PASSWORD
R2.5 : Certification Authority Requirements

Michael Roe
Cambridge University Computer Laboratory
Computer Security Group

Version 1.0
November 1992

Contents

List of Figures

Acknowledgments

This report incorporates technical input from many members of the PASSWORD project, particularly Wolfgang Schneider, Christian Huitema, Suzan Mendes and Peter Kirstein.

I would like to thank Roger Needham, David Wheeler, Bill Harbison and Mark Lomas for their participation in discussions leading to the writing of this report.

The work on statistical testing of random number generators referred to in chapter 5 was carried out in collaboration with Ross Anderson, Frank Kelly, Richard Gibbens and Chris Jagger.

The design for a prototype hardware random number generator was supplied by Andrew Findlay.

Preface

This report is deliverable R2.5 of the VALUE "PASSWORD" project. Deliverable R1.1 (Service Requirements) dealt with security requirements specific to the particular applications which are to be piloted under the PASSWORD project. This report examines in detail the requirements for one aspect of the common infrastructure underlying all these applications, namely the use of off-line certification authorities for key distribution. Deliverable R2.6 will address the common requirements for secure local storage and management of private keys and other security-critical information.

Chapter 1 describes the requirements for hardware and software used by a certification authority, using the "security target" format required for evaluation under the EC Information Technology Evaluation Criteria (ITSEC) [7]. Although it is not intended to obtain ITSEC evaluation of any product or system under the PASSWORD project, we have found the security target format to be a convenient means of describing requirements.

ITSEC-style security targets, such as the one in the chapter 1, are only concerned with security requirements and do not address the acceptability of the system to end users or administrators. These user interface issues are covered in chapter 2.

The initial exchange of keys between a user and a CA, or between two CAs, must be performed using some physically secure means. As this exchange does not (and cannot) take place over an open network using OSI protocols, the procedures used are not specified in the relevant standards. However, in order to produce an actual implementation, a decision must be taken as to how this is done. Some possible procedures are described in chapter 3.

If a user's name or public key changes, the CA must revoke their old certificate and issue a new one. The procedures to be used are described in chapter 4.

In order to generate cryptographic keys, either the CA or the User Agent must have access to a source of random numbers. The generation of such random numbers is usually performed by a hardware device specifically designed for this purpose. The requirements and evaluation techniques for these devices are discussed in chapter 5.

In the course of preparing to pilot X.509 authentication, it has become clear that the current standard is deficient in several major areas. These deficiencies have been recognised within ISO, and there should eventually be new or modified standards to correct them. The currently known defects are listed in appendix A.

Privacy Enhanced Mail defines a revocation list format which corrects some of the deficiencies of the ISO 9594-8 format. In order to distribute PEM revocation lists using the OSI Directory Service, it is necessary to define some new attributes and object classes. These are described in appendix B.

Any system or product which is to be evaluated at ITSEC level E6 must be modelled using a formal notation in the security target. In order to illustrate what such a security target might look like, appendices C and D contain formal models of certificate generation and verification processes.

Chapter 1

Security Target

1.1 System Description

A *Certification Authority* (CA) is an organisation (or subdivision of an organisation) responsible for verifying the security attributes of computer system users, and entering this verified information into the computer system.

In order to perform this function, the CA operator interacts with computer system via a *Certificate Generation Unit* (CGU). This security target defines the security requirements for a Certificate Generation Unit; the security requirements for communication channels between the CGU and other systems; and the procedures that should be followed by the CA operator to ensure secure operation.

The Certificate Generation Unit can be used to generate a cryptographically protected data structure (a *Certificate*) which associates a user identifier with the public component of a (public key, private key) pair. A user in possession of a valid certificate and the corresponding private key may authenticate themselves as the enclosed user id.

The GCU can also be used to generate a *CA Certificate*, which associates the name of another Certification Authority with a public key. CA Certificates are used to delegate authority to other CAs.

1.2 System Security Policy

1.2.1 Objectives

The primary objective of the system security policy is to ensure that a certificate will only be issued if the corresponding private key is known only to the identified user.

Other objectives are to limit the adverse consequences of malicious action by a CA operator; and to ensure that CA operators can be held accountable for their actions.

1.2.2 Threats

Loss of Confidentiality

Anyone in possession of a user's private key can authenticate themselves as that user. Accordingly, it is vital to preserve the confidentiality of private keys. Loss of confidentiality could occur in the following ways:

- Compromise of the user's local key storage. This threat occurs in a system other than the Certificate Generation Unit (the User Agent), and hence is outside the scope of this security target.
- Interception of private keys transmitted between the User Agent and the Certificate Generation Unit.
- Compromise of the key generation process, (for example, if an attacker can mathematically predict which key will be generated). To protect against this threat, the key generation process must contain at least one genuinely random, not just pseudo-random, process.
- Disabling of the confidentiality mechanisms by a malicious CA operator.

Modification of Data

The security of applications using certificates is dependent on the information contained in certificates being true. It is therefore essential to prevent unauthorised modification of certificate contents. Modification of this data could potentially occur in the following ways:

- Modification of certificate contents during transmission.
- Modification of stored certificates.
- Modification of stored security attributes prior to their being packaged in a certificate.
- Modification of security attributes during transmission to a CA, prior to their being packaged in a certificate.
- Modification of security attributes by a malicious CA operator.

Masquerade

Masquerades occur when an entity pretends to be a different entity. Masquerades could potentially occur in the following ways:

- Masquerade of a user requesting a certificate.
- Masquerade of a CA issuing a certificate.
- Masquerade of a CA during cross-certification.

False Repudiation

False repudiation occurs when an entity denies sending or receiving information when it has in fact done so. False repudiation could potentially occur in the following ways:

- Repudiation by a user of having requested a certificate.
- Repudiation by a user of having received a secret key.
- Repudiation by a user of having requested the revocation of a certificate.
- Repudiation by a CA of having issued a certificate.
- Repudiation by a CA of having revoked a certificate.
- Repudiation by a CA of having requested a cross-certificate.

Exceeding Authority

Exceeding authority occurs when an entity performs functions for which it has not been authorised. This threat could potentially occur in the following ways:

- A CA issuing a certificate for a member of an organisation over which it has no jurisdiction.
- A CA revoking a certificate issued by another authority over which it has no jurisdiction.

Misuse of Privilege

Misuse of privilege occurs when a trusted entity performs an action that it was trusted never to do. Misuse of privilege can occur in the following ways:

- A CA using a user's private key to forge signatures.
- A CA using a user's private key to decrypt confidential information.
- A CA issuing incorrect certificates in order to subvert a security mechanism.
- A CA issuing incorrect revocation lists in order to cause denial of service.

1.2.3 Technical Security Policy

As will be explained in chapter A, the certificate format defined in ISO 9594-8 [8] is not ideal. In order to provide adequate security while remaining conformant with the standard, the security policy must impose considerable restrictions on the way in which the standard is used. In particular, it is necessary to impose restrictions on the conventions used in naming end users and authorities. We recognise that these restrictions make the system harder to use and to manage. This security policy attempts to find a workable compromise between the need for security and the need for manageability, given the current International Standards and the current state of the art. It is closely aligned with solution adopted by Privacy Enhanced Mail [9].

Public and Private Trust Models

Each user of the system trusts different CAs to different extents. These different levels arise because the relationship between the user and the CA may differ. For example, users will normally place a great deal of trust in their local CA, because they know who runs it and they trust them not to act maliciously. Geographically or politically distant CAs will not be trusted quite as much, because their relationship with the user is less close. Thus, each user imposes their own private trust hierarchy on the set of CAs.

It is very difficult for a user of the system to accurately describe their trust model starting from scratch. To make trust models easier to formulate, a public trust hierarchy is defined. Each user describes their own private trust model in terms of the public one. For this to be efficient, the public trust model must be reasonably close to beliefs of the majority of users.

Rights and Obligations of CAs

In order to join the public trust hierarchy, a CA must accept certain obligations. These vary according to the position of the CA in the public hierarchy. Individual users may position CAs differently in their private hierarchies. That is, an individual may believe that a CA is not fulfilling its declared obligations, or alternatively may believe that a CA is more trustworthy than it is obliged to be.

These obligations fall into three categories:

- An obligation to not knowingly make false statements.

  That is, the CA shall only make statements which it believes.

- An obligation to take reasonable care.

  That is, the CA shall only make statements based on justified beliefs.

- An obligation to not make ambiguous statements.

  That is, the language used to express statements (the certificate format) shall assign one and only one meaning to the statement made. The meaning attached to a particular certificate is defined partly by the ISO 9594-8 standard and partly by the security policy adopted (ie. this document).

In return for accepting these obligations, a CA is granted *jurisdiction* over statements of a particular form. That is, the public trust model declares that when a CA makes statements of this form, it should be believed. Individuals may accord a CA different jurisdiction in their private trust models, and either disbelieve statements over which it has public jurisdiction, or believe statements for which is has no such jurisdiction.

CAs are permitted to make statements for which they have no jurisdiction, provided that in doing so they still meet their obligations to be truthful, to be unambiguous and to take reasonable care. The definition of what constitutes reasonable care is to be derived from the position of the CA in the public hierarchy, not from the statement made.

When certificates are used for the purpose of *non-repudiation* (demonstrating to a third party that a message was sent, or an action was performed), it is the third party's trust model that is used, not that of either of the parties in dispute. To be more exact, the trust model used is defined by the rules for arbitrating disputes, since the third party is acting as an impartial arbiter, rather than as a private individual.

If the public trust model declares that a certificate should not be believed, then it will not be accepted by the arbiter. Hence, users who choose to believe such certificates do so at their own risk.

This policy does not place any obligation on the users of certificates to either believe or disbelieve them. However, the designers and evaluators of certificate-handling software are obliged to ensure that certificates are checked against the public trust model. Where the public model declares that a certificate should not be believed, it should not be accepted unless the user has performed an explicit action to override the public model.

Geometry of Trust Models

Mathematically, any partial ordering of the set of authorities can be used as a trust model. The public trust model should also meet the following constraints:

1. It should be reasonably close to the actual beliefs of the majority of users.
2. It should be simple to compute.
3. All information necessary to compute it must be obtainable in a secure manner.
4. The amount of information that each user needs to keep locally in secure storage should remain small even when the total number of users becomes very large.

There are several simple structures that meet the last three of these conditions:

- All CAs are equal. That is, every CA is trusted to sign any certificate. This trust model is implicitly assumed by ISO 9594-1. Although it is internally consistent, this model does not accurately represent user's beliefs, as some CAs will be regarded as much more trustworthy than others.

- CAs are arranged into layers. Each CA is trusted to sign certificates for CAs in lower layers, but is not trusted to sign certificates for CAs in the same or higher layers.
- The Directory Information Tree is used to derive the trust model. A is trusted to sign a certificate for B if and only if A's name is a prefix of B's name. This approach has the distinct disadvantage of confusing two abstractions which ought to be kept separate: naming and trust. However, it is close to many user's actual beliefs, at least for regions low down in the DIT and far away from the user's own name. That is, this trust model is close to user's actual beliefs when they have almost no information about the entities concerned. This approach breaks down near the top of the DIT, where the entities (such as countries) are large enough to be well-known, and close to the user, where there is likely to be considerable local information.

The public trust model we have adopted is a combination of the last two of these structures.

CA Hierarchy

Certification Authorities will be divided into the following categories:

- Top-Level Certifying Authorities
- Policy Certifying Authorities
- Organisational Certifying Authorities

It is permissible for a single organisation to act in more than one of these roles, using the same hardware. However, the authority should distinguish these roles by using a different name and public key for each.

Each class of Certification Authority is restricted in the range of certificates it has jurisdiction to sign. The enforcement of jurisdiction rules is to be performed by the recipient of the certificate, not the Certification Authority or the Certificate Generation Unit. Even if a CA generates a certificate for which it has no jurisdiction, this does not directly make possible any attacks on the system; the invalid certificate should be rejected by all recipients.

Certification Authorities need to be aware of their jurisdiction in order to construct certificates which are likely to be believed. A CA which is believed by no-one is performing no useful function.

Organisational CAs

An *Organisational Certifying Authority* certifies that users belong to a particular organisation. An Organisational CA only has jurisdiction to sign certificates of members of that organisation. Where the same operator and hardware are used to provide Organisational CA service for more than one organisation, these roles shall be distinguished by the use of distinct names and public keys. Very large organisations may require different Organisational CAs for different parts of the organisation. In this case, the CAs for the individual organisational units may be certified directly by a Policy CA. Alternatively, the Policy CA may certify a single Organisational CA which then delegates authority for individual organisational units to subordinate CAs by signing an appropriate CA Certificate.

An Organisational CA must be named in the Directory Service with the same name as the organisation or organisational unit that it serves. An Organisational CA only has jurisdiction over certificates of users whose names are beneath its own in the Directory Information Tree. In order to distinguish between certificates of users and certificates of Organisational CAs, the last component of an Organisational CA's name must be an *organisationName* or *organisationalUnitName* attribute, while the last component of a user name must not be either of these attributes.

Policy CAs

A *Policy CA* certifies that Organisational CAs act in accordance with a specified security policy and are authorised to act on behalf of particular organisations. This model allows multiple policy CAs in order to permit the use of CAs with different levels of assurance in different contexts. For example, many interpersonal messages need only be protected to a low level of assurance, and users may opt for reduced assurance in return for a cheaper and more widespread service. In contrast, the banking, government, and military communities will require higher levels of assurance, and will probably not be prepared to trust commercial CAs.

It is possible for there to be several Policy CAs with the same stated security policy. This may be useful when several countries have agreed on a policy and have agreed to recognise each other's certificates, but cannot agree on which country will run the Policy CA.

Policy CAs may be named anywhere in the Directory Information Tree, although it is strongly recommended that the last component of a Policy CA name be an *organisationName* or *organisationalUnitName* attribute. Policy CAs have jurisdiction to sign CA certificates for any Organisational CA, regardless of their relative position in the DIT. In particular, Policy CAs may sign certificates for Organisational CAs in other countries. However, we expect that Organisational CAs will usually be certified by a Policy CA within their own country.

A Policy CA should only certify users or Organisational CAs, and should not certify other Policy CAs. The reason for this restriction is that the ISO 9594-8 certificate format does not contain an indication of the type of the certificate subject (ie. whether the subject is a Policy CA, Organisational CA or user). Users and Organisational CAs can be distinguished by the form of their names, but there is no third distinct name form that can be used to identify policy CAs. If Policy CAs were allowed to certify other Policy CAs, the resulting certificates would be ambiguous. This restriction is also enforced by the recipient of a certificate, not the CGU; any certificate issued by a PCA that is not a user certificate will be assumed to be a Organisational CA certificate.

Each Policy CA should make available a statement of its security policy, including the level of assurance of the software components used and the stringency of operating procedures. Policy CAs should only certify Organisational CAs when they have reason to believe that the software and operating procedures of the Organisational CA conform to their published security policy.

Users who are not affiliated to any organisation (or who are affiliated to an organisation which does not have its own CA) may be certified directly by a Policy CA. In this case, the Policy CA will itself perform all the identification checks that its policy requires an Organisational CA to carry out.

Top Level CAs

A *Top-Level CA* certifies Policy CAs. TLCAs are used to simplify key distribution when the number of Policy CAs becomes large. As with Policy CAs, there will be multiple TLCAs providing different levels of assurance, and serving different communities. The number of TLCAs is likely to be similar to the number of Policy CAs, and often the same organisation will run both a Policy CA and a TLCA. However, one TLCA can certify many Policy CAs. A user requiring a particular level of assurance need only obtain (by out of band means) the public key of a single TLCA operated to that assurance level. The CA certificates issued by that TLCA can then be used to determine the public keys of all Policy CAs worldwide that provide an equivalent level of assurance. We estimate that there will eventually be approximately 20 different assurance levels in use, and we expect that each country will provide its own TLCA for most of these assurance levels. Certificates issued in one country will be normally be acceptable in another, as the issuing Policy CA will have been certified by the receiver's TLCA. By running their own TLCAs, national governments retain the ability to cancel these bilateral agreements by revoking certificates they have issued for foreign Policy CAs. In contrast, agreeing to let one country or organisation run the only TLCA at a particular assurance level transfers authority in a way which would be extremely difficult to revoke.

Structure for the PASSWORD project

For the purposes of the PASSWORD project, each national consortium (France, Germany and the UK) will run one TLCA and one or more Policy CAs. Each TLCA will certify all the Policy CAs. Although it will be usual for users to trust a TLCA in the same country and for organisations to be certified by a Policy CA in the same country, everyone is free to use the services of foreign CAs instead. We expect that organisations from other EC countries will become involved in the PASSWORD pilot. Such organisations can be certified by Policy CAs run by any of the PASSWORD contractors, and their users can choose to trust any (or none) of the TLCAs.

Example Hierarchy

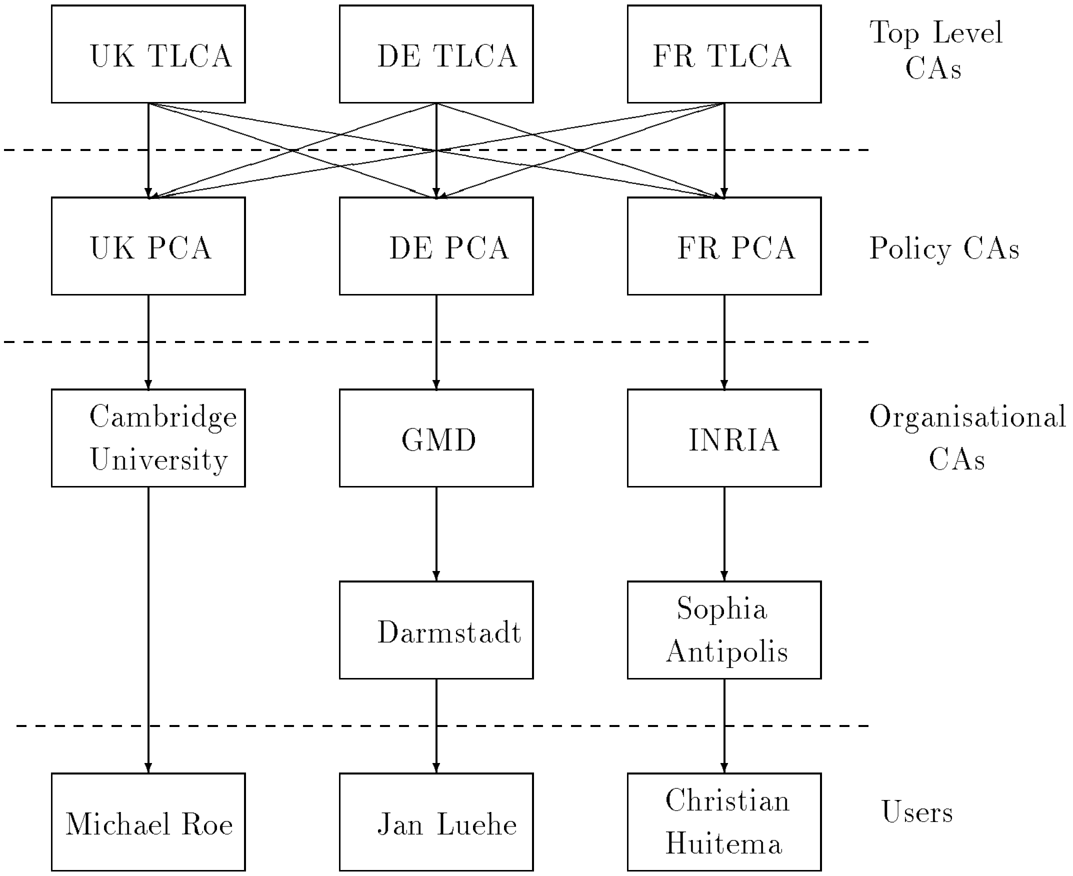

Figure 1.1: Example CA Hierarchy

Figure 1.1 shows a hypothetical example of this CA hierarchy. All the TLCAs shown have certified all the policy CAs, ensuring complete connectivity within a particular community. It is not a requirement of the model that there be complete connectivity between all entities in existence. There may be other TLCAs and PCAs, not shown in the diagram, that neither need nor desire total security connectivity with this particular group of users; the classified military community is a possible example of this.

Organisations which have multiple sites (such as GMD) are shown as delegating authority to an Organisational CA at each site. For the purposes of illustrating that this is possible, Cambridge University is shown as using a single CA for the entire organisation. In practise, most organisations may find that at least one delegation is needed to make the system manageable.

Persona Certificates

The purpose of a certificate is to guarantee the integrity and authenticity of some items of security information. The various mechanisms used for distributing certificates ensure that this integrity protected information becomes known to everyone that requires it. This widespread distribution makes it very easy for an attacker of the system to obtain someone else's certificate. Possession of a captured certificate should not give an attacker any advantage, as the security information contained within is supposed to be public knowledge. However, care must be taken that certificates never contain information which is confidential.

There are security attributes for which both integrity and confidentiality is required. For example, security clearances must be integrity protected to ensure that the user really does have the access rights she claims; however, user's clearances are often also treated as confidential. (Knowledge of everyone's clearance gives an attacker a good indication of who to bribe, or whose password they should try to guess). Information of this type should not be distributed using X.509 certificates.

It is sometimes the case that the user's real identity is confidential. In this case, a certificate must be used which provides assurance of other security attributes, but which does not include her real name. These are known as *persona certificates*.

Persona certificates may appear paradoxical, as it is frequently the case that the primary use of a certificate is to convey the user's identity. However, there are attributes other than identity which may usefully be included in a certificate. These include:

- Organisational affiliation
- Occupancy of a particular role
- An access control identifier

Identifiers used for the purpose of access control can be random bit strings, and need bear no relation to the actual name of the associated user.

A CA may only issue persona certificates if this is permitted by the policy of its PCA. For the purposes of determining jurisdiction, a persona certificate is treated identically to a normal user certificate.

Certificates for Non-Human Users

In many OSI protocols, processes (eg. Directory System Agents, Mail Transfer Agents) have their own public keys and certificates, which they use to identify themselves. In this case, the certificate identifies a service rather than a person.

Many of the checks a CA would perform when registering a human user (eg. asking to see their passport and driver's license) are inapplicable when registering a process. Instead, the CA must be provided with evidence that an organisation or person has agreed to provide the service and has agreed to accept responsibility if the process fails to perform correctly. The evidence needed and the extent of the service provider's liability is determined by the policy statement issued by the Policy CA above the registering CA. We recommend that the CAs and Policy CAs used to register processes are different from those used to register human users, to indicate to users of these certificates that different security policies apply.

Certificates for DSAs

There are particular problems associated with issuing certificates for Directory System Agents, because they have a special relationship with the name space. Some Directory implementations (eg. QUIPU) require that the names of DSAs have a particular form, in order to ensure that Directory operations are guaranteed to terminate. If DSAs were permitted to have arbitrary names, the situation could arise that the information needed to contact DSA X could only be obtained by contacting DSA X (and hence it can't be obtained). Our security policy also places restrictions on the names a certificate subject can have, and it is possible that these two sets of naming restrictions could come into conflict.

The ideal way to resolve this conflict would be to remove both sets of naming restrictions, by extending both the certificate format and the Directory name resolution protocols. Unfortunately, it is not possible to make changes as drastic as this within the timescale of the PASSWORD project.

As a short term solution, we suggest that the DSAs for which this is a problem should be certified directly by a Policy CA. Policy CAs have jurisdiction to sign certificates for any name, and so can accommodate restrictions imposed by particular Directory implementations.

It can be argued that this is the correct thing to do anyway; a DSA that can sign the result of a query is in effect acting as an on-line certification authority, and so ought to be delegated authority to do so by a Policy CA or a superior Organisational CA.

Proxy CAs

Many small organisations will not be able to afford either the equipment or the time to run their own Organisational Certification Authority. In this case, another organisation can run a CA on their behalf. A CA should always use a name appropriate to the role in which it is acting. Thus, if organisation X provides proxy CA service for organisation Y, it should use Y's name in the certificate issuer field when issuing certificates on behalf of Y.

The hardware used to generate certificates will usually be capable of generating certificates with more than one issuer name. (This is certainly true is a general-purpose computer is used. The BBN "Safekeyper" device also has this facility). Hence, a organisation providing proxy CA service will not need to buy different hardware to run its own CA and the proxy CA.

Uniqueness of Distinguished Names

Certification Authorities must ensure that two different users cannot both be issued with certificates for the same Distinguished Name. If this were to happen, it would clearly be a threat to both integrity and confidentiality, as the two users would be able to masquerade as each other and decrypt confidential information intended for the other person.

A single CA can avoid this by keeping track of which Distinguished Names it has issued certificates for, and to whom. If two people called "John Smith" both request certificates from the same CA, the CA can detect this by comparing the evidence of identity they provide (eg. they will have different passport numbers, driver's license numbers, signatures and photographs). The CA can then require them to use Distinguished Names which differ in some respect, such as Organisational Unit Name. It would be hard to do this automatically. However, existing Directory Service pilots have shown that it is possible to ensure unique names with a small amount of manual intervention to choose the distinguishing attribute.

It is harder to ensure that two different CAs do not both issue certificates with the same subject name to different users. Organisational CAs have jurisdiction over disjoint parts of the name space, and so need not consult with each other when issuing certificates within their jurisdiction. The possibility remains that a Policy CA and an Organisational CA might both issue certificates for the same name. Policy CAs should obtain the consent of the organisation involved before issuing a certificate that declares a user to be a member of a particular organisation. As part of this process, the Policy CA should also obtain confirmation that there is no Organisational CA which could issue the certificate instead.

Relationship with Privacy Enhanced Mail

This CA hierarchy has been designed to be compatible with the one defined for Privacy Enhanced Mail. That is, it should be possible for a CA to be a Policy CA in both the PASSWORD and PEM hierarchies. Such a PCA must comply with the obligations of both schemes (including paying any necessary fees to the Internet Society). If a Policy CA does not wish to be part of the PEM hierarchy, then it is not required to comply with any additional obligations defined by PEM.

The trust model defined by PEM can be treated as if it were a private trust model that has been derived from the PASSWORD public trust model. The main difference of the PEM model is that only one Top Level Certifying Authority is trusted, the *Internet Certifying Authority*.

Relationship with PGP

The main competitor to Privacy Enhanced Mail is *Pretty Good Privacy* (PGP), a public-domain program by Philip Zimmermann. Unlike PEM, PGP assumes that the name space is unstructured, and it is left up to the user to decide who has jurisdiction to sign which certificates. In terms of our architecture, this corresponds to a completely flat public trust model, where by default no-one has jurisdiction to sign a certificate for anyone else. The design of PGP expects physically secure exchanges to be the primary means of transferring public keys. PGP also has a special message format for transferring public keys, which provides a similar function to X.509 certificates. These messages are only acted upon if the user performs an explicit action to accept them. By choosing to accept some of these control messages, PGP users can gradually build up a structured private trust model.

1.2.4 Security Policy Model

The security requirements for a Certificate Generation Unit may be described in terms of the Clark-Wilson model [6]. Although it was originally intended as a model of integrity, the Clark-Wilson model can be extended to describe confidentiality requirements by the addition of an additional condition ([C6] below). Data items are separated into those which have integrity or confidentiality requirements (*Constrained Data Items*, CDIs) and those which do not (*Unconstrained Data Items*, UDIs). CDIs may only be accessed by invoking trusted programs called *Transformation Procedures* (TPs). The CDIs are set to an initial secure state by invoking an *Integrity Verification Procedure* (IVP).

The Clark-Wilson model requires the following security enforcing functions to be provided:

- [E1] The system must maintain a list of relations of the form (TP_i, (CDI_a, CDI_b, ...)), where the list of CDIs defines a particular set of arguments for which the TP has been certified. It must ensure that the only manipulation of any CDI is by a TP, where the TP is operating on the CDI as specified in one of these relations.
- [E2] The system must maintain a list of relations of the form (*UserId*, TP_i, (CDI_a, CDI_b,...) which relate a user, a TP and the data objects that the TP may reference on behalf of that user. It must ensure that only executions described in one of the relations are performed.
- [E3] The system must authenticate the identity of each user before attempting to execute a TP.
- [E4] The relations described in [E1] and [E2] may only be modified by the agent who is authorised to certify the system.

The model also requires that all trusted programs are certified to be correct. The evaluator of the system must ensure that the following conditions hold:

- [C1] All IVPs must be certified to be valid. That is, all CDIs will be in a valid state after the IVP is run.
- [C2] All TPs must be certified to be valid. That is, they must take a CDI to a valid final state, given that it was in a valid state to begin with. For each TP, and each set of of CDIs that it may manipulate, the security officer must specify a relation which defines that execution, using the mechanism of clause [E1].
- [C3] The list of relations in clause [E2] must be certified to meet the separation of duty requirement.
- [C4] All TPs must be certified to write to an append-only CDI (the log) all information necessary to permit the nature of the operation to be reconstructed.
- [C5] Any TP that takes a UDI as an input value and a CDI as an output value must be certified to perform only valid transformations, or else no transformation, for all possible values of the UDI.

- [C6] Any TP that takes a CDI as an input value and a UDI as an output value must be certified to meet the confidentiality requirements.

1.3 Security Enforcing Functions

The following descriptions of the security enforcing functions to be implemented are derived from the ITSEC example functionality classes F-B3, F-DI and F-DC. As a Certificate Generation Unit is not a general purpose computing device, some of the F-B3 clauses are not applicable. This section lists only those functions that we consider to be relevant to this particular application.

The ITSEC F-B3 access control section requires the implementation to conform to the Bell-LaPadula model of confidentiality [2, 3]. We have changed this part of the ITSEC text to require support of the Clark-Wilson model [6], as this is used to express our integrity requirements.

Unlike ITSEC F-B3 and Orange Book B3 [11], this security target does not require access control enforcement functions to be implemented as a reference monitor. Any implementation methodology that provides the required functions to the required level of assurance is acceptable. For example, it would be acceptable to use static analysis tools to demonstrate that forbidden accesses never occur (provided that there were mechanisms to prevent corruption of the validated code by malicious operators).

When applying the ITSEC criteria to a Certificate Generation Unit, the phrase *Target of Evaluation* (TOE) shall be taken to mean a Certificate Generation Unit, and the word "User" shall be taken to mean the CA operator.

1.3.1 Identification and Authentication

The TOE shall uniquely identify and authenticate users. This identification and authentication shall take place prior to all other interactions between the TOE and the user. Other interactions shall only be possible after successful identification and authentication. The authentication information shall be stored in such a way that it can only be accessed by authorised users. Identification and authentication shall be handled via a trusted path between user and TOE initialised by the user or by the TOE. For every interaction the TOE shall be able to establish the identity of the user.

1.3.2 Access Control

The TOE shall be able to distinguish and administer access rights between each user and the objects which are subject to the administration of rights, on the basis of an individual user, on the basis of membership of a group of users, or both. It shall be possible to use group access rights to support roles. As a minimum, the roles of TOE operator and administrator shall be definable. The roles of the TOE operator, TOE administrator and

TOE security officer shall be separated. It shall be possible to completely deny users or user groups access to an object. It shall be possible to restrict a user's access to an object to those operations which do not modify it. It shall be possible to grant the access rights to an object down to the granularity of an individual user. It shall not be possible for anyone who is not an authorised user to grant or revoke access rights to an object.

The TOE shall maintain a list of trusted programs. The actions for adding and deleting programs from this list shall be restricted to the TOE security officer. The TOE shall provide each such program with a unique identifier. The value of this identifier shall serve as a basis for mandatory access rights.

For each object which is subject to the administration of rights, it shall be possible to supply a list of the programs which are permitted access to this object. When such a list is provided, all attempted accesses using other programs shall be rejected. The actions for adding and deleting programs from this list shall be restricted to the TOE security officer.

With each attempt by users to access objects which are subject to the administration of rights, the TOE shall verify the validity of the request. Unauthorised access attempts shall be rejected. The identifier of the program used to perform the access shall serve as the basis for decisions concerning mandatory access control. The rules should unambiguously specify when a subject is allowed access to such a protected object. If discretionary access rights are also assigned for an object, access shall only be permitted provided that both the discretionary and the mandatory access rights allow such accesses.

1.3.3 Accountability

The TOE shall contain an accountability component which is able, for each of the following events, to log that event together with the required data:

- Issue of a Certificate

  Required data: Date; time; certificate subject name; subject public key; validity start time; validity end time; certificate serial number

- Revocation of a Certificate

  Required data: Date; time; certificate serial number

- Issue of a Revocation List

  Required data: Date; time

Unauthorised users shall not be permitted to access accountability data. It shall be possible to selectively account for the actions of one or more users. Tools to examine and to maintain the accountability files shall exist and be documented. These tools shall allow the actions of one or more users to be identified selectively.

1.3.4 Audit

Tools to examine the accountability files for the purpose of audit shall exist and be documented. These tools shall allow the actions of one or more users to be identified selectively.

1.3.5 Object Reuse

All storage objects returned to the TOE shall be treated before reuse by other subjects, in such a way that no conclusions can be drawn regarding the preceding content.

1.3.6 Data Confidentiality

The TOE shall have a facility to encrypt user information prior to exchange and (at the receiving end) to decrypt it automatically. It shall be assured that the parameter values (eg. keys) required for decrypting are protected in such a manner that no unauthorised person can access this data.

1.3.7 Data Integrity

Methods for error detection and error correction shall be applied in the case of data exchange. These mechanisms shall be designed in such a way that intentional manipulations of the address fields and user data can be identified. Knowledge only of the algorithms applied in the mechanisms without any special additional knowledge shall not enable unrecognised manipulations of the aforementioned data. The additional knowledge required for this shall be protected in such a manner that it can only be accessed by a few authorised users.

Moreover, mechanisms shall be used which uniquely identify as an error the unauthorised replay of data.

1.4 Required Security Mechanisms

Certificates shall be encoded in the format described in ISO 9594-8, "The Directory — Authentication Framework".

Where a public-key cryptosystem is required, the RSA algorithm shall be used. Where a cryptographic hash function is required, either the MD2, MD4 or MD5 algorithms shall be used.

1.5 Minimum Strength of Mechanisms

The mechanisms used shall have a strength against direct attack of "high" as defined in the ITSEC/ITSEM.

1.6 Target Evaluation Level

It is not intended to obtain ITSEC evaluation of any software under the PASSWORD pilot project. For service provision beyond the pilot stage it would be highly desirable to use Certificate Generation Units validated to at least the E4 level.

Chapter 2

User Acceptability and Manageability

ITSEC-style security targets, such as the one in the previous chapter, are only concerned with security requirements and do not address the acceptability of the system to end users or administrators. Indeed, the requirement that the system be secure is often in direct conflict with the requirement that it be easy to administer and use. The "Ease of Use Analysis" performed during ITSEC evaluation has the objective of ensuring that it is very difficult to use the system in an insecure manner. This chapter is concerned with the converse problem of ensuring that the system is easy to use in a secure manner.

2.1 User Acceptability

It cannot be assumed that all users will be familiar with the theory underlying the security mechanisms. Equally, the user should not have to deal directly with the control information (such as encryption keys) used by these mechanisms. Clearly the user must input some information (such as a password or PIN) to identify herself to the system. The length of this authenticating information should be chosen so as to be long enough to provide effective security but not be so long as to be difficult to enter. It is highly desirable that the user only have to input this information once per login session, rather that having to rekey it each time a new application is run.

One means of meeting these objectives is an *Enrollment Agent*: a program which the user runs to initialise their local security control information. This program can hide the mechanism specific details (eg. large prime numbers, certificates, mathematical models of policy) and present a simple password or PIN interface.

2.2 Manageability

We identify a number of common activities which an operator must be able to perform quickly, simply, and with a minimum of opportunity for error:

- Registration of a new user
- Revocation of a user's key
- Registration of a large number of new users

  In many institutions (particularly Universities) it is often necessary to register hundreds of new users simultaneously. It should be possible to enter a list of names and have the remaining processing performed without operator intervention. Systems which require the entry of single name, perform a few minutes of processing and then ask for the next name are not acceptable, as they vastly increase the amount of operator time required.
- Revocation of a large number of user's keys

  In the event of a serious security breach, it may be necessary to revoke the current key and reissue new keys to all users of a particular system. The software should allow this to be performed simply, in a similar fashion to bulk user registration described above.
- Bulk re-issue of certificates

  In the event of a CA's key being changed, it will be necessary to re-issue the certificates of all users. If the cause of a key change is a security breach, then it may be best to change all user keys by the procedure outlined above. However, there are many cases when it is desired to change the CA key without changing user keys. For example, some authorities may change their key periodically to reduce the risk of cryptanalysis. For this reason, there should be a simple procedure for re-issuing all previously issued (but not revoked) certificates.

Chapter 3

Procedures for Initial Key Exchange

When a user or an organisation is registered with a CA for the first time, it is necessary for the two parties to exchange some information that they will use identify each other in subsequent interactions. The authenticity of the information provided by the user must be guaranteed in order to prevent an attacker from registering in someone else's name. The authenticity of the information provided by the CA must be guaranteed so that the user cannot be tricked into accepting forged certificates. However, cryptographic techniques cannot be used to ensure authenticity at this stage, as there are no keys shared between the two parties. Some physically secure means must be used to exchange the first key. The initial key can then be used to provide cryptographic protection for exchanging subsequent keys.

Protocols of this type are outside the scope of the Open Systems Interconnection series of standards, as they involve operating procedures and other "local matters". These procedures must nevertheless be specified somewhere before a practical system can be implemented. This chapter describes possible protocols for each of the different types of initial key exchange needed by a CA.

3.1 Notation

This chapter uses the following notation for describing cryptographic protocol exchanges:

K_{ab}	A symmetric key shared between A and B
K_a	The public key of A
K_a^{-1}	The private key of A
$\{m\}K_{ab}$	m encrypted with the key K_{ab}
$\{m\}K_a$	m encrypted with the public key K_a
$\{m\}K_a^{-1}$	m signed with the private key K_a^{-1}
$A \rightarrow B : m$	A sends m to B
$A \rightarrow^* B : m$	A sends m to B over a secure channel
$H(m)$	A one-way hash function applied to m

3.2 Exchange of the User's Key

There are three methods of exchanging a user's key between a user and a CA, depending on whether the key was initially generated by the user's Enrollment Agent, by the CA, or by a third party.

3.2.1 User Generates Key

Firstly, the Enrollment Agent generates an asymmetric key pair. The private key is stored locally and should never be divulged to any other entity. The public key must be transmitted to the CA in such a way that the CA can be sure that the key has not been modified in transit, and that the user requesting the certificate is the same person who caused the key to be generated.

The Enrollment Agent then generates a *prototype certificate* in the format of ISO 9594-8, but with the issuer and subject names equal and the digital signature generated using the user's private key. This digital signature *cannot* be used to provide data origin authentication or integrity, as at this stage no-one has obtained the public key in a trusted manner. Instead, the signature is used to demonstrate that the public key is a valid one (not all bit patterns are valid) and that the corresponding private key is known to someone (although it provides absolutely no evidence about *who* knows it).

The prototype certificate is then sent electronically to a Certification Authority, either by placing it in the user's Directory entry, by sending it in an X.400(88) message, or by some other means. At this point the CA cannot act on the prototype certificate, as it has no evidence that the request is legitimate.

A standard certificate request form is then printed out on paper, and the user signs it with a traditional, manual, signature. The certificate request form contains all the information contained in the certificate, and a cryptographic hash function (eg. MD5) computed on the certificate body. The hash is not signed using any public key, as there is no benefit to be gained by doing so.

The user then takes the certificate request form to the CA, along with whatever proof of identify is required by the CA policy (eg. a passport or driver's licence, and a letter from the organisation's personnel department confirming that the named person is employed by them).

The CA checks the validity of the physical documents presented, and checks that the information stored electronically (in the prototype certificate) matches that on the certificate request form. A convenient way to do this is to compute a cryptographic hash on the electronic information and compare it with the hash printed on the form.

If the CA accepts the request, it signs a certificate and distributes it electronically to the user, by placing it in the user's Directory entry, by sending it in an X.400(88) mail message, or by some other means.

The CA retains the manually signed certificate request form for the purposes of accountability, in case the user later attempts to deny having requested a certificate containing that information.

Abstract Protocol

$$U \rightarrow C : \{U, K_u, t_1, t_2, r_1\}_{K_u^{-1}}$$

$$U \rightarrow^* C : U, H(U, K_u, t_1, t_2, r_1)$$

$$C \rightarrow U : \{U, K_u, t_1, t_2, r_2\}_{K_c^{-1}}$$

3.2.2 CA Generates Key

Firstly, the user submits a written request for a certificate to the certification authority, using a standard form. The CA checks the documentation submitted by the user as evidence of identity (eg. passport or driver's licence as above), and if it is acceptable proceeds to generate a certificate.

The CA generates an asymmetric key pair for the user and signs a certificate containing the user's name and public key. This certificate needs no further integrity or confidentiality protection, and so may be 'published' by storing it in the Directory Service. If the CA does not have permission to modify the user's Directory entry, the certificate may be transferred to the user by some other means (eg. X.400(88) email) and later entered into the Directory by the user themselves.

Note that is technically possible for the CA to masquerade as the user by using the newly generated private key, and hence modify the user's entry. As a matter of policy, CAs should not do this, as it undermines accountability: the user may claim that operations authenticated under their key were in fact performed by a malicious CA. To guard against this accusation, the CA software and physical procedures should prevent the CA operator from using a user's private key for any purpose.

The private key must then by transferred to the user while preserving its confidentiality. In principle, it is possible for the CA to give the user a print-out of their private key, which they must re-enter into their own computer. However, private keys are long and difficult to type, so this method is not acceptable. Instead, the CA can generate a much shorter key for a symmetric encryption algorithm (eg. DES), and electronically send the user their private key encrypted under this symmetric key. The user can then be given a print-out of this shorter key, which they can enter into their own system and hence recover their private key. The special stationery used by banks to advise customers of their ATM PINs would be suitable for this purpose.

At the end of the process, the CA should destroy all records of the value of the user's private key.

Abstract Protocol

$$U \rightarrow C : U$$

$$C \rightarrow U : \{U, K_u, t_1, t_2, r_1\}_{K_c^{-1}}$$

$$C \rightarrow U : \{K_u^{-1}\}_{K_{cu}}$$

$$C \rightarrow^* U : K_{cu}$$

3.2.3 Third Party Generates Key

With this approach, keys are generated by a special-purpose device known as a *Key Generation Box* (KGB). The only function performed by the KGB is to generate keys; it does not take part in any authentication exchanges, and it does not need to know how users are named. The KGB is assumed to be physically secure, and to be connected with a printer loaded with secure mailing stationary (such as that used by banks for informing customers of their PINs). It is also assumed that there is a connection such as a serial line between the KGB and a networked host computer.

Each time the KGB is invoked (eg. by a button being pressed) it generates both an asymmetric key pair and a symmetric key. It then increments a counter which holds the key serial number, and prints out (on the secure printer) the key serial number and the symmetric key. The key serial number, the public key and an encrypted version of the private key are transmitted to the host computer.

The user collects the printed envelope and takes it away. At some later point, she opens the envelope and reads the enclosed symmetric key and serial number. She then runs the Enrollment Agent on her own computer and types in the key and serial number. The Enrollment Agent contacts the host connected to the KGB and retrieves the public key and encrypted private key. It decrypts the private key, and checks that it is really the inverse of the public key. (This ensures that neither have been modified in transit). The private key is stored locally, and the public key is transferred to a CA using the procedure described in section 1 ("User Generates Key").

Abstract Protocol

$$G \rightarrow^* U : K_{gu}, r_1$$

$$G \rightarrow U : \{K_u^{-1}\}_{K_{gu}}, K_u, r_1$$

3.3 Exchange of the CA's Key

3.3.1 Using One-way Hash Functions

The protocol of section 3.2.1 may be used in reverse to transfer the CA's public key to the user. Firstly, the Certification Authority generates a prototype certificate for its own key and sends it to the user. As the CA will use exactly the same prototype certificate regardless of which user it is interacting with, it is sufficient for the CA to generate this prototype certificate once and store it in its own Directory entry. The user can then obtain it by reading it from the Directory.

As anyone can generate a prototype certificate containing the name of a CA, the user must obtain independent confirmation that the certificate she has obtained is the right one. To do this, the CA sends the user a cryptographic hash of the certificate's contents using a physically secure method. This hash value will be the same for all users, so the CA could include it on its headed notepaper or in its promotional literature, and give a paper copy to all users that come to be registered.

The user runs the Enrollment Agent on her own computer system and types in the CA's name and the hash value. The Enrollment Agent retrieves the prototype certificate from the Directory, recalculates the hash value and checks that it matches the value typed in. If it does match, the CA's name and public key is added to a local cache of known public keys.

Abstract Protocol

$$C \rightarrow U : \{C, K_c, t_1, t_2, r_1\}_{K_c^{-1}}$$

$$C \rightarrow^* U : H(C, K_c, t_1, t_2, r_1)$$

3.3.2 Using Symmetric Cryptography

If the user and CA have previously exchanged a symmetric key, then this can be used to transfer the CA's pubic key to the user. This will be the case if the CA has generated the user's private key and transferred it using the protocol of section 3.2.2.

In addition to the symmetric key, the user must obtain the CA's prototype certificate and a message authentication code, computed by applying the symmetric key to the body of the prototype certificate.

The CA will use the same prototype certificate in all such interactions, and so can broadcast its value to all users by storing it in the CA's Directory entry. The message authentication code will be different for each user, and so the CA must send this in a separate message to each user.

Abstract Protocol

$$C \rightarrow U : \{C, K_c, t_1, t_2, r_1\}_{K_c^{-1}}$$

$$C \rightarrow^* U : K_{cu}$$

$$C \rightarrow U : \{C, K_c, t_1, t_2.r_1\}_{K_{cu}}$$

3.4 Exchange of Keys between CAs

When CAs cross-certify each other, they must also exchange keys in a secure manner. As in the user to CA case, at least one key or hash value must be exchanged in a physically secure manner to start the process.

The protocol of section 3.3.1 ("Using hash functions") is suitable for exchanging keys between CAs. The protocol of section 3.2.2 ("CA Generates Key") will not generally be used during cross-certification of CAs, as CAs will almost always prefer to generate their own keys. There are several reasons for this. Firstly, the compromise of a CA has a far greater impact on security than the compromise of a user's key. The potential for a malicious CA operator to compromise another's key will usually be unacceptable when the certified key is that of a CA, although it may be acceptable for user keys. Secondly, the usual reason for preferring the second protocol for user keys is that the users computer system may not have special hardware support for random number generation, while that of the CA does. In the case of one CA certifying another, both parties have the necessary hardware and there is no reason for the subordinate CA to generate its own key. Thirdly, a CA will often be certified by more than one other CA. These CAs cannot all generate the single key used by the certified CA, and so the first protocol must be used.

3.5 Realisation of Abstract Protocols

Before they can can implemented, the abstract protocols of the previous section must be expressed in terms of real communication protocols.

3.5.1 Realisation using the Directory Service

The transfer of the CA's prototype certificate in sections 3.3.1 and 3.3.2 may be realised by storing the prototype certificate in the *caCertificate* attribute of the CA's Directory entry.

3.5.2 Realisation using X.400(88)

In order to use X.400(88) for the cryptographic exchanges, both the user and the CA must have X.400 mailboxes. X.400 mail is addressed using *Originator Recipient Addresses* rather than Distinguished Names, so there must be some means of mapping between the two. This is done by storing the Originator Recipient Address in the *mhsORAddresses* attribute of the appropriate Directory entry. An attacker could cause messages to be misrouted by changing the value of this attribute. However, the protocols of the previous section are designed so that misrouting of messages may cause denial of service, but will never result in either the user or the CA accepting forged credentials.

The exchange of the user's prototype certificate in section 3.2.1 may be achieved by sending an X.400(88) message with the prototype certificate contained in the *originator-certificate* extension field.

3.5.3 Realisation using Privacy Enhanced Mail

The protocols of section 3.2.2 and 3.3.2 can be realised by using Privacy Enhanced Mail in symmetric key management mode, with the key K_{cu} used as an *Interchange Key* (IK).

Chapter 4

Procedures for Certificate Revocation

Although the information contained in a certificate was verified to be valid at the time it was signed, this information may cease to be true at some later time.

For example:

- The user's organisational affiliation (and hence her Distinguished Name) may change.
- The user's public key may change because the old private key has been lost.
- The user's public key may change because the old private key has been compromised.
- The user's signature algorithm may change because the old algorithm has been shown to be insecure.
- The user's key size may be increased to protect against advances in cryptanalysis and the increasing amount of computer power available to an attacker.
- The user may desire to be certified by a different CA.

The reasons for changing a certificate fall into three categories:

- Those where it is vital that the old certificate be cancelled (revoked) and a new one issued as soon as possible.
- Those where a new certificate must be issued as soon as possible, but it is acceptable for the old certificate to remain valid.
- Those where the cancellation of the old certificate and the issue of a new one can be delayed until a time that is convenient for the CA.

To accommodate this range of possibilities, there are two mechanisms for cancelling a certificate:

- Every certificate contains an expiry date. The certificate should not be accepted after this date.
- Each CA issues a signed *Revocation List* (RCL) containing the serial numbers of certificates which have been revoked.

4.1 Revocation Lists

4.1.1 Issuing Revocation Lists

Each CA should issue a new revocation list at frequent intervals, even if the list of revoked certificates has not changed. This is done so that applications retrieving a revocation list can be sure that it is the most recent one available. Each revocation list contains the time and date on which it was issued and the time and date on which its replacement should be issued. Applications using revocation lists should compare these two times against the current time. If these indicate that a more recent RCL is available, the application should attempt to obtain the more recent version. If this attempt fails, the certificate should be treated as if it were revoked and a warning should be sent to a security administrator.

CAs must ensure that a new revocation list is issued before the old one expires, as failure to do so results in all certificates ever issued by the CA being treated as if they were revoked.

4.1.2 Size of Revocation Lists

Care must be taken that revocation lists do not become extremely large, as this may adversely affect the performance of the Directory service. To reduce the size of revocation lists, revoked certificates can be removed from the revocation list when their original expiry date has been reached.

If certificates are issued with very long lifetimes, then those that are revoked will need to be kept on the revocation list for a very long time, and the revocation list will become unmanageably large. Conversely, if certificates are issued with too short a lifetime, then the CA will have to spend an unacceptable amount of time re-issuing certificates which have expired. We recommend that the lifetime of a certificate be one year in normal circumstances. If the CA has reason to believe that the information in the certificate will become invalid sooner, then it should set the expiry date to reflect this.

4.1.3 Format of Revocation Lists

Two formats have been defined for revocation lists, one by ISO 9594-8 and the other by Privacy Enhanced Mail. For the purposes of the PASSWORD project, we have chosen to use the PEM format, which has several advantages:

- It uses a far more compact encoding. The saving of the resources needed to transmit and store RCLs will be significant when the number of revoked certificates becomes large.

- Only one digital signature operation is need to sign a PEM RCL, but two are needed to sign an ISO 9594-8 RCL.

- The ISO 9594-8 RCL does not contain an indication of when the next RCL is to be issued, so this information would have to communicated by some out of band means (such as requiring that the time between RCLs be a well-known constant which is the same for all CAs).

- The ASN.1 description of a Revocation List in ISO 9594-8 is widely believed to contain an error [4].

4.2 Informing the CA of Key Changes

The CA must be informed when the user's name or key changes, so that it can revoked the old certificate and issue a new one. It is desirable to ensure the authenticity and integrity of these requests, to prevent an attacker causing someone else's certificate to be revoked. However, the reason for changing the user's public key might be that the corresponding private key has been lost, in which case it is not possible to use it to sign the request. The CA must be able to accept written requests for a certificate to be revoked. The authenticity of such a request can be established by physical means (such as comparing the signature on the revocation request form with the signature on the certificate request form), and the user will always be able to generate such a request no matter what has happen to their private key.

4.3 Informing the User of Key Changes

Similarly, the CA must be able to inform its clients when its own key changes. CA keys will be very well protected, and so the compromise of a CA's key should be a very rare event. The usual reason for changing a CA's key will be an increase in the key size or the adoption of an improved cryptographic algorithm. In these cases, it is not necessary that the old key be revoked immediately, and certificates can be issued under both keys during the transitional period. Transition to a new CA key is made easier if client applications can be given two keys for a CA and select the appropriate one to use when checking signatures.

Chapter 5

Random Number Generation

5.1 Pseudo-Random Number Generators

Most computers and computer languages are completely deterministic; for the same input, the same program will produce the same output. It is therefore impossible to create randomness purely by software, no matter how complex the program used. The "random number generation" subroutines available on many systems generate *pseudo-random*, not truly random, sequences; the numbers generated share many statistical properties with true random sequences, and hence may be used instead in many applications. Cryptography is one of the few applications where these conventional pseudo-random number generators are not acceptable.

Some pseudo-random number generation algorithms have the additional property of being *cryptographically strong*. They take as input a genuinely random seed value and produce a pseudo-random sequence based upon it. It is computationally infeasible to derive the value of unknown members of the pseudo-random sequence given only knowledge of other members. The output of these strong random number generators is still not genuinely random; it may be predicted by an attacker who knows the value of the seed or who has sufficient computing power to break the cryptosystem used. Provided that the initial seed is kept secret and the cryptosystem is strong enough, these special random number generators can be used in some cryptographic applications. The use of cryptographic RNGs does not eliminate the need for a hardware random number generator, as the initial genuinely random seed must be generated somehow. However, they do greatly reduce the number of genuinely random bits that are needed, and so are effective in speeding up slow RNGs.

5.2 Hardware Random Number Generators

To produce truly random numbers, a computer system must be extended by the addition of some hardware that behaves in a random manner. That is, the behaviour of the device as observed by the computer must be influenced by some underlying physical process in a way that cannot be predicted. Almost any peripheral (eg. discs, keyboards, network interfaces) could be used in this way, but with most it is difficult to *guarantee* the lack of predictability. For example, counting the number of ethernet packets received in each second gives an apparently random result, but is highly predictable by anyone monitoring the same ethernet. A better approach is to use a peripheral which has been specially designed to generate random numbers, and which has been carefully engineered to provide a maximum level of unpredictability.

A common approach used in many hardware random number generators (RNGs) is to take an electronic noise source (all electronic components generate some amount of noise), amplify it and then sample its value at regular intervals.

There are three main sources of electronic noise:

- Voltage noise. The thermal motion of electrons causes momentary voltage fluctuations across any conductor. Circuits which are physically hotter generate more voltage noise.
- Current noise. Electrons are the smallest possible unit of electrical charge, and any charge will be equal to an integer multiple of electrons. As a result, any electrical current is a statistical process; in a measuring interval sometimes more electrons than average will flow and sometimes less. In absolute terms, large currents exhibit bigger fluctuations than small ones. However, the fluctuations are proportional to the square root of the current, so smaller currents exhibit bigger fluctuations relative to the average flow.
- Inductive noise. All electrical components act as radio transmitters and receivers, picking up part of the signal from nearby circuits and from other parts of the same circuit.

The first two sorts of noise are ideal for cryptographic applications, as they have well-known statistical properties and are totally unpredictable by anyone not in physical contact with the component concerned.

Inductive noise is far less useful, for the following reasons:

1. An attacker's circuit might be able to pick up the same radio signals, enabling her to predict the number generated.
2. The statistical properties of the signal are highly dependent on the exact nature of nearby electrical apparatus. In computer rooms, the main component of these signals is usually the clock frequencies of nearby computers. The periodic nature of this signal can make it too easy to predict.

Thus, a good design for a RNG must introduce and amplify signals of the first two types, and at the same time take great care to exclude signals of the third type.

We have constructed and measured several RNG designs, some aimed at amplifying current noise and some aimed at amplifying voltage noise. Many designs were found to be highly susceptible to inductive pickup between the amplified signal and the intended source. When this happens, a periodic oscillation builds up which swamps the desired random signal with a much larger, highly predictable one. This effect can be reduced by careful attention to circuit layout and by the use of decoupling capacitors. Of the designs we tested, the one shown in figure 5.2 (supplied by Andrew Findlay) was found to be the most immune to pickup, even without particular care as to layout:

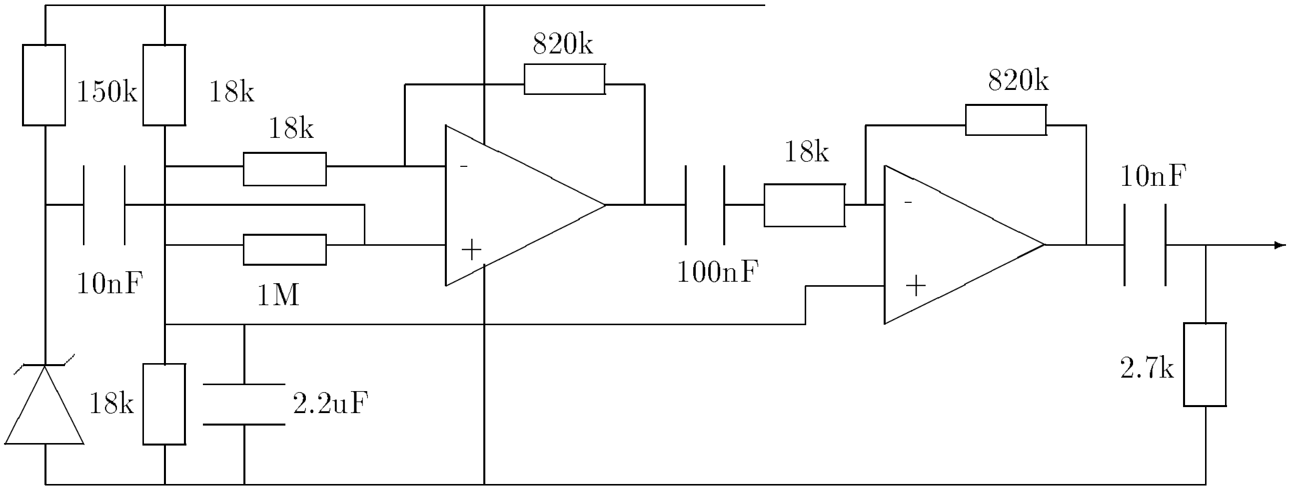

Figure 5.1: Random Number Generator

Our initial assessment of the quality of the random number generators was performed by visual examination of waveforms on an oscilloscope. We considered it highly desirable to have a test that was more objective and repeatable, and that was capable of rejecting generators that failed to be random in ways that were not visually apparent.

To do this, we needed a formal definition of which statistical distributions were acceptable. We adopted the following model: For some integers k and n, consider $k.n$ consecutive output bits forming a population of n k-bit long sequences. Then we define the (k, n)-diversity as the inverse of the probability that two samples from this population will be the same. The k-diversity is the limit of this as n tends to infinity.

The ideal noise source produces independently distributed bits, each of which is 1 with probability 1/2 and 0 with probability 1/2. This source has a k-diversity of 2^k; any other distribution will give a lower diversity.

The RNGs which we felt intuitively were poor can be objectively rejected on the grounds that they have too low a k-diversity. The importance of this measure can be justified on theoretical grounds by its close relation to the *index of coincidence* [10], an important tool in cryptanalysis.

5.3 Use for Key Generation

The security of public-key cryptosystems relies on the values of private keys not being known to potential attackers. To maintain this secrecy, there must be security mechanisms to prevent unauthorised access to stored keys; this is dealt with in report R2.6. Even if the protection of the storage area were infallible, an attacker could still determine the value of a key by knowing how the key was initially generated. One of the major advantages of Open Systems is that implementations can be bought by anyone from a variety of different manufacturers. As a result, all details of how the key generation software works can be readily determined by anyone who buys the software and examines it. For this reason, the key generation process must incorporate a genuinely random element that an attacker has no means of predicting.

There are two ways in which a random element could be introduced into the key generation process. Firstly, a new random input could be obtained for each key that is to be generated. Secondly, a single random element could be obtained once during system initialisation and used as the seed for a cryptographically strong pseudo-random number generator. Then each key could be generated from a different element of the cryptographically strong pseudo-random sequence.

The second approach is only applicable to situations where the same entity must generate many different keys. Thus, it is only useful with the "CA generates key" distribution method and not with the "User generates key" method. It has the advantage that far fewer genuinely random bits need to be generated. However, it is vulnerable to some additional risks:

- The cryptosystem used to generate pseudo-random numbers might be broken, enabling an attacker with knowledge of a single secret key to compute all secret keys issued by the same authority. Using this method makes the overall system vulnerable to mathematical attacks on three different cryptosystems (the cryptographic pseudo-random number generator, the one way hash function and the public key signature algorithm) rather than just two. For this reason, it may be desirable to use a RNG whose security has been mathematically proven to be equivalent to the security of one of the other two algorithms employed.

- If an attacker manages to obtain the seed value, then she will be able to calculate all secret keys that were ever generated. Although the seed value will be heavily protected, it might be disclosed by a CA operator who had been bribed or threatened, or by someone who had stolen the physical hardware used for key generation [13].

  To prevent this attack, it is desirable to use a cryptographic RNG algorithm that does not require the original seed value to be retained, but instead updates a stored value after the generation of each key, using a one-way function. After each member of the sequence is generated, the information that was needed to calculate it can be destroyed so that it may never again be recovered by anyone.

5.4 Evaluation of Random Systems

If a system is entirely deterministic, then it is possible to specify what its behaviour should be for particular inputs, and to test that it behaves according to the specification. If a process includes genuinely random elements, then its behaviour will not be repeatable. This causes problems with both the specification and the evaluation of systems.

For this reason, it is highly desirable to separate systems into deterministic and non-deterministic components, with the interface between the two being accessible during testing. The deterministic components can then be completely specified, and their observed behaviour compared with the specification. The random components should be made as simple as possible, in order to reduce the chance of their containing an undetected fault. Ideally, the random component of the system should only produce a random bit stream, and do nothing else. The statistical diversity tests of section 5.2 can then be used to help verify correct operation.

Appendix A

Deficiencies of the X.509 Certificate and Revocation List Formats

It is widely recognised that the certificate and revocation list formats described in ISO 9594-8 contain serious deficiencies which ought to be corrected. A liason statement between the IST 21/1/1 and IST 33 subcommitees of the British Standards Institute identified the following problems [4]:

A.1 Certificates

- In the event that DistinguishedNames are re-used over time, the certificate should also contain a unique numeric identifier to indicate which holder of the name was intended to be the certificate subject. (This will be added in ISO 9594-8 (1992)).
- In the event that different keys are used for integrity and confidentiality, the certificate should include multiple keys, and indicate which security services each key may be used for.
- It is not currently possible to distinguish user certificates from CA certificates. Thus, there is a risk of unauthorised users posing as certification authorities.
- When the certificate is a CA certificate, there should be a means for the issuer to indicate which part of the name-space has been delegated to the subject.
- The algorithm identifier used to identify the signature algorithm should also identify the encoding rules which were used to translate the presentation data value to be signed into a sequence of octets for input to the signature algorithm.
- When a certificate is stored, the concrete syntax used to encode the certificate should be held with it.

A.2 Revocation Lists

- The SIGNED SEQUENCE OF SEQUENCE should be replaced by SEQUENCE OF SIGNED SEQUENCE. This would allow individual revoked certificates to be signed. This is particularly useful when non-repudiation of the revocation is required. (See Directory defect report 9594/033).
- If there is a difference between the date at which a user requested the CA to revoke their certificate and the date at which the CA carried out the revocation, the revoked certificate should contain both dates. (See SC21 N1734, summary of voting on extensions to 9594-8, AFNOR comments).
- There is no indication of when the next revocation list will be issued. Without this information, it is difficult to tell whether the revocation list is the most recent. (For example, an attacker knowing a compromised key and revoked key might try to make the key valid again by replacing the current revocation list with an old one).
- The CertificateList structure is liable to become very large. It would be useful to have a more compact format.

Within the PASSWORD project, we have chosen to circumvent the above problems with revocation lists by using the alternative format defined by Privacy Enhanced Mail [9].

We avoid the problems with certificates by imposing strict rules on which names may be used for authorities and users. If in the future ISO 9594-8 is extended to overcome the identified problems, then it will longer be necessary to impose these inconvenient restrictions.

Appendix B

New Attributes and Object Classes

We have chosen to use the certificate revocation list format defined by Privacy Enhanced Mail, rather than that defined by ISO 9594-8. In order to store PEM revocation lists in the Directory Service, it is necessary to define some new attributes and object classes. We intend to register these as extensions to the COSINE/Internet naming architecture [1]. At the time of writing, these definitions have not been registered and so have not been assigned unique object identifier values.

```
internetCertificationAuthority OBJECT-CLASS
    SUBCLASS OF top
    MUST CONTAIN {
        caCertificate,
        internetCertificateRevocationList,
        internetAuthorityRevocationList }
    MAY CONTAIN {
        crossCertificatePair }
    ::= { pilotObjectClass x }  -- not yet assigned
```

Figure B.1: New Object Classes

```
IMPORTS CertificateRevocationList FROM PrivacyEnhancedMail;

internetCertificateRevocationList ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX CertificateRevocationList
    ::= { pilotAttributeType y } -- not yet assigned

internetAuthorityRevocationList ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX CertificateRevocationList
     ::= { pilotAttributeType z } -- not yet assigned
```

Figure B.2: New Attributes

Appendix C

Z Model of Certificate Generation Unit

This section uses the Z notation [12] to construct a formal model of the Constrained Data Items, Integrity Verification Procedures and Transformation Procedures implemented by a Certificate Generation Unit.

It is possible to construct formal proofs of some properties of Z specifications. In particular, it is possible to prove that Clark-Wilson conditions [C1], [C2] and [C5] have been met. It is fairly clear that the following specifications meet condition [C4] as well, but the way they have been structured makes this difficult to prove formally. The separation of duty condition [C3] is not clearly defined in the original paper by Clark [6], and for the purposes of this section it will be ignored.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{Keys}} \\
\hline
secretKey : HASH \rightarrow SIGNATURE \\
publicKey : SIGNATURE \rightarrow HASH \\
\hline
secretKey = publicKey^{-1} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{SerialNumbers}} \\
\hline
issuedSerialNumbers : \mathbb{P}(\mathbb{N}) \\
revokedSerialNumbers : \operatorname{seq} \mathbb{N} \\
revocationDates : \operatorname{seq} DATE \\
\hline
\operatorname{ran}(revokedSerialNumbers) \subset issuedSerialNumbers \\
\operatorname{dom}(revokedSerialNumbers) = \operatorname{dom}(revocationDates) \\
\hline
\end{array}
$$

Figure C.1: Constrained Data Items

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{IssuedAuditTrail}} \\
\hline
subjectNameRecord : \mathbb{N} \pfun NAME \\
issuedSerialNumberRecord : \mathbb{N} \pfun \mathbb{N} \\
startDateRecord : \mathbb{N} \pfun DATE \\
endDateRecord : \mathbb{N} \pfun DATE \\
issueDate : \mathbb{N} \pfun DATE \\
issuedAuditRecordNumber : \mathbb{N} \\
\hline
\mathrm{dom}(subjectNameRecord) = 1..issuedAuditRecordNumber \\
\mathrm{dom}(issuedSerialNumberRecord) = 1..issuedAuditRecordNumber \\
\mathrm{dom}(startDateRecord) = 1..issuedAuditRecordNumber \\
\mathrm{dom}(endDateRecord) = 1..issuedAuditRecordNumber \\
\mathrm{dom}(issueDate) = 1..issuedAuditRecordNumber \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{RevokedAuditTrail}} \\
\hline
revokedSerialNumberRecord : \mathbb{N} \pfun \mathbb{N} \\
revocationDate : \mathbb{N} \pfun DATE \\
revokedAuditRecordNumber : \mathbb{N} \\
\hline
\mathrm{dom}(revokedSerialNumberRecord) = 1..revokedAuditRecordNumber \\
\mathrm{dom}(revocationDate) = 1..revokedAuditRecordNumber \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{RevocationListAuditTrail}} \\
\hline
rclDate : \mathbb{N} \pfun DATE \\
rclAuditRecordNumber : \mathbb{N} \\
\hline
\mathrm{dom}(rclDate) = 1..rclAuditRecordNumber \\
\hline
\end{array}
$$

Figure C.2: Audit Trail

$$
\begin{array}{|l}
\multicolumn{1}{l}{\underline{\quad InitialState \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}} \\
\Delta Keys \\
\Delta SerialNumbers \\
\Delta IssuedAuditTrail \\
\Delta RevokedAuditTrail \\
MakeKeyPair \\
\hline
secretKey' = secretKey! \\
publicKey' = publicKey! \\
issuedSerialNumbers' = \{\} \\
revokedSerialNumbers' = \{\} \\
revocationDates' = \{\} \\
issuedAuditRecordNumber' = 0 \\
revokedAuditRecordNumber' = 0 \\
subjectNameRecord' = \{\} \\
issuedSerialNumberRecord' = \{\} \\
startDateRecord' = \{\} \\
endDateRecord' = \{\} \\
issueDate' = \{\} \\
\hline
\end{array}
$$

Figure C.3: Integrity Verification Procedure

$$
\begin{array}{|l}
\multicolumn{1}{l}{\underline{\quad Sign \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}} \\
hash! : HASH \\
signature? : SIGNATURE \\
\Xi Keys \\
\hline
signature? = secretKey(hash!) \\
\hline
\end{array}
$$

Figure C.4: Underlying Procedures

$$
\begin{array}{l}
\textbf{AuditIssueCertificate} \\
\hline
\Delta IssuedAuditTrail \\
\Xi TimeOfDay \\
\hline
issuedAuditRecordNumber' = issuedAuditRecordNumber + 1 \\
subjectNameRecord' = subjectNameRecord' \cup (issuedAuditRecordNumber \mapsto subjectName!) \\
issuedSerialNumberRecord' = issuedSerialNumberRecord' \cup \\
(issuedAuditRecordNumber \mapsto serialNumber?) \\
startDateRecord' = startDateRecord \cup (issuedAuditRecordNumber \mapsto startDate!) \\
endDateRecord' = endDateRecord \cup (issuedAuditRecordNumber \mapsto endDate!) \\
issueDate' issueDate \cup (issuedAuditRecordNumber \mapsto currentTime) \\
\hline
\end{array}
$$

$$
\begin{array}{l}
\textbf{AuditRevokeCertificate} \\
\hline
\Delta RevokedAuditTrail \\
\Xi TimeOfDay \\
\hline
revocationAuditRecordNumber' = revocationAuditRecordNumber + 1 \\
revokedSerialNumberRecord' = revokedSerialNumberRecord \cup \\
(revocationAuditRecordNumber \mapsto serialNumber!) \\
revocationDate' = revocationDate \cup (revocationAuditRecordNumber \mapsto currentTime) \\
\hline
\end{array}
$$

$$
\begin{array}{l}
\textbf{AuditIssueRCL} \\
\hline
\Delta RevocationListAuditTrail \\
\Xi TimeOfDay \\
\hline
rclAuditRecordNumber' = rclAuditRecordNumber + 1 \\
rclDate' = rclDate \cup (rclAuditRecordNumber \mapsto currentTime) \\
\hline
\end{array}
$$

Figure C.5: Audit Procedures

$$
\begin{array}{|l}
\hline
\textit{MakeKeyPair} \\
secretKey! : HASH \rightarrow SIGNATURE \\
publicKey! : SIGNATURE \rightarrow HASH \\
randomNumber? : RANDOM \\
\hline
secretKey! = publicKey!^{-1} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{IssueCertificate} \\
subjectName! : NAME \\
subjectPublicKeyInfo! : KEY \\
startDate! : TIME \\
endDate! : TIME \\
serialNumber? : \mathbb{N} \\
signature? : SIGNATURE \\
\Delta SerialNumbers \\
Sign \\
AuditIssueCertificate \\
\hline
hash! = hashCertificate(issuerName, subjectName!, subjectPublicKeyInfo!, \\
startDate!, endDate!, serialNumber?) \\
serialNumber \notin issuedSerialNumbers \\
issuedSerialNumbers' = issuedSerialNumbers \cup serialNumber \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{RevokeCertificate} \\
serialNumber! : \mathbb{N} \\
\Delta SerialNumbers \\
\Xi TimeOfDay \\
AuditRevokeCertificate \\
\hline
serialNumber \in issuedSerialNumbers \\
revokedSerialNumbers' = revokedSerialNumbers \frown \{serialNumber\} \\
revocationDates' = revocationDates' \frown \{currentTime\} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{IssueRevocationList} \\
signature? : SIGNATURE \\
\Xi SerialNumbers \\
\Xi TimeOfDay \\
AuditIssueRCLSign \\
\hline
hash! = hashRCL(revokedSerialNumbers, revocationDates, currentTime) \\
\hline
\end{array}
$$

Figure C.6: Transformation Procedures

Appendix D

Z Model of Certificate Verification Process

Applications which use certificates for security-relevant activities must contain functions to check that these certificates obey the rules specified in the application's Technical Security Policy. This enforcement is performed by the application, not the Certificate Generation Unit, and so its specification is not part of the CGU Security Target. However, we believe that many different applications will have exactly the same requirements for checking certification paths. This section models the certificate verification process using the Z notation [12].

In general, a certification path consists of a chain of certificates from the verifier to the claimant. When the security policy defines a hierarchy of CAs, each CA certificate may be classified as a forward, reverse, or cross-certificate, depending on whether the issuer is above, below or on the same level as the subject in the hierarchy. A *Reverse Certification Path* is a fragment of a certification path containing only reverse certificates, while a *Forward Certification Path* is a fragment containing only forward certificates.

For the purposes of the policy defined here, CAs are divided into the hierarchy described in section 1.2.3. To be acceptable, a certification path must consist of a *Reverse Certification Path* from the verifier up to a *Point of Common Trust* (PCT), and a *Forward Certification Path* from the PCT down to the claimant. The PCT may be either an organisational, policy or top-level CA.

In general, the claimant in an authentication exchange will be obliged to construct the forward certification path and pass it to the verifier. The verifier can then combine this with a locally cached reverse certification path to form the complete path. If the provided forward certification path is incomplete or unacceptable, the verifier may consult the Directory Service and attempt to construct an alternative forward path. However, doing so may involve expending substantial resources in respect of an access request that may eventually turn out to be initiated by a malicious intruder. Therefore, this policy permits the verifier to immediately reject access requests which are not accompanied by a complete forward path.

The specification of procedures for the construction of reverse certification paths is considered to be a local matter, and is not included in the formal model. Different implementations may adopt different strategies provided that this does not compromise security. However, the strategy used by particular implementation should be documented. Typically, the public key of a TLCA will be conveyed to the application by a local trusted mechanism, such as being typed into a file by a security administrator. Sites with many computer systems will need a more automated mechanism for this key distribution that does not compromise security.

The certificates in a forward certification path must follow the naming restrictions described in section 1.2.3. The path verifying software must reject as invalid all forward certificates that do not conform these rules. It is permitted for implementations to support a more general security policy, provided that they may be easily configured to reject all certificates disallowed under the naming rues of section 1.2.3.

$$
\begin{array}{|l}
\hline
\textit{KeyRecords} \\
\textit{authorityName} : \mathbb{N} \nrightarrow \textit{NAME} \\
\textit{publicKey} : \mathbb{N} \nrightarrow \textit{KEY} \\
\textit{startDate} : \mathbb{N} \nrightarrow \textit{DATE} \\
\textit{endDate} : \mathbb{N} \nrightarrow \textit{DATE} \\
\textit{indices} : \mathbb{P}(\mathbb{N}) \\
\hline
\mathrm{dom}(\textit{authorityName}) = \textit{indices} \\
\mathrm{dom}(\textit{publicKey}) = \textit{indices} \\
\mathrm{dom}(\textit{startDate}) = \textit{indices} \\
\mathrm{dom}(\textit{endDate}) = \textit{indices} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{TopLevelCAs} \\
\textit{topLevelCAs} : \mathbb{P}(\mathbb{N}) \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{PolicyCAs} \\
\textit{policyCAs} : \mathbb{P}(\mathbb{N}) \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{OrganisationalCAs} \\
\textit{organisationalCAs} : \mathbb{P}(\mathbb{N}) \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{Users} \\
\textit{users} : \mathbb{P}(\mathbb{N}) \\
\hline
\end{array}
$$

Figure D.1: Constrained Data Items

$$
\begin{array}{|l}
\hline
\textit{AcceptOrganisationalCACertificate} \\
\Delta \textit{OrganisationalCAs} \\
\Xi \textit{PolicyCAs} \\
\textit{CheckCertificate} \\
\hline
\textit{index} \in \textit{policyCAs} \\
\textit{organisationalCAs}' = \textit{organisationalCAs} \cup \{\textit{newIndex}\} \\
\textit{nameType}(\textit{subject}!) = \textit{caNameType} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{AcceptOrganisationalUnitCACertificate} \\
\Delta \textit{OrganisationalCAs} \\
\textit{CheckCertificate} \\
\hline
\textit{index} \in \textit{organisationalCAs} \\
\textit{organisationalCAs}' = \textit{organisationalCAs} \cup \{\textit{newIndex}\} \\
\textit{issuer}! > \textit{subject}! \\
\textit{nameType}(\textit{subject}!) = \textit{caNameType} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{AcceptUserCertificateFromOrgCA} \\
\Delta \textit{Users} \\
\Xi \textit{OrganisationalCAs} \\
\textit{CheckCertificate} \\
\hline
\textit{index} \in \textit{organisationalCAs} \\
\textit{users}' = \textit{users} \cup \{\textit{newIndex}\} \\
\textit{issuer}! > \textit{subject}! \\
\textit{nameType}(\textit{subject}!) = \textit{userNameType} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{AcceptUserCertificateFromPolicyCA} \\
\Delta \textit{Users} \\
\Xi \textit{PolicyCAs} \\
\textit{CheckCertificate} \\
\hline
\textit{index} \in \textit{policyCAs} \\
\textit{users}' = \textit{users}\{\textit{newIndex}\} \\
\textit{nameType}(\textit{subject}!) = \textit{userNameType} \\
\hline
\end{array}
$$

$\textit{AcceptUserCertificate} \mathrel{\widehat{=}} \textit{AcceptUserCertificateFromPolicyCA} \lor$
$\textit{AcceptUserCertificateFromOrgCA}$

Figure D.2: Transformation Procedures

$$
\begin{array}{|l}
\hline
\multicolumn{1}{l}{\textit{VerifySignature}} \\
issuerPublicKey! : SIGNATURE \rightarrow HASH \\
hash! : HASH \\
signature! : SIGNATURE \\
\hline
issuerPublicKey!(signature!) = hash! \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\multicolumn{1}{l}{\textit{CheckCertificate}} \\
issuer! : NAME \\
subject! : NAME \\
startDate! : DATE \\
endDate! : DATE \\
subjectPublicKey! : SIGNATURE \rightarrow HASH \\
signature! : SIGNATURE \\
index : \mathbb{N} \\
VerifySignature \\
AddKeyRecord \\
\hline
authorityName(index) = issuer! \\
issuerPublicKey! = publicKey(issuer!) \\
hash! = hashCertificate(issuer!, subject!, subjectPublicKey!, ...) \\
newStartDate! > startDate! \\
newStartDate! > startDate(index) \\
newEndDate! < endDate! \\
newEndDate! < endDate(index) \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\multicolumn{1}{l}{\textit{AddKeyRecord}} \\
newEndDate! : DATE \\
newStartDate! : DATE \\
subjectPublicKey! : SIGNATURE \rightarrow HASH \\
subject! : NAME \\
newIndex : \mathbb{N} \\
\Delta KeyRecords \\
\hline
newIndex \notin indices \\
indices' = indices \cup \{newIndex\} \\
endDate' = endDate \cup (newIndex \mapsto newEndDate!) \\
startDate' = startDate \cup (newIndex \mapsto newStartDate!) \\
publicKey' = publicKey \cup (newIndex \mapsto subjectPublicKey!) \\
authorityName' = authorityName \cup (newIndex \mapsto subject!) \\
\hline
\end{array}
$$

Figure D.3: Underlying Procedures

Appendix E

BAN Logic Model of Certificate Verification

Some of the integrity requirements cannot be expressed using the Z notation. This section uses the Burrows-Abadi-Needham notation [5] to formalise these additional requirements.

Let V be the verifier. Then for each authority A_i and key K_i in *KeyRecords*, we require that V believes K_i to be A_i's key. Formally:

$$V \models K_i \rightarrow A_i$$

We also require that the sets *topLevelCAs*, *policyCAs*, *organisationalCAs* and *users* correctly represent V's trust model. That is, if the index of a particular key record belongs to one of these sets, then V trusts the associated CA to issue certificates of a particular form. Formally:

$$\forall A \in J_i : V \models A_i \Rightarrow K \rightarrow A$$

$$\forall A \in J_i, \forall B : V \models A_i \Rightarrow B \Rightarrow K \rightarrow A$$

(Where the set J_i is defined by the naming rules).

These variables are initialised by the user or system administrator providing information about what they are prepared to trust. By hypothesis, the above equations are true in this initial state.

It remains to be shown that the application of a Transformation Procedure always results in a state where these equations are still true.

Each Transformation Procedure explicitly checks that the naming rules have been followed, that is $A \in J_i$. They also check that the pre-condition *CheckCertificate* is met. This ensures that the certificate signature is valid and the certificate is in date:

$$V \triangleleft \{t, K \rightarrow A, \forall B \in J : A \Rightarrow K' \rightarrow B\} K_i$$

$$\# t$$

(Where the set J is defined by the naming rules).

The rules of the BAN logic allow us to conclude from this that the verifier believes the name and key contained in the certificate, and believes that the certified authority is of the indicated type.

$$V \models K \rightarrow A$$

$$\forall B \in J : V \models A \Rightarrow K' \rightarrow B$$

$$\forall B \in J, \forall C : V \models A \Rightarrow C \Rightarrow K' \rightarrow B$$

That is, the integrity equations also hold for the new values of the constrained data items.

References

[1] P. Barker and S. Kille. *RFC 1274 : The COSINE and Internet X.500 Schema*. University College London, November 1991.

[2] D. E. Bell and L. J. LaPadula. Secure computer systems: Unified exposition and multics interpretation. Technical Report MTR 2997, The Mitre Corporation, March 1976.

[3] K. J. Biba. Integrity considerations for secure computer systems. Technical Report MTR 3153, The Mitre Corporation, April 1977.

[4] British Standards Institution, IST 21/1/1. *Liaison Statement to SC33 on Certificates*, March 1992.

[5] M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. Technical Report 39, DEC SRC, Palo Alto, CA., February 1989.

[6] D. Clark and D. R. Wilson. A comparison of commercial and military computer security policies. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 1987.

[7] Commission of the European Communities, Directorate-General XIII. *Information Technology Security Evaluation Criteria (ITSEC)*, 1991.

[8] International Standards Organization. *ISO 9594-8 : The Directory : Authentication Framework*, 1988.

[9] S. Kent and J. Linn. *RFC 1114: Privacy Enhancement for Internet Electronic Mail: Part II: Certificate Based Key Management*. IAB Privacy Task Force, 1989.

[10] A. Konheim. *Cryptography: A Primer*. John Wiley and Sons, 1981.

[11] National Computer Security Center. *Trusted Computer System Evaluation Criteria*, December 1985.

[12] J. M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall International, 1988.

[13] H. O. Yardley. *The American Black Chamber*. Bobs-Merrill Company, 1931.