")
    (description "accredited authorities must be auditable, and all authorities
        must keep sufficient records for a period of at least three years")
    (params ((ava (archive available−to−external−auditors))
             (dur (archive duration))))
    (func (and ava
               (>= dur 3)))
    (weight must)
    (refs (igtf 9) (doc 90 r)))

  (rule
    (name "CA disaster recovery")
    (description "Each authority must define a compromise and disaster recover
        procedure, and be willing to discuss the procedure within the accrediting
         PMA.")
    (params ((com−proc (recovery has−compromise−recovery−procedure))
             (dis−proc (recovery has−disaster−recovery−procedure))))
    (func (and com−proc dis−proc))
    (weight must)
    (refs (igtf "11.2")(doc 91 r)))

  )
)
```

Appendix D

CA Description

This appendix presents a sample CA description.

```
(ca-description
  (name "CA@NUM@")
  (dn "/DC=ie/DC=grid/DC=te/DC=CA@NUM@/CN=CA@NUM@")
  (alias "CA@NUM@")
  (country "@NUM@land")
  (country-code "@NUM@")
  (security-level low)
  (CA-email "david.ocallaghan@cs.tcd.ie")
  (archive
    (available-to-external-auditors #t)
    (ca-system-logs #t)
    (cert-requests #t)
    (issued-certs #t)
    (issued-crls #t)
    (revocation-requests #t)
    (duration 3)) ;years
  (audit
    (allows-pma-audit #t)
    (internal-audits-per-year 1))
  (cert
    (basic-constraints
      (has-extension #t)
      (is-critical #t)
      (value "CA:true"))
    (key-usage
      (has-extension #t)
      (is-critical #t))
    (format
      (is-x509 #t)
      (version 3))
  (lifetime 3650)) ;days
(community
  (size 1000)
  (area-served country))
(cp-cps
  (has-cp-cps #t)
  (format rfc3647)
  (has-uniqiue-oid #t)
  (oid "1.3.6.1.1.1.1.@NUM@")
  (all-versions-available #t))

(issues-ee-certs #t)

(key
  (passphrase
    (min-length 15) ;characters
    (backup
      (offline #t)
      (safe #t)
      (seperate #t)))
  (size 2048)
  (has-backup #t))

(message-digest sha1)

(ocsp-service
  (runs-service #f)
  (is-production-service #f))


(privacy
  (has-compliant-policy #t)
  (information-release on-legal-reqiest)
  (is-identification-recorded #t))

(publication
  (allows-redistribution #t)
  (contact-mechanism
    (email-address "david.ocallaghan@cs.tcd.ie")
    (postal-address "c/o David O'Callaghan, Computer Science, Trinity College,
        Dublin 2"))
  (cp
    (is-published #t))
  (cps
    (is-published #t))
  (der-crl
    (http-url "http://ca@NUM@.te.grid.ie/crl.der"))
  (general-info
    (http-url "http://ca@NUM@.te.grid.ie/")
    (https-url "https://ca@NUM@.te.grid.ie/"))
  (integrity-validation )
  (pem-cert
    (http-url "http://ca@NUM@.te.grid.ie/cacert.pem")
    (https-url "https://ca@NUM@.te.grid.ie/cacert.pem"))
  (pem-crl
    (http-url "http://ca@NUM@.te.grid.ie/crl.pem"))
  (repository
    (is-in-repository #t)
    (name tacar))
  (root-path
    (is-published #f))
  (self-signed-root
    (is-published #t))
  )

(recovery
  (has-compromise-recovery-procedure #t)
  (has-disaster-recovery-procedure #t))

(revocation-requestors (ee ra ca))

(sponsoring-organization-type university)

(system
  (connected-to-network #f)
  (controlled-access #t)
  (is-dedicated #t)
  (runs-other-services #f)
  (secure-environment #t)
  (has-hsm #f)
  (trained-personnal #t))

(verification-methods (compare-request-with-logs))

(ee
  (cert
    (authority-info-access
      (has-extension #f))
    (basic-constraints
      (has-extension #t)
      (is-critical #t)
      (value CA:false))
    (crl-distribution-points
      (has-extension #t)
      (is-critical #t)
      (value "http://ca@NUM@.te.grid.ie/crl.pem"))
    (format
      (is-x509 #t)
      (version 3)
      (rfc3280-compliant #t))
    (key-usage
      (has-extension #t)
      (is-critical #t))
    (policy-oid
      (has-policy-oid #t)
      (policy-oid-only #t))
    (is-sharing-allowed #f)
    (max-lifetime 420)  ; days
    (natural-person
      (min-passphrase-length 12))  ;characters
    (network-entity
        (has-fqdn #t)))
    (subject-dn
      (unique-linking #t)
      (has-cn #t)
      (cn-is-appropriate-presentation-of-name #t))
    (crypto-data-generation applicant))
  (key
    (protection
      (is-informed #t))
    (size 2048))
    (revocation
      (reasons (key-lost key-compromised data-invalid)))

(CA-web-server
 (URL "http://ca@NUM@.te.grid.ie/")
 (publishes-CA-cert #t)
 (publishes-user-certs #f)
 (publishes-CRL #t)
 (publishes-CP #t)
 (cert-publication-max-latency 0)      ; < days >
 (CRL-publication-min-freq 23)         ; < freq in days >
 (CRL-publication-max-latency 0)       ; < days >
 (restricted-access #f))
(CA-LDAP-server
  (URL "none")
  (publishes-CA-cert #f)
  (publishes-user-certs #f)
```

```
    (publishes-CRL #f)
    (publishes-CP #f)
    (cert-publication-max-latency 0)          ; < days >
    (CRL-publication-min-freq 0)              ; < freq in days >
    (CRL-publication-max-latency 0)           ; < days >
    (restricted-access #f))

  (ras
    (acceptable-id (government-photo-id institute-photo-id
        institute-official-document))
    (archive (requests confirmations))
    (csr-association-validation #t)
    (personal-validation (face-to-face))
    (server-validation (check-requestor-cert))
    (secure-comms-methods (ssl-web))
    (num-ras 1)
    (verification-methods (check-ID compare-ID-with-logs))

    (RA
      (name "RA00")
      (email "ra00@ca@NUM@.te.grid.ie")
      (separate-from-CA #f)
      (secure-comms-to-CA #t)))

  (cert-issuance
    (identity-check personal-contact)
    (CA-obtains-proof-of-key-possession #f)
    (subject-keys-generated-by-CA #f))
  (confidential-user-info
    (CA-retains #f))
  (crl
    (is-published #t)
    (how-published 'http)
    (lifetime 30)
    (max-reaction-time 1)
    (lifetime-after-revocation 0)
    (min-before-revocation 7)
    (revocation-checkable-online #f)
    (status-checkable-online #f)
    (version 1)
    (rfc3280-compliant #t))
  (compliance-auditing
    (frequency 0)
    (auditor ""))
  (cert-signing-host
    (controlled-physical-access #t)
    (offline #t))
  (CA-private-keys
    (generation 'software)
    (storage 'removable-media)
    (backed-up #t)
    (activation 'passphrase))
  (certs
    (CA-key-size 2048)
    (CA-key-lifetime 3650)
    (minimum-subject-key-size 1024)
    (maximum-subject-key-lifetime 420))
  (cert-profile
    (version 3)
    (cert-extensions
       (SubjectKeyIdentifier #f)
       (AuthorityKeyIdentifier #f)
       (BasicConstraints 'critical)
       (BasicConstraints-value "CA")
       (KeyUsage 'critical)
       (Digital-Signature #t)
       (Non-Repudiation #t)
       (Key-Encipherment #t)
       (Data-Encipherment #f)
       (Certificate-Sign #f)                     ; < true | false >
       (CRL-Sign #f)                             ; < true | false >
       (CertificatePolicies #f)
       (CRLDistributionPoint #f)
       (SubjectAltName #f)
       (IssuerAltName #f)                ; < present | absent >
       (AuthorityInformationAccess #f)  ; < present | absent >
       (CA-Issuer-URL #f)                 ; < present | absent >
       (DirName #f)                      ; < present | absent >
       ;(Serial )                           ; < cert serial number >
       (NetscapeCertType #f)             ; < present | absent >
        (SSL-CA #f)                         ; < true | false >
        (SMime-CA #f)                       ; < true | false >
        (Object-Signing-CA #f)              ; < true | false >
       (NetscapeCARevocationURL #f)     ; < present | absent >
       (NetscapeCAPolicyURL #f)         ; < present | absent >
       (NetscapeComment #f))              ; < present | absent >
     )
    )
```

Appendix E

SXML Schema

E.1 Introduction

A system for SXML Schema validation was introduced in Chapter 4. This appendix describes the syntax of the schema language for SXML [100]. The syntax is quite close to an S-expression rendering of XML Schema [171, 20]. The SXML schema syntax supports a more compact notation for specifying elements, and does not use *attributes*. Section E.2 informally describes the syntax and Section E.3 presents an example use.

E.2 Syntax

The SXML Schema has syntax for defining *elements* and their *types*. These are used in a recursive fashion to build up a description of the SXML format.

E.2.1 Elements

An SXML element is an S-expression that begins with an element name and contains some content. For example, `(h1 "Introduction")` is a `h1` element which contains the string `"Introduction"`.

The syntax for defining an element in SXML Schema is as follows:

```
(element
  (name el-name)
  (type el-type)
  [optional attributes ...])
```

The `el-name` is a string or symbol value and the `el-type` is an in-line type definition or a reference to a previously defined type. The optional attributes are described below. A more compact positional syntax is also supported:

```
(element el-name el-type [optional attributes ...])
```

E.2.2 Types

Elements can be defined to have a basic type (such as integer, string, Boolean, etc.) or a defined type. Basic types are specified by name, for example:

```
(element h1 string)
```

Both *simple* and *complex* types can be defined. Simple types are derived from basic types. The first form of simple type is the *restriction* which defines a type in terms of a base type and a number of *restriction facets.* These facets can include a numeric range, a regular expression pattern or an enumeration of symbolic values.

```
(simple-type
  (name type-name)
  (restriction
    (base-type type)
    [restriction facets]
```

where the restriction facets could be

```
(pattern regexp)
(enumeration [enumeration values...])
(min-exclusive num)
(min-inclusive num)
(max-exclusive num)
(max-inclusive num)
```

Note, the facets must be compatible with the base type. The base type itself can be a basic or simple type.

Another simple type is a *union* type:

```
(simple-type
  (name type-name)
  (union
    (member-types [member types...])))
```

The member types can be other basic or simple types, defined by reference or in-line.

The final simple type is a *list.*

```
(simple-type
  (name type-name)
  (list-of
    (base-type type)))
```

Again, the base type can be other basic or simple types, defined by reference or in-line.

In addition to these simple types there is support for complex types, which allow the definition of nested elements. There is support for *sequences* and *choices.* A *sequence* is an ordered collection of elements. A minimum and maximum number of occurrences can be specified for each element in a sequence.

```
(complex−type
  (name type−name)
  (sequence
    [elements ...]))
```

The elements may be specified in-line or by reference to an element name. Inline element definitions can use the optional attributes mentioned above to further control the sequence. These sequence attributes are:

```
(max−occurs [unbounded|num])
(min−occurs num)
```

A *choice* allows selection between different elements as union allows selection between types.

```
(complex−type
  (name type−name)
  (choice
    [elements ...]))
```

Again, the elements may be specified in line or by reference.

When specifying a type in-line in an element definition, the **name** is optional.

E.3 Example

This section presents a full SXML Schema for a CA description format as an example use of the syntax. It should be noted that this Schema corresponds to a slightly different format to that used for the CA description in Appendix D as it was developed for use with a different ruleset. The differences are in the set of feature names used.

```
; CA Description
(simple−type
  (name te:entity−type)
  (restriction
    (base−type symbol)
    (enumeration CA VO)))

(simple−type
  (name te:security−level)
  (restriction
    (base−type symbol)
    (enumeration low medium high)))

(simple−type
  (name te:email−address)
  (restriction
    (base−type string)))
```

```
(simple-type
  (name te:oid)
  (restriction
    (base-type string)))

(simple-type
  (name te:uri)
  (restriction
    (base-type string)
    (pattern "(([a-zA-Z][0-9a-zA-Z+\\-\\.]*:)?/{0,2}[0-9a-zA-Z;/?:@&=+$\\.\\-_
        !~*'()%]+)?(#[0-9a-zA-Z;/?:@&=+$\\.\\-_!~*'()%]+)?")))

(simple-type
  (name te:identity-check-type)
  (restriction
    (base-type symbol)
    (enumeration personal-contact remote-contact identity-database)))

(simple-type
  (name te:x509-version)
  (restriction
    (base-type number)
    (enumeration 1 2 3)))

(complex-type
  (name te:cp-and-cps)
  (sequence
          (element RFC-2527-compliant boolean (min-occurs 0)(default #f))
          (element RFC-3647-compliant boolean (min-occurs 0)(default #f))
          (element OID-identifier te:oid (min-occurs 0))
          (element OID-in-cert boolean (min-occurs 0)(default #f))))

(complex-type
  (name te:ca-server)
  (sequence
    (element URL te:uri)
    (element publishes-CA-cert boolean)
    (element publishes-user-certs boolean)
    (element publishes-CRL boolean)
    (element publishes-CP boolean)
    (element cert-publication-max-latency integer)     ; days
    (element CRL-publication-min-freq integer)         ; freq in days
    (element CRL-publication-max-latency integer)      ; days
    (element restricted-access boolean)))

(element
(name ca-description)
(type
  (complex-type
  (sequence
    (element name string (min-occurs 1))
    (element dn string (min-occurs 1))
    (element alias string (min-occurs 0))
    (element country string (min-occurs 0))
    (element country-ID string (min-occurs 0))
    (element entity-type te:entity-type (min-occurs 0))          ;  CA | VO
    (element security-level te:security-level (min-occurs 0))    ;  high | medium
        | low
    (element CA-email te:email-address (min-occurs 0))           ;  email address
    (element CP-and-CPS te:cp-and-cps)
    (element CA-web-server te:ca-server (min-occurs 1))
    (element CA-LDAP-server te:ca-server (min-occurs 0))
    (element RAs
             (complex-type
               (sequence
                 (element RA
                          (complex-type
                            (sequence
                              (element name string)
                              (element email te:email-address)
                              (element separate-from-CA boolean)
                              (element secure-comms-to-CA boolean)))
                          (max-occurs unbounded))))
             (min-occurs 0))


    (element cert-issuance
             (complex-type
               (sequence
                 (element identity-check te:identity-check-type)
                 (element CA-obtains-proof-of-key-possession boolean)
                 (element subject-keys-generated-by-CA boolean))))
    (element confidential-user-info
             (complex-type
               (sequence
                 (element CA-retains boolean))))
    (element CRLs
             (complex-type
               (sequence
                 (element lifetime integer)
                 (element lifetime-after-revocation integer)
                 (element revocation-checkable-online boolean)
             (element status-checkable-online boolean))))
(element compliance-auditing
         (complex-type
           (sequence
             (element frequency integer)
             (element auditor string))))
(element cert-signing-host
         (complex-type
           (sequence
             (element controlled-physical-access boolean)
             (element offline boolean))))
(element CA-private-keys
         (complex-type
           (sequence
             (element generation
                      (simple-type
                        (restriction
                          (enumeration 'software 'hardware))))
             (element storage
                      (simple-type
                        (restriction
                          (enumeration 'removable-media 'paper 'hardware))))
             (element backed-up boolean)
             (element activation
                      (simple-type
                        (restriction
                          (enumeration 'passphrase 'token)))))))
(element certs
         (complex-type
           (sequence
             (element CA-key-size integer)
             (element CA-key-lifetime integer)
             (element minimum-subject-key-size integer)
             (element maximum-subject-key-lifetime integer))))
(element cert-profile
         (complex-type
           (sequence
             (element version te:x509-version)
             (element cert-extensions
                      (complex-type
                        (sequence
                          (element SubjectKeyIdentifier boolean)
                          (element AuthorityKeyIdentifier boolean)
                          (element BasicConstraints
                                   (simple-type
                                     (restriction
                                    (enumeration
                                      'critical 'not-critical))))
                           (element BasicConstraints-value string)
                           (element KeyUsage
                                    (simple-type
                                      (restriction
                                        (enumeration
                                          'critical 'not-critical))))
                           (element Digital-Signature boolean)
                           (element Non-Repudiation boolean)
                           (element Key-Encipherment boolean)
                           (element Data-Encipherment boolean)
                           (element Certificate-Sign boolean)
                           (element CRL-Sign boolean)
                           (element CertificatePolicies boolean)
                           (element CRLDistributionPoint boolean)
                           (element SubjectAltName boolean)
                           (element IssuerAltName boolean)
                           (element AuthorityInformationAccess boolean)
                           (element CA-Issuer-URL boolean)
                           (element DirName boolean)
                           (element NetscapeCertType boolean)
                           (element SSL-CA boolean)
                           (element SMime-CA boolean)
                           (element Object-Signing-CA boolean)
                           (element NetscapeCARevocationURL boolean)
                           (element NetscapeCAPolicyURL boolean)
                           (element NetscapeComment boolean)))))))
(element CRL-profile
         (complex-type
           (sequence
             (element version te:x509-version))))))))
```