PKI standards

Chris Mitchell
Information Security Group, Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK
c.mitchell@rhul.ac.uk

14th November 2000

Abstract

This paper provides a review of the current state of the art in standards for Public Key Infrastructures. The main focus of the paper is the recent work by the Internet Engineering Task Force, ITU-T, and ISO/IEC.

1 Introduction

1.1 The growth of PKIs

Recent years have seen rapid growth in the number and scope of standards dealing with aspects of Public Key Infrastructures (PKIs). This has primarily been fuelled by the much increased interest in implementing PKIs, which is itself largely a result of the development of commercial and wider public use of the Internet, not least for e-commerce activities.

The growth of e-commerce in particular has raised awareness of some of the security issues involved in the use of the Internet. To enable purchasers and merchants to conduct their transactions securely, there is a need for them to interwork securely. This will typically involve supporting secure communications between entities who have no prior formal relationship. PKIs are widely seen as a solution to the key management problems arising from this type of need.

With the growth in awareness of, and requirements for, PKIs, there has been a parallel increase in development effort devoted to standardising all aspects of PKIs. The potential benefits are clear, including the possibility of large scale interworking between PKIs, and lower costs through economies of scale and increased competition. In this paper we provide an introduction to some of the more prominent PKI standards, including some which are still under development.

1.2 Standards bodies

PKI standardisation has been carried out by a number of different standards making bodies. In general efforts have been made to harmonise the results of this work. Of particular note are the following.

- *ITU-T*. This body led the way in PKI standardisation with the publication in 1988 of the first edition of the X.509 Recommendation, [32], providing a standardised format for public key certificates and Certificate Revocation Lists (CRLs). In collaboration with ISO/IEC, ITU-T has continued to develop the X.509 recommendation (see [34]).
- *Internet Engineering Task Force (IETF)*. The IETF is responsible for the standards governing the operation of the Internet. Over the last five years or so,

the IETF has been developing a suite of standards governing operation of an X.509-based PKI for use across the Internet. This suite of standards has the title PKIX. Note that the Internet draft, [15], provides a useful general introduction to PKIX.

- *ISO/IEC*. Apart from the joint work with ITU-T on X.509 (see [18]), ISO/IEC has also been developing a series of more general PKI-related standards, including standards on certificate management and time-stamping services.

1.3 Types of PKI standards

Before considering individual PKI-related standards, we first consider what types of issue these standards deal with. That is, we consider which areas of operation of a PKI are covered by standards.

- The most fundamental issue for a PKI is the format of the public key certificates. The development of standards in this area is discussed in more detail in Section 2.
- Certificate management is a term used to cover a range of different types of interaction between Certification Authorities (CAs) and their clients, including issues such as initial registration, certificate requests, and revocation requests. Standards for certificate management are the focus of Section 3.
- Once certificates have been created, there is a need for certificate users to be able to retrieve them from where they are stored; this is covered in Section 4.
- Once certified, there may be a need to withdraw a public key from use prior to the expiry date specified in the certificate – this is known as revocation. As a result there is a need for a certificate user to be able to determine the status of a certificate, i.e. whether it has been revoked. Standards covering revocation and certificate status are discussed in Section 5.
- One of the main reasons for the introduction of time-stamping services has been to support long-term use of signed documents, and in particular to enable a digitally signed document to have validity after the expiry or revocation of the public key. Standards covering such services are considered in Section 6.
- There are a variety of other Trusted Third Party (TTP) services relevant to PKIs, and some of the recent standards developments in this area are the focus of Section 7.
- Finally, underlying any effective use of public key certificates must be an understanding of what a certificate means. This is dependent on the Certificate policy and certification practice statements of the CA. Standards covering this issue are considered in Section 8.

2 Certificate standards

2.1 The development of public key certificate standards

The first work on developing PKI standards predates the term PKI by some years. What has become known as the X.509 certificate format was first standardised in 1988, [32], as part of the first edition of the ITU-T X.500 directory services recommendations (note that they were then referred to as CCITT recommendations). The first edition of an aligned ISO/IEC standard, ISO 9594-8, was published a couple

of years later. Two subsequent versions of aligned ITU-T recommendations and ISO/IEC standards have been subsequently published, and to avoid confusion the three different certificate formats defined in these documents are referred to as X.509 versions 1, 2 and 3 certificates.

The original work on X.509 was performed as part of the development of the X.500 directory series recommendations. The main initial 'customer' for the standardised public key certificates were the parallel X.400 series of recommendations specifying the operation of an email system. The 1988 version of the X.400 standards incorporated a large range of security features, the key management for which was based round the use of X.509 certificates. Interestingly, while the X.400 recommendations have hardly set the world alight, the X.509 public key certificate format dominates the field.

After the publication of the first edition of the X.509 recommendation, the next main customer for the X.509 certificate format was again a secure email system – this time the Internet Privacy Enhanced Mail (PEM) system, [5]. This system again used X.509 certificates as the basis of its key management – however, a number of additional certificate features were required by PEM which were incorporated into the X.509 version 2 certificate format, [33]. Subsequent growing interest in deploying X.509 PKIs revealed the need for further additions to the certificate format, and these have been incorporated into the version 3 certificate format, [18], [34].

2.2 X.509 certificates

We now consider in a little more detail the main elements in a certificate constructed according to the X.509 version 3 standard, [18], [34]. The X.509 certificate format is specified in a language called Abstract Syntax Notation One (ASN.1), [35], [36], [37], [38]. ASN.1 is widely used for the specification of ITU-T and ISO (and other standard) communication protocols. The purpose of ASN.1 is to have a standardised and platform independent language with which to express data structures, and to have a standardised set of rules for the transformation of values of a defined type into a stream of bytes. This stream of bytes can then be sent on a communication channel set up by the lower layers in the stack of communication protocols, e.g. TCP/IP, or encapsulated within UDP packets. As a result, two different applications written in two completely different programming languages running on different computers with different internal representations of data can exchange instances of structured data types. This frees the programmer from a great deal of work, since no code has to be written to process the transport format of the data.

When the first ITU-T recommendation on ASN.1 was released in 1988, it was accompanied by the *Basic Encoding Rules* (BER) as the only option for encoding. BER is a somewhat verbose protocol. It adopts a so-called TLV (type, length, value) approach to encoding, in which every element of the encoding carries some type information, some length information and then the value of that element. Where the element is itself structured, then the Value part of the element is itself a series of embedded TLV components, to whatever depth is necessary. In summary BER is not a compact encoding but is fairly fast and easy to produce.

The Basic Encoding Rules come in three variants: BER – which allows options for the encoder, DER (*Distinguished Encoding Rules*) – which resolves all options in a particular direction, and CER (*Canonical Encoding Rules*) – which resolves all

options in the other direction, [39]. That is DER and CER are unambiguous, since there are no encoding options.

A more compact encoding is achieved with the *Packed Encoding Rules* (PER), [40], which were introduced with the revised ASN.1 recommendation in 1994. PER takes a rather different approach from that taken by BER. The first difference is that the T (Type) part is omitted from the encodings, and any tags in the notation are completely ignored. A second difference is that PER takes full account of the sub-typing information while BER completely ignores it. PER uses the sub-typing information, for example, to omit length fields whenever possible. In summary, use of PER results in compact encodings that require much more computation to produce than does BER.

The 'top level' X.509 v3 certificate syntax is as below (note that this syntax is the same for all three versions of the X.509 certificate). For signature calculation, the certificate is encoded using the 'tag, length, value' ASN.1 distinguished encoding rules (DER), [39].

```
Certificate  ::=  SEQUENCE  {
        tbsCertificate       TBSCertificate,
        signatureAlgorithm   AlgorithmIdentifier,
        signatureValue       BIT STRING  }
```

The `signatureValue` field contains a digital signature computed upon the ASN.1 DER-encoded `tbsCertificate`. That is, the ASN.1 DER encoded `tbsCertificate` is used as the input to the signature function. This signature value is then ASN.1-encoded as a `BIT STRING` and included in the Certificate's signature field.

The `signatureAlgorithm` field contains the identifier for the cryptographic algorithm used by the CA to sign this certificate. An algorithm identifier is defined by the following ASN.1 structure.

```
AlgorithmIdentifier  ::=  SEQUENCE  {
      algorithm         OBJECT IDENTIFIER,
      parameters        ANY DEFINED BY
            algorithm OPTIONAL  }
```

The algorithm identifier is used to identify a cryptographic algorithm. The `OBJECT IDENTIFIER` component identifies the algorithm (such as DSA with SHA-1). The contents of the optional parameters field will vary according to the algorithm identified. This field must contain the same algorithm identifier as the signature field in the sequence `tbsCertificate`.

The `tbsCertificate` field contains the names of the subject and issuer, a public key associated with the subject, a validity period, and other associated information. The `tbsCertificate` may also include extensions (the name given to the additional fields introduced into version 3 X.509 certificates). The ASN.1 structure is as follows.

```
TBSCertificate  ::=  SEQUENCE  {
  version         [0]  EXPLICIT Version DEFAULT v1,
  serialNumber         CertificateSerialNumber,
  signature            AlgorithmIdentifier,
  issuer               Name,
  validity             Validity,
  subject              Name,
  subjectPublicKeyInfo SubjectPublicKeyInfo,
  issuerUniqueID  [1]  IMPLICIT UniqueIdentifier
      OPTIONAL,  -- If present, must be v2 or v3
  subjectUniqueID [2]  IMPLICIT UniqueIdentifier
      OPTIONAL,  -- If present, must be v2 or v3
  extensions      [3]  EXPLICIT Extensions OPTIONAL
                       -- If present, must be v3
  }
```

The `Version` field describes the version of the encoded certificate. The default value is v1 (in which case this field can be omitted).

The serial number is an integer assigned by the CA to each certificate. It must be unique for each certificate issued by a given CA (i.e., the issuer name and serial number identify a unique certificate). This is especially useful when constructing Certificate Revocation Lists (CRLs), where the serial number can be used to identify the certificate being revoked. The syntax of CRLs is also defined in X.509.

```
CertificateSerialNumber  ::=  INTEGER
```

The certificate validity period is the time interval during which the CA warrants that it will maintain information about the status of the certificate. The field is represented as a `SEQUENCE` of two dates: the date on which the certificate validity period begins (`notBefore`), and the date on which the certificate validity period ends (`notAfter`).

```
Validity ::= SEQUENCE {
        notBefore      Time,
        notAfter       Time }
```

The `SubjectPublicKeyInfo` field is used to carry the public key and identify the algorithm with which the key is used. The algorithm is identified using the `AlgorithmIdentifier` structure.

```
SubjectPublicKeyInfo  ::=  SEQUENCE  {
        algorithm            AlgorithmIdentifier,
        subjectPublicKey     BIT STRING  }
```

The `issuerUniqueID` and `subjectUniqueID` fields may only appear if the X.509 certificate version is 2 or 3. They are present in the certificate to handle the possibility of reuse of subject and/or issuer names over time.

The Extensions field may only appear if the X.509 certificate version is 3. If present, this field is a `SEQUENCE` of one or more certificate extensions. The extensions allow the encoding of policy information within a certificate.

```
Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension
Extension   ::=  SEQUENCE  {
        extnID      OBJECT IDENTIFIER,
        critical    BOOLEAN DEFAULT FALSE,
        extnValue   OCTET STRING  }
```

There are a large number of standardised extensions. The standard also allows implementers to define their own extensions. Some of the more important standardised extensions are as follows.

- **Key usage**. The key usage extension defines the purpose (e.g., encipherment, signature, certificate signing) of the key contained in the certificate. The usage restriction might be employed when a key that could be used for more than one operation is to be restricted.
- **Certificate policies**. A *certificate policy* is a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements. For example, a particular certificate policy might indicate applicability of a type of certificate to the authentication of electronic data interchange transactions for the trading of goods within a given price range.
- **Subject alternative name**. The subject alternative names extension allows additional identities to be bound to the subject of the certificate. Defined options include an Internet electronic mail address, a DNS name, an IP address, and a uniform resource identifier (URI). Other options exist, including completely local definitions. Multiple name forms, and multiple instances of each name form, may be included. Whenever such identities are to be bound into a certificate, the subject alternative name (or issuer alternative name) extension must be used.

2.3 The PKIX certificate profile

As should be clear from the description immediately above, the X.509 certificate format is very flexible, and allows a large number of options to be chosen by the implementer using the standard. As a result, in practice it is necessary to define a *profile* for X.509, specifying which options should be used.

RFC 2459, [8], is the PKIX X.509 profile. The goal of RFC 2459 is to facilitate the use of X.509 certificates within Internet applications for those communities wishing to make use of X.509 technology. Such applications may include the web, electronic mail, user authentication, and IPsec. In order to remove some of the obstacles to using X.509 certificates, RFC 2459 defines a profile to promote the development of certificate management systems, the development of application tools, and interoperability determined by policy.

A number of values of `AlgorithmIdentifier` are defined in the PKIX profile. Examples of combinations of signature techniques and hash-functions for which IDs are defined include the following:

- RSA with MD2,
- RSA with MD5,
- RSA with SHA-1, [41], and
- DSA, [42] with SHA-1, [41].

2.4 Proprietary certificate formats

Apart from the international standard certificate formats, there are a number of other certificate formats defined for use in specific application domains. It is outside the scope of this paper to list them all, but we mention one that is of some practical importance.

This is the 'EMV certificate', [2], defined in the Europay-MasterCard-Visa (EMV) standards governing communications between a payment smart card (e.g. a credit or

debit card) and a merchant terminal. The main reason that EMV certificates were developed (as opposed to adopting X.509 certificates) was the need to minimise the length of certificates. In the card/terminal environment, both storage space and communications bandwidth are in short supply.

EMV certificate are encoded using a Tag-Length-Value technique, much as for BER; however, the number of fields is much less than for X.509. Moreover, the signature algorithm employed (following ISO/IEC 9796-2, [19]) minimises data storage/bandwidth requirements by enabling as much data as possible to be recovered from the signature.

2.5 Underlying technology

When discussing standards governing the creation and format of certificates, it is also worth briefly mentioning standards for the cryptographic techniques employed as part of the certificate creation and verification processes. Fundamental to the creation of a public key certificate are two types of cryptographic function:

- a digital signature scheme, and
- a cryptographic hash-function (used almost invariably as part of a signature scheme).

Digital signature schemes are specified in a number of different standards, notably in IEEE P1363, [4], ISO/IEC 9796, [19], ISO/IEC 14888, [25], [26], [27], NIST FIPS PUB 186-2, [41], and PKCS #1, [4]. Hash-functions are also specified in a number of standards, including ISO/IEC 10118, [20], [21], [22], [23], and NIST FIPS PUB 180-1, [41].

3 Certificate management

We next consider standards for certificate management, i.e. for protocols governing communications between a client of a CA and the CA itself. Note that in environments where the role of a Registration Authority (RA) is distinguished from that of the CA, part of these protocols may actually be conducted between the RA and the client rather than the CA and the client.

3.1 Certificate management protocol

The PKIX Certificate Management Protocol (CMP) is specified in RFC 2510, [9]. At a high level the set of operations for which management messages are defined within RFC 2510 can be grouped as follows.

1. *CA establishment*: When establishing a new CA, certain steps are required (e.g., production of initial CRLs, export of the CA public key).
2. *End entity initialisation*: This includes importing a CA public key to the end entity and requesting information about the options supported by a PKI management entity.
3. *Certification*: Various operations result in the creation of new certificates:
    - Initial registration/certification. This is the process whereby an end entity first makes itself known to a CA or RA, prior to the CA issuing a certificate or certificates for that end entity. The end result of this process (when it is successful) is that a CA issues a certificate for an end entity's public key, and

returns that certificate to the end entity and/or places that certificate in a public repository.

This process may, and typically will, involve multiple 'steps', possibly including initialisation of the end entity's equipment. For example, the end entity's equipment must be securely initialised with the public key of a CA, to be used in validating certificate paths. Furthermore, an end entity typically needs to be initialised with its own key pair(s).

- Key pair update. Every key pair needs to be updated regularly (i.e., replaced with a new key pair), and a new certificate needs to be issued.
- Certificate update. As certificates expire they may be 'refreshed' if nothing relevant in the environment has changed.
- CA key pair update. As with end entities, CA key pairs need to be updated regularly; however, different mechanisms are required.
- Cross-certification request. One CA requests issue of a cross-certificate from another CA. For the purposes of this standard, the following terms are defined. A 'cross-certificate' is a certificate in which the subject CA and the issuer CA are distinct and `SubjectPublicKeyInfo` contains a verification key (i.e., the certificate has been issued for the subject CA's signing key pair). When it is necessary to distinguish more finely, the following terms may be used: a cross-certificate is called an 'inter-domain cross-certificate' if the subject and issuer CAs belong to different administrative domains; it is called an 'intra-domain cross-certificate' otherwise.

  The above definition of 'cross-certificate' aligns with the defined term 'CA-certificate' in X.509. Note that this term is not to be confused with the X.500 '`cACertificate`' attribute type, which is unrelated. In many environments the term 'cross-certificate', unless further qualified, will be understood to be synonymous with 'inter-domain cross-certificate' as defined above. Issue of cross-certificates may be, but is not necessarily, mutual; that is, two CAs may issue cross-certificates for each other.
- Cross-certificate update: Similar to a normal certificate update but involving a cross-certificate.

4. *Certificate/CRL discovery operations*: Some PKI management operations result in the publication of certificates or CRLs.
   - Certificate publication. Having gone to the trouble of producing a certificate, some means for publishing it is needed. This might involve the use of LDAPv2 (see [12]), or could be by other means, including those specified in the body of RFC 2510, [9].
   - CRL publication. As for certificate publication.
5. *Recovery operations*: Some PKI management operations are used when an end entity has 'lost' its Personal Security Environment (PSE), i.e. its local store for security-related material.
   - Key pair recovery. As an option, user client key materials (e.g., a user's private key used for decryption purposes) may be backed up by a CA, an RA,

or a key backup system associated with a CA or RA. If an entity needs to recover these backed up key materials (e.g., as a result of a forgotten password or a lost file), a protocol exchange may be needed to support such recovery.

6. *Revocation operations*: Some PKI operations result in the creation of new CRL entries and/or new CRLs:
   - Revocation request. An authorized person advises a CA of an abnormal situation requiring certificate revocation.
7. *PSE operations*: Whilst the definition of PSE operations (e.g., moving a PSE, changing a PIN, etc.) is beyond its scope, RFC 2510 does define a `PKIMessage` (`CertRepMessage`) which can form the basis of such operations.

As pointed out in RFC 2510, it is important to note that on-line protocols are not the only way of implementing the above operations. For all operations there are off-line methods of achieving the same result, and the RFC 2510 specification does not mandate use of on-line protocols. For example, when hardware tokens are used, many of the operations may be achieved as part of the physical token delivery.

RFC 2510 defines a set of standard messages supporting the above operations. The protocols for conveying these exchanges in different environments (file based, on-line, E-mail, and web-based) are also specified.

3.2 Certificate request messages

At the core of RFC 2510 is the definition of a Certificate Request Message (CRM), in which the public key subject requests the CA for a new certificate. The format of this message is defined in a separate document, RFC 2511, [10].

A certificate request message is composed of the certificate request, an optional *proof of possession* (POP) field and an optional registration information field.

```
CertReqMessages ::= SEQUENCE SIZE (1..MAX) OF CertReqMsg

CertReqMsg ::= SEQUENCE {
    certReq   CertRequest,
    pop       ProofOfPossession  OPTIONAL,
    -- content depends upon key type
    regInfo   SEQUENCE SIZE(1..MAX) of AttributeTypeAndValue
OPTIONAL }
```

The proof of possession field is used to demonstrate that the entity to be associated with the certificate is actually in possession of the corresponding private key. This field may be calculated across the contents of the `certReq` field and varies in structure and content by public key algorithm type and operational mode.

In order to prevent certain attacks and to allow a CA/RA to properly check the validity of the binding between an end entity and a key pair, the PKI management operations specified here make it possible for an end entity to prove that it has possession of (i.e., is able to use) the private key corresponding to the public key for which a certificate is requested. A given CA/RA is free to choose how to enforce POP (e.g., out-of-band procedural means versus the CRMF in-band message) in its certification exchanges (i.e., this may be a policy issue). However, it is mandated by RFC 2511 that CAs/RAs must enforce POP by some means; the stated reason for this is that there are many non-PKIX operational protocols in use (various electronic mail

protocols are one example) that do not explicitly check the binding between the end entity and the private key.

POP can be accomplished in different ways depending on the type of key for which a certificate is requested. For example, for signature keys, the end entity can sign a value to prove possession of the private key.

The `regInfo` field should only contain supplementary information related to the context of the certification request when such information is required to fulfil a certification request. This information may include subscriber contact information, billing information or other ancillary information useful to fulfilment of the certification request.

Information directly related to certificate content should be included in the `certReq` content. However, inclusion of additional `certReq` content by RAs may invalidate the pop field. Data therefore intended for certificate content may be provided in `regInfo`.

3.3 ISO/IEC certificate management standard

Very similar protocols to those defined in RFCs 1510 and 1511 are also defined in a draft international standard, ISO/IEC 15945, [26].

4 Accessing certificates

It is clear that there will be general requirement for users of certificates (wishing to verify the binding between a public key and a user) to have a standard means of accessing a repository for certificates. We now briefly consider two different standardised approaches to providing this access. Note that a more detailed comparison of the two approaches is provided in [3].

4.1 Directory service

As has already been mentioned, the X.509 recommendation is actually just one part of the X.500 series of ITU-T recommendations covering the Directory Service. These X.500 recommendations specify how directory service users can access this service to obtain information about other entities, including their public key certificate. The recommendations also specify how CRLs (with format as defined in X.509) can be retrieved.

4.2 Certificate retrieval using LDAP version 2

The PKIX certificate access protocol is defined in RFC 2559, [12] (see also RFC 2587, [13]). The protocol described in RFC 2559 is designed to satisfy some of the operational requirements within the Internet X.509 PKI. Specifically, this document addresses requirements to provide access to PKI repositories for the purposes of retrieving PKI information, including certificates and CRLs. RFC 2559 also addresses requirements to add, delete and modify PKI information in a repository. The mechanism is based on the Lightweight Directory Access Protocol (LDAP) v2, defined in RFC 1777, [6], and defines a profile of that protocol for use within PKIX, and also updates encodings for certificates and revocation lists given in RFC 1778, [7].

5 Revocation and determining the status of a certificate

There are two standardised approaches to enabling the user of a certificate to determine its status (i.e. determine whether or not it has been revoked). The first is the use of CRLs, namely lists of serial numbers of revoked certificates, signed by the CA. The second is where a Trusted Third Party (the CA or an agent of the CA) provides on-line information regarding the status of a certificate, namely whether it has been revoked. Both approaches have been the subject of standardisation efforts.

5.1 Certificate revocation lists (CRLs)

The X.509 recommendation includes a standard format for CRLs. Each CRL entry contains the serial number (see Section 2.2) of the X.509 certificate being revoked. It is a general requirement of CRLs that they are updated at regular defined intervals, enabling the CRL user to verify that they are in possession of the 'latest' version.

5.2 On-line certificate status protocol

An alternative approach to certificate revocation is defined in Internet RFC 2560, [14]. This document specifies the Online Certificate Status Protocol (OCSP), which enables the user to determine the current status of a digital certificate without requiring CRLs. Instead of, or in addition to, checking against a periodically updated CRL, it may be necessary to obtain up to date information regarding the revocation status of a certificate. Examples include high-value funds transfers or large share deals.

OCSP enables applications to determine the (revocation) state of an identified certificate. OCSP may also be used to obtain additional status information about a certificate. An OCSP client issues a status request to an OCSP responder and suspends acceptance of the certificate in question until the responder provides a response. This protocol specifies the data that needs to be exchanged between an application checking the status of a certificate and the server providing that status. The same protocol is also defined in the draft standard ISO/IEC 15945, [26].

6 Time stamping services

There are currently two parallel efforts to introduce standards for time-stamping services. While a time-stamping service is not an essential part of a PKI, it can be used to enhance the value of certain security services supported by the use of a PKI.

One important use relating to PKIs concerns the use of a time-stamping service to give long term validity to signatures. A secure time-stamp computed on a signed message can be used as evidence that a signed message was generated prior to the date of expiry or revocation of the key used to sign the message.

6.1 ISO/IEC time-stamping standard

Draft international standards ISO/IEC 18014 parts 1-3, [29], [30], [31], are concerned with time-stamping services. The use of digital data stored on easily modifiable media raises the issue of how to certify when these data were created or last changed. Digital time stamping provides help to achieve a proof of timeliness. Digital time stamping must fulfil the following requirements:

- A time variant parameter must be tied to the data in a non-forgeable way to avoid repudiation of the data's existence prior to a certain point in time.
- Data must be provided in a way that its confidentiality may be guaranteed.

The time-stamping methods in use solve these requirements by time-stamping the hash-value of data, which allows for the control of integrity and confidentiality. The data themselves are not exposed. The data's hash-code will be cryptographically bound to the current time value by the TSA. This binding demonstrates the integrity and authenticity of the time stamp. A time certificate providing these elements will be sent to the requester of the time stamp.

Timestamp tokens may also include information relating to previously generated tokens. Here the data's representation and additional information from data time-stamped prior to that time stamp request are input parameters to the time stamping process.

6.2 Internet draft time-stamping standard

A parallel Internet draft, [17], describes a time stamping service that supports assertions of proof that a datum existed before a particular time. This document describes the format of a request sent to a Time Stamping Authority (TSA) and of the response that is returned. It also establishes several security-relevant requirements for TSA operation, with regards to processing requests to generate responses. A TSA may be operated as a Trusted Third Party (TTP) service, though other operational models may be appropriate, e.g. an organisation might require a TSA for internal time stamping purposes. An example of how to prove that a digital signature was generated during the validity period of a public key certificate is given in an annex.

7 Other trusted third party services

We conclude this discussion of technical PKI standards by mentioning two further standardisation activities both still in progress.

7.1 Guidelines for the use and management of TTP services

The use and management of a wide range of Trusted Third Party (TTP) services, many relating to PKIs, are described in an ISO/IEC Technical Report nearing completion, namely ISO/IEC TR 14516,

Associated with the provision and operation of a Trusted Third Party (TTP) are a number of security related issues for which general guidance is necessary to assist business entities, developers and providers of systems and services, etc. This includes guidance on issues regarding the roles, positions and relationships of TTPs and the entities using TTP services, the generic security requirements, who should provide what type of security, what the possible security solutions are, and the operational use and management of TTP service security.

TR 14516 provides guidance for the use and management of TTPs, a clear definition of the basic duties and services provided, their description and their purpose, and the roles and liabilities of TTPs and entities using their services. It is intended primarily for system managers, developers, TTP operators and enterprise users to select those TTP services needed for particular requirements, their subsequent management, use and operational deployment, and the establishment of a Security Policy within a TTP.

It is not intended to be used as a basis for a formal assessment of a TTP or a comparison of TTPs. This document identifies different major categories of TTP services including: time stamping, non-repudiation, key management, certificate management, and electronic notary public services.

7.2 SCVP

Certificate validation is a difficult problem. If certificate handling is to be widely deployed in a variety of applications and environments, the amount of processing an application needs to perform before it can accept a certificate must be reduced. There are a variety of applications that can use public key certificates but are burdened by the overhead of validating certificates, when all the application really wants is the public key and name from the certificate, and to find out whether the certificate may be used for a particular purpose. There are other applications that can perform certificate path validation but have no reliable method of obtaining a current chain to a trusted certificate.

The primary goals of the Simple Certificate Validation Protocol (SCVP), [16], are to make it easier for applications to deploy systems using a PKI, and to allow centralisation of PKI policy administration. Parts of SCVP can be used by clients that do much of the PKI processing themselves and simply want a useful but untrusted server that will collect information for them. Other parts can be used by clients that have complete trust in the server to both offload the work of certificate validation and to ensure that policies are enforced in a consistent fashion across an enterprise.

Untrusted SCVP servers can give clients the certificate chains needed for path validation. They can also give clients revocation information such as CRLs and OCSP responses that the client can use in the client's path validation. These services can be valuable to client systems that do not include the protocols needed to find and download all of the intermediate certificates, CRLs, and OCSP responses needed for the client to perform complete path validation.

Trusted SCVP servers can perform full certificate validation for the client. If a client uses these services, it inherently trusts the SCVP server as much as it would its own path validation software (if it contained such software). There are two main reasons that a client may want to trust such an SCVP server:

- The client does not want to incur the overhead of including path validation software and running it for each certificate it receives.
- The client is in an enterprise that wants to centralise its PKI validation policies, such as which root certificates are trusted and which types of policy checking are performed during path validation.

8 Policy and certification practice statements

RFC 2527, [11], presents a framework to assist the writers of certificate policies or certification practice statements for CAs and PKIs. In particular, the framework provides a comprehensive list of topics that potentially need to be covered in a certificate policy definition or a certification practice statement.

The degree to which a certificate user can trust the binding embodied in a certificate depends on several factors. These factors include the practices followed by the CA in authenticating the subject; the CA's operating policy, procedures, and security

controls; the subject's obligations (for example, in protecting the private key); and the stated undertakings and legal obligations of the CA (for example, warranties and limitations on liability).

A Version 3 X.509 certificate may contain a field declaring that one or more specific certificate policies applies to that certificate, [18], [34]. According to X.509, a certificate policy is 'a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements'. A certificate policy may be used by a certificate user to help in deciding whether a certificate, and the binding therein, is sufficiently trustworthy for a particular application. The certificate policy concept is an outgrowth of the policy statement concept developed for Internet PEM, [5].

A more detailed description of the practices followed by a CA in issuing and otherwise managing certificates may be contained in a certification practice statement (CPS) published by or referenced by the CA. According to the American Bar Association (ABA) Digital Signature Guidelines, 'a CPS is a statement of the practices which a certification authority employs in issuing certificates', [1].

The purpose of RFC 2527 is to establish a clear relationship between certificate policies and CPSs, and to present a framework to assist the writers of certificate policies or CPSs with their tasks. In particular, the framework identifies the elements that may need to be considered in formulating a certificate policy or a CPS. The purpose is not to define particular certificate policies or CPSs.

The scope of RFC 2527 is limited to discussion of the contents of a certificate policy (as defined in X.509) or CPS (as defined in the ABA Guidelines). In particular, this document describes the types of information that should be considered for inclusion in a certificate policy definition or a CPS. While the framework as presented generally assumes use of the X.509 version 3 certificate format, it is not intended that the material be restricted to use of that certificate format. Rather, it is intended that this framework be adaptable to other certificate formats that may come into use.

The scope does not extend to defining security policies generally (such as organisation security policy, system security policy, or data labelling policy) beyond the policy elements that are considered of particular relevance to certificate policies or CPSs. RFC 2527 does not define a specific certificate policy or CPS.

9 Further information

Further information about many of the standards discussed in this paper are available online.

- For copies of the EMV standards, and related documentation, go to the EMVCo site at `http://www.emvco.com/`.
- General information on IEEE standards is available at `http://standards.ieee.org/`. Information on IEEE P1363 is available at the P1363 home page (`http://grouper.ieee.org/groups/1363/`).
- Internet (IETF) RFCs are available at many sites on the web; see for example the IETF home page at `http://www.ietf.org/`.
- For information regarding published ISO standards, see the ISO web site at `http://www.iso.ch/`.

- For information regarding published ITU-T recommendations, see http://www.itu.int/.
- The NIST FIPS publications are all available on-line at http://www.itl.nist.gov/fipspubs/by-num.htm.
- The PKCS standards are available at the RSA Laboratories site (http://www.rsasecurity.com/rsalabs/pkcs/).

References

[1] ABA, *Digital signature guidelines: Legal infrastructure for Certification Authorities and electronic commerce*. American Bar Association, 1995.

[2] EMV '96, *Integrated Circuit Card Specification for Payment Systems*. Version 3.1.1, May 31, 1998.

[3] V. Hassler, 'X.500 and LDAP security: A comparative overview'. *IEEE Network* Vol. 13 no. 6 (November/December 1999) pp. 54-64.

[4] IEEE P1363, *Standard Specifications For Public Key Cryptography*, 2000.

[5] Internet RFC 1422, *Privacy enhancement for Internet electronic mail, Part II: certificate-based key management* (by S. Kent), February 1993.

[6] Internet RFC 1777, *Lightweight Directory Access Protocol* (by Y. Yeong, T. Howes and S. Kille), March 1995.

[7] Internet RFC 1778, *The string representation of standard attribute syntaxes* (by T. Howes, S. Kille, W. Yeong, and C. Robins), March 1995.

[8] Internet RFC 2459, *Internet X.509 public key infrastructure – Certificate and CRL profile* (by R. Housley, W. Ford, W. Polk and D. Solo), January 1999.

[9] Internet RFC 2510, *Internet X.509 public key infrastructure – Certificate management protocols* (by C. Adams and S. Farrell), March 1999.

[10] Internet RFC 2511, *Internet X.509 certificate request message format* (by M. Myers, C. Adams, D. Solo and D. Kemp), March 1999.

[11] Internet RFC 2527, *Internet X.509 public key infrastructure – Certificate policy and certification practices framework* (by S. Chokhani and W. Ford), March 1999.

[12] Internet FFC 2559, *Internet X.509 public key infrastructure – Operational protocols – LDAPv2* (by S. Boeyen, T. Howes and P. Richard), April 1999.

[13] Internet RFC 2587, *Internet X.509 public key infrastructure – LDAPv2 schema* (by S. Boeyen, T. Howes and P. Richard), June 1999.

[14] Internet RFC 2560, *X.509 internet public key infrastructure – Online certificate status protocol (OCSP)* (by M. Myers, R. Ankney, A. Malpani, S. Galperin and C. Adams), June 1999.

[15] Internet draft, *Internet X.509 public key infrastructure – PKIX roadmap* (by A. Arsenault and S. Turner), March 2000.

[16] Internet draft, *Simple Certificate Validation Protocol (SCVP)* (by A. Malpani and P. Hoffman), June 2000.

[17] Internet draft, *Internet X.509 public key infrastructure – Time Stamp Protocol (TSP)*, (by C. Adams, P. Cain, D. Pinkas, and R. Zuccherato), October 2000.

[18] ISO/IEC 9594-8: 1998 (3rd edition), *Information technology – Open Systems Interconnection – The Directory: Authentication framework.*

[19] ISO/IEC 9796-2: 1997, *Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Mechanisms using a hash-function.*

[20] ISO/IEC 10118-1: 1994, *Information technology – Security techniques – Hash-functions – Part 1: General.*

[21] ISO/IEC 10118-2: 1994, *Information technology – Security techniques – Hash-functions – Part 2: Hash-functions using an n-bit block cipher algorithm.*

[22] ISO/IEC 10118-3: 1998, *Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions.*

[23] ISO/IEC 10118-4: 1998, *Information technology – Security techniques – Hash-functions – Part 4: Hash-functions using modular arithmetic.*

[24] ISO/IEC 2nd DTR 14516, *Information technology – Security techniques – Guidelines on the use and management of Trusted Third Party services,* April 2000.

[25] ISO/IEC 14888-1: 1998, *Information technology – Security techniques – Digital signatures with appendix – Part 1: General.*

[26] ISO/IEC 14888-2: 1999, *Information technology – Security techniques – Digital signatures with appendix – Part 2: Identity-based mechanisms.*

[27] ISO/IEC 14888-3: 1998, *Information technology – Security techniques – Digital signatures with appendix – Part 3: Certificate-based mechanisms.*

[28] ISO/IEC FDIS 15945, *Information technology – Security techniques – Specification of TTP services to support the application of digital signatures,* May 2000.

[29] ISO/IEC WD 18014-1, *Information technology – Security techniques – Time stamping services – Part 1: Framework*, May 2000.

[30] ISO/IEC WD 18014-2, *Information technology – Security techniques – Time stamping services – Part 2: Mechanisms producing independent tokens,* May 2000.

[31] ISO/IEC WD 18014-3, *Information technology – Security techniques – Time stamping services – Part 3: Mechanisms producing linked tokens,* May 2000.

[32] ITU-T Recommendation X.509 (1988), *Information technology – Open Systems Interconnection – The Directory: Authentication framework.*

[33] ITU-T Recommendation X.509 (11/93), *Information technology – Open Systems Interconnection – The Directory: Authentication framework.*

[34] ITU-T Recommendation X.509 (08/97), *Information technology – Open Systems Interconnection – The Directory: Authentication framework.*

[35] ITU-T Recommendation X.680 (12/97), *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

[36] ITU-T Recommendation X.681 (12/97), *Information technology - Abstract Syntax Notation One (ASN.1): Information object specification.*

[37] ITU-T Recommendation X.682 (12/97), *Information technology - Abstract Syntax Notation One (ASN.1): Constraint specification.*

[38] ITU-T Recommendation X.683 (12/97), *Information technology - Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.*

[39] ITU-T Recommendation X.690 (12/97), *Information technology - ASN.1 encoding rules - Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).*

[40] ITU-T Recommendation X.691 (12/97), *Information technology - ASN.1 encoding rules - Specification of Packed Encoding Rules (PER).*

[41] NIST FIPS PUB 180-1, *Secure hash standard,* April 1995.

[42] NIST PIPS PUB 186-2, *Digital signature standard,* January 2000.

[43] PKCS #1, *RSA cryptography standard,* Version 2.1 (draft), September 1999.