CHAPTER 6

Certificates and Certification

As we discussed in Chapter 2, *public-key cryptography* involves the use of public/private-key pairs to facilitate digital signature and key management services. The fundamental principle that enables public-key technology to scale is the fact that the public component of the public/private-key pair may be distributed freely among the entities that need the public component to use the underlying security services. (See Chapters 4 and 5 for more information regarding security services enabled through the use of a PKI.)

However, distribution of the public component without some form of integrity protection would defeat the very foundation for these security services. Thus, the public-key component must be protected in such a way that it will not impact the overall scalability that public-key cryptography techniques offer. Further, we need to bind certain attributes with the public key.

Thus, a data integrity mechanism is required to ensure that the public key (and any other information associated with that public key) is not modified without detection. However, a data integrity mechanism alone is not sufficient to guarantee that the public key belongs to the claimed owner. A mechanism that binds the public key to the claimed owner in a trustworthy manner is also required. In the end, the goal is to provide a single mechanism by which a relying party (that is, the "user" of the key and associated data, as defined in [RFC2527]) is assured that

- The integrity of the public key (and any other associated information) is sound.
- The public key (and any other associated information) has been bound to the claimed owner in a trusted manner.

Our purpose in this chapter is to explain how the use of public-key certificates accomplishes these goals. In particular, we will focus on the syntax and semantics of the X.509 Version 3

public-key certificate, and we will explain how the integrity of these certificates is established and maintained.

Certificates

Kohnfelder first introduced the concept of using a signed data structure or certificate to convey the public key to a relying party in his 1978 bachelor's thesis entitled "Towards a Practical Public-Key Cryptosystem" [Kohn78]. Thus, over two decades ago, it was recognized that a scalable and secure method (from an integrity perspective) would be required to convey the public keys to the parties that needed them. Simply stated, public-key certificates are used to bind an entity's name (and possibly additional attributes associated with that entity) with the corresponding public key.

When discussing the concept of a "certificate," it is important to recognize that a number of different types of certificates exist, including

- X.509 public-key certificates
- Simple Public Key Infrastructure (SPKI) certificates
- Pretty Good Privacy (PGP) certificates
- Attribute certificates

The certificate types listed here have separate and distinct formats. In some cases, one type of certificate may be defined in several different versions, and a single version may be instantiated in a number of different ways. For example, there are three versions of an X.509 public-key certificate. Version 1 is a subset of Version 2, and Version 2 is a subset of Version 3. Because a Version 3 public-key certificate includes numerous optional extensions (as discussed later), it can be instantiated in a number of application-specific ways; for example, Secure Electronic Transaction (SET) certificates are X.509 Version 3 public-key certificates with specific extensions defined solely for SET exchanges.

To complicate matters further, multiple terms commonly denote the same thing. For example, in many environments the terms *certificate* and *digital certificate* are synonymous with an X.509 public-key certificate.

For the purposes of this book, a *certificate* is synonymous only with a *Version 3 public-key certificate* as defined in the X.509 Recommendation [X509–00]. (See Box 6.1.) Any other type of certificate will be further qualified to avoid any confusion with this usage. In this chapter, we discuss the structure and content of a certificate. As we describe the structure of a certificate in the next section, you will notice that certificates can be issued to Certification Authorities

Box 6.1 Versions 1–3 of X.509

Three versions of an X.509 public-key certificate are defined. The original Version 1 public-key certificate, defined in the 1988 X.509 Recommendation, suffers from inherent inflexibility because this version cannot be extended to support additional attributes. The Version 2 public-key certificate did little to correct this shortcoming because it simply augments Version 1 with the addition of two optional fields. Because the demand for these fields was (and continues to be) negligible and the same inability to support extensions also applies, the Version 2 public-key certificate has failed to gain widespread acceptance.

Not surprisingly, Version 3 public-key certificates, as specified in the 1997 X.509 Recommendation [X509–97], were introduced to correct the deficiencies associated with the Version 1 and Version 2 definitions. Specifically, Version 3 offers significant improvements over Version 1 and Version 2 through the addition of optional extensions.

More recently (circa June 2000), the 2000 X.509 Recommendation [X509–00] was completed. The 2000 version includes numerous changes from the previous version, including the definition of two additional extensions to the Version 3 public-key certificate.

In the enterprise domain, it is fair to say that Version 3 public-key certificates are the preferred choice as they are the most flexible, and many of the extensions are required to fully support the requirements of the enterprise.

(CAs) and to end entities (for example, end users or devices). These are referred to as CA certificates and end-entity certificates, respectively.

Digital Certificate

The term *digital certificate* is sometimes used to denote a certificate in electronic form. However, this term can be somewhat confusing in some circumstances because a number of quite different certificates (for example, an X.509 public-key certificate, an attribute certificate, a PGP certificate, and so on) are "digital." For that matter, a digitized birth certificate might be considered to be a "digital certificate." Thus, unless the term is either explicitly defined or further qualified, it is not precise enough to convey any intended meaning.

The use of this term has also introduced confusion when describing the relationship between *digital certificates* and *digital signatures*. In particular, a common mistake is to assume that Alice can authenticate herself by simply supplying a "digital certificate" without the corresponding "digital signature." Further, although a digital signature is meaningful in itself (that is, a digital signature is distinguished from a handwritten or a generic electronic signature), digital

certificate does not offer the same connotation—especially when referring solely to a public-key certificate.

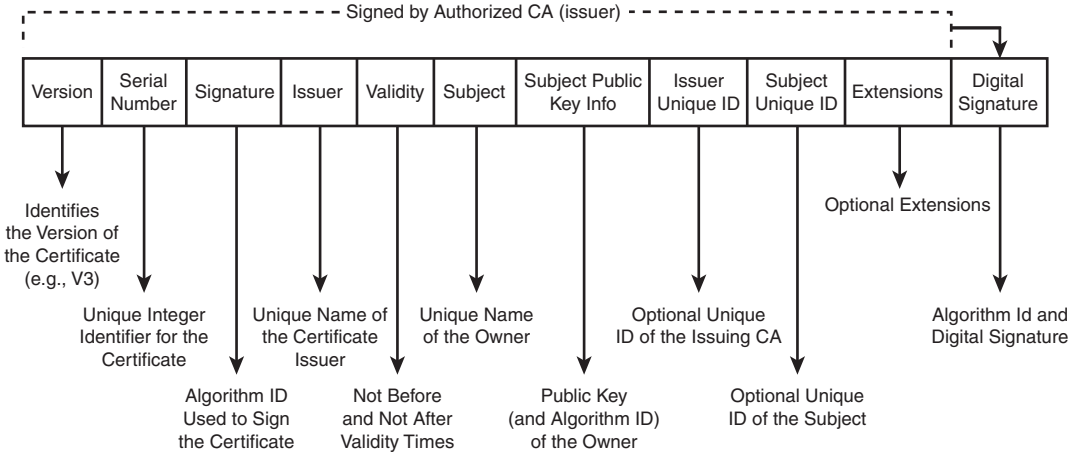
To be perfectly precise, any reference to a *certificate* should be fully qualified to avoid unnecessary confusion. However, in accordance with common practice in the PKI industry, we will simply use the term *certificate* as a shorthand notation for an X.509 Version 3 public-key certificate. We will explicitly identify all other types of certificates where appropriate—even the generic use of *certificate* will be qualified wherever any doubt may arise.

Certificate Structure and Semantics

Although X.509 defines certain requirements associated with the standard fields and extensions of a certificate, other issues still must be further refined in specific profiles to fully address interoperability considerations. The Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) Working Group introduced such a profile in January 1999 with the publication of RFC2459. In April 2002, RFC2459 was replaced with RFC3280. Although RFC3280 is targeted for the Internet community, a number of its recommendations could equally apply in the enterprise environment, and consistency should be maintained wherever possible. Therefore, we will provide references to some of the recommendations made in RFC3280 where appropriate.

Figure 6.1 shows the generic structure of a Version 3 certificate. The following list defines the fields represented in Figure 6.1:

Figure 6.1 Version 3 certificate structure.



- *Version* indicates the version of the certificate (either 1, 2, or 3).
- *Serial Number* is the unique identifier for this certificate relative to the certificate issuer.
- *Signature* indicates the algorithm identifier (that is, the Object Identifier, or OID, plus any associated parameters) of the algorithm used to calculate the digital signature on the certificate.[1] For example, the OID for SHA-1 with RSA might be present, indicating that the digital signature is an SHA-1 hash encrypted using RSA.
- *Issuer* is the *Distinguished Name (DN)* of the CA that issued the certificate and must always be present.[2]
- *Validity* indicates the window of time that this certificate should be considered valid unless otherwise revoked (refer to Chapter 8 for more information on revocation). This field is composed of *Not Valid Before* and *Not Valid After* dates/times that may be represented in UTC Time or in Generalized Time (however, RFC3280 has specific rules associated with the use of these time representations).
- *Subject* indicates the DN of the certificate owner and must be non-null unless an alternate name form is used (refer to the Extensions field later).
- *Subject Public Key Info* is the public key (and algorithm identifier) associated with the subject and must always be present.
- *Issuer Unique ID* is an optional unique identifier of the certificate issuer present in Version 2 and Version 3 only; this field is rarely used in implementation practice, and it is not recommended for use by RFC3280.

[1]An OID is simply a unique representation for a given object. When expressed verbally or in writing among human beings, an OID is represented as a sequence of integers that are separated by decimal points or dots (much the same as an Internet Protocol (IP) address is expressed in the familiar dotted decimal notation). OIDs are hierarchical in nature, and they are registered with international, national, or organizational registration authorities in order to ensure that the allocated OID for a given object is unique. As an example, the OID for SHA-1 with RSA (which might be present in the Signature field of a certificate) is 1.2.840.113549.1.1.5.

[2]A DN is a hierarchical naming convention defined in the X.500 Recommendations. DNs are designed to help ensure entity names are unique. Without getting into too much detail regarding Directories and Directory Information Trees, DNs are expressed as a concatenation of Relative Distinguished Names (RDNs) from the top- or root-level node down to the last node of the DN. For example, "C = CA, O = ADGA, OU = AEPOS Technologies, CN = Steve Lloyd" is an example of a DN. Note that each RDN (C = CA is an RDN, O = ADGA is an RDN, and so on) must be unique at each level, otherwise the uniqueness of the DN as a whole would not be guaranteed.

- *Subject Unique ID* is an optional unique identifier of the certificate owner present in Version 2 and Version 3 only; this field is rarely used in implementation practice, and it is not recommended for use by RFC3280.

Each certificate extension described next is associated with a criticality flag. In general, extensions can be marked critical or noncritical, although the standards often recommend or even mandate the criticality associated with certain extensions.

An extension that has been marked critical must be processed and understood, or the certificate is not to be used. A noncritical extension is to be processed if possible, but it may be gracefully ignored if it is not recognized.

Extensions are optional standard and private extensions (present in Version 3 only) and include the following:

- *Authority Key Identifier* is a unique identifier of the key that should be used to verify the digital signature calculated over the certificate; it distinguishes between multiple keys that apply to the same certificate issuer. RFC3280 mandates the inclusion of this field for all but self-signed certificates. (Chapter 9 discusses self-signed certificates.)
- *Subject Key Identifier* is a unique identifier associated with the public key contained in this certificate; it distinguishes between multiple keys that apply to the same certificate owner. RFC3280 mandates this field for CA certificates, and it is recommended for end-entity certificates.
- *Key Usage* is a bit string used to identify (or restrict) the functions or services that can be supported by using the public key in this certificate; it can be used to indicate support for digital signature, non-repudiation, key encipherment, data encipherment, key agreement, certificate signature, Certification Revocation List (CRL) signature, encipher only, and decipher only. A profile typically specifies allowable combinations (for example, the U.S. Federal PKI (FPKI) profile [FPKIpro] explicitly identifies permitted key usage combinations).
- *Extended Key Usage* is a sequence of one or more OIDs that identify specific usage of the public key in the certificate. Although X.509 does not explicitly define identifiers for this purpose, RFC3280 identifies several OIDs associated with this extension, including Transport Layer Security (TLS) server authentication, TLS client authentication, code signing, e-mail protection, time stamping, and Online Certificate Status Protocol (OCSP) signing. (OCSP is discussed in more detail in Chapter 8.) The list of OIDs will likely be augmented over time as needs dictate, so this should not be considered an exhaustive list. RFC3280 also points out that this extension is typically used with end-entity certificates.
- *CRL Distribution Point* indicates the location of the CRL partition where revocation information associated with this certificate resides. (Refer to Chapter 8 for more informa-

tion regarding revocation techniques and CRL Distribution Points.) RFC3280 provides additional guidance with respect to which attributes within the CRL Distribution Point extension should be populated and what it means when multiple distribution points are present.

- *Private Key Usage Period* indicates the time window that the private key associated with the public key in this certificate can be used; it is intended for use with digital signature keys/certificates. Like the certificate validity period, this window is specified in terms of *Not Valid Before* and *Not Valid After* dates/times (although only Generalized Time is permitted here). Judicious use of this extension can establish a buffer between the time the signing private key expires and the time the corresponding public key used to verify the digital signatures created with that private key expires. This should help eliminate many instances where perfectly valid digital signatures needlessly come into question because the key lifetimes of both the private and public key were too close together (or even identical).

  Note that when this extension is absent, the validity periods of the public key and the private key are identical. As Chapter 7 discusses, a new key pair should be issued before the private key expires in order to avoid any unnecessary downtime. It is interesting to note that RFC3280 recommends against the use of this extension. One reason for this is that the interpretation of this field is not universally agreed upon, which could give rise to inconsistent implementations.

- *Certificate Policies* indicates a sequence of one or more policy OIDs and optional qualifiers associated with the issuance and subsequent use of the certificate. If this extension is marked critical, the processing application must adhere to at least one of the policies indicated, or the certificate is not to be used. Although RFC3280 recommends that policy qualifiers should not be used (in order to promote interoperability), it does define two possible qualifiers: the Certification Practice Statement (CPS) qualifier and the User Notice qualifier. The CPS qualifier is a Uniform Resource Identifier (URI) at which one can find the CPS that applies to this certificate. A notice reference, an explicit notice (up to 200 characters), or both, can comprise the User Notice qualifier.

- *Policy Mappings* indicates one or more policy OID equivalencies between two CA domains and are present only in CA certificates. The Certificate Policies section in this chapter provides additional information related to policy mappings.

- *Subject Alternative Name* indicates alternative name forms associated with the owner of the certificate (for example, e-mail address, IP address, URI, and so on). Alternative name forms are to be considered just as binding as the subject DN, if present. RFC3280 further specifies that if the subject DN is null, one or more alternative name forms must be present, and this extension must be marked critical.

- *Issuer Alternative Name* indicates alternative name forms associated with the issuer of the certificate (for example, e-mail address, IP address, URI, and so on). RFC3280 specifies the same processing rules as specified under the Subject Alternate Name extension with the exception that the issuer's DN must always be present in the Issuer field. One reason for this requirement is to maintain compatibility with the S/MIME specification.
- *Subject Directory Attributes* indicates a sequence of attributes associated with the owner of the certificate. Although this extension is not currently in widespread use, several known applications exist where this extension conveys access control information. However, we recommend exercising caution when using a certificate to convey privilege-related information, because any change in those privileges would force the revocation of the existing certificate and a new certificate would have to be issued. (See Box 6.2.)
- *Basic Constraints* indicates whether this is a CA certificate. Typically, this field is absent in end-entity certificates. If it is present in an end-entity certificate, the value of the CA attribute in the Basic Constraints field must be *false*. For CA certificates, the Basic Constraints field should always be present, and the CA attribute in the Basic Constraints field must be set to a value of *true*. Note that X.509 [X509–00] recommends (but doesn't mandate) that this extension be marked critical. RFC3280 mandates that this extension be present and marked critical if the associated public key is to be used to verify digital signatures on certificates. However, RFC3280 makes a distinction between public keys used to verify digital signatures on certificates and those that might be used to verify digital signatures on CRLs only or those that are used in conjunction with certificate management protocols. In these cases, the extension may be marked critical or noncritical.

  For CA certificates, the Basic Constraints extension can also include a Path Length Constraint. In accordance with X.509 [X509–00], the Path Length Constraint indicates "the maximum number of CA certificates that may follow this certificate in a certification path." (Chapter 9 describes certification paths and certification path processing.) A value of zero indicates that the CA can issue only end-entity certificates. The absence of the

Box 6.2 Certificate Perishability

Any change to the information contained within a given certificate before it naturally expires necessarily means that the existing certificate must be revoked and a new certificate must be issued. Therefore, exercise care to ensure that the attributes placed within the certificate are fairly static in order to avoid wasteful certificate revocation and (re)issuance. Attributes associated with an end entity that will tend to be fairly dynamic in nature should be conveyed through some other means (for example, via attribute certificates, which are briefly discussed later in this chapter and in Chapter 5).

Path Length Constraint in a CA certificate indicates that no restriction is placed on the length of the certification path (that is, the length of the certification path is unbounded). Note that the Path Length Constraint should not be present in an end-entity certificate and is meaningless therein; it should be ignored if it is present.

- *Name Constraints,* an extension present only in CA certificates, indicates required and/or excluded subtree names through the use of the *Permitted Subtrees* and/or *Excluded Subtrees* attributes, respectively. The specified names can take the form of a DN, a URI, an e-mail address, or any other name form that lends itself to a hierarchical structure. The idea is to qualify the name space that applies to the subject names that follow the CA certificate in which the restrictions took effect. If present, this extension should be marked critical [X509–00]. RFC3280 mandates that this extension be marked critical and points out that Name Constraints should not be applied to certificates where the Issuer and Subject fields are the same unless the certificate is the last one in the certification path. (Chapter 9 provides additional information regarding Name Constraints in relation to certification path validation.)

- *Policy Constraints,* an extension present only in CA certificates, indicates required policy identifiers and/or prohibited policy mappings through the use of the *Require Explicit Policy* and/or *Inhibit Policy Mapping* attributes, respectively. It is important to note that if the Require Explicit Policy comes into effect anywhere in a given certification path, it applies to the entire certification path. This is not the case with the Inhibit Policy Mapping attribute that applies only to subsequent certificates in the certification path. A value of zero for either attribute indicates that the restrictions apply immediately. A nonzero value for either attribute indicates where the restrictions might apply; that is, the value is an offset in the certification path, and the nominated CA certificate may or may not be encountered. If present, this extension should be marked critical [X509–00]. RFC3280 states that this extension may be critical or noncritical.

- *Inhibit Any Policy,* an extension present only in CA certificates, indicates that the any-policy identifier (OID value 2.5.29.32.0) should not be considered a legitimate match for other policy identifiers. A value of zero indicates that this restriction applies from this point forward. A nonzero value indicates where the restriction might apply; that is, the value is an offset in the certification path, and the nominated CA certificate may or may not be encountered. This extension was first standardized in the 2000 version of X.509 [X509–00]. In accordance with X.509, this extension may be marked critical or noncritical, but it is recommended that it be marked critical. RFC3280 states that this extension must be marked critical.

- *Freshest CRL Pointer,* an extension present in end-entity and CA certificates, provides a pointer to the "freshest" CRL information. In practice, this is likely to be a pointer to a Delta CRL. (Refer to Chapter 8 for more information on Delta CRLs.) The syntax is the

same as is used with CRL Distribution Points. This extension was first standardized in the 2000 version of X.509 [X509–00]. The extension may be marked critical or noncritical, but if it is marked critical, the relying party must retrieve and use this information. RFC3280 states that this extension should be marked noncritical.

Private extensions can also be defined in accordance with X.509. Private extensions are typically defined for domain-specific use. For example, RFC3280 defines two private extensions for Internet use as follows:

- *Authority Information Access,* a private extension present in end-entity and CA certificates, indicates how information or services offered by the issuer of the certificate can be obtained. The type of information and services include on-line validation services and policy information but do not include CRL location information. (The CRL Distribution Point discussed earlier is used for that purpose.) The extension syntax is composed of a sequence of an access method OID that describes the type and format of the service/information and the associated location of the service/information in the form of a General Name (for example, a URI, Directory Name, or RFC822 Name). Two access method OIDs have been defined. One access method, referred to as "CA Issuers," retrieves information about CAs that are superior to the issuer of the certificate. This information can be used to help build certification paths. The other defined access method is for on-line validation services based on OCSP. (Refer to Chapter 8 for more information on OCSP.) Other access methods may be defined in the future. RFC3280 states that this extension must be marked noncritical.
- *Subject Information Access,* a private extension present in end-entity and CA certificates, indicates how information and services offered by the subject in the certificate can be obtained. Like the Authority Information Access private extension, the extension syntax is composed of a sequence of an access method OID and the associated location of the service/information in the form of a General Name. One access method OID has been defined for CAs (CA Repository), and one access method OID has been defined for end entities (time stamping). The CA Repository access method identifies the location of the repository where the CA publishes certificate and CRL information. The Time-Stamping access method indicates that the subject identified in the certificate offers a time stamping service. Other access methods may be defined in the future. RFC3280 states that this extension must be marked noncritical.

Alternative Certificate Formats

As discussed previously in this chapter, certificate types other than the X.509 Version 3 public-key certificate are available. We discuss these further in the next few pages.

SPKI

In contrast to the IETF PKIX Working Group that focused on X.509 issues for the Internet (see Chapter 18), a separate IETF working group was formed to address a (potentially) simpler public-key infrastructure, referred to as the Simple Public Key Infrastructure (SPKI), for the Internet. Specifically, the charter of the IETF SPKI Working Group[3] was to

> develop Internet standards for an IETF sponsored public-key certificate format, associated signature and other formats, and key acquisition protocols. The key certificate format and associated protocols are to be simple to understand, implement, and use.

The IETF SPKI Working Group produced a number of technical and informational documents, including

- SPKI certificate format
- SPKI certificate theory
- SPKI requirements
- SPKI examples

You can retrieve the relevant SPKI documents from `http://www.ietf.org/html._charters/spki-charter.html`. Because the focus of the SPKI work was on authorization rather than on identity, the SPKI certificate is referred to as an *authorization certificate*. The primary purpose of the SPKI authorization certificate is to convey permissions. It also includes the ability to delegate permissions to others.

Although the SPKI authorization certificate has some things in common with an X.509 public-key certificate (for example, Issuer and Validity), the syntax and, in many cases, the semantics of these fields are not the same. Further, a number of fields are defined for one type of certificate that does not correspond to an equivalent mapping in the other. In addition, the naming conventions (and assumptions) are completely different because the SPKI work adopted the naming conventions as defined in SDSI.

The IETF work on SPKI has concluded. However, the extent to which this work will be used in practice still remains to be seen. There is currently very little demand for SPKI-based certificates, and in the absence of market demand, CA and PKI vendors are not likely to implement a completely different certificate syntax in addition to X.509 Version 3 public-key certificates.

---

[3] See `http://www.ietf.org/html.characters/spki-charter.html`.

Chapter 18 provides additional information regarding the role and status of the SPKI work.

PGP

Essentially, *Pretty Good Privacy (PGP)* is a method for encrypting and digitally signing e-mail messages and files. Phil Zimmermann introduced the first version of PGP in the early 1990s [Zimm95]. Version 2.x of PGP was published a number of years later as an IETF standards-track specification entitled PGP Message Exchange Formats [RFC1991]. The latest version of PGP, referred to as *OpenPGP*, has been published as an IETF standards-track specification entitled OpenPGP Message Format [RFC2440]. A document on the Internet standards track also incorporates MIME with PGP and is entitled MIME Security with Pretty Good Privacy [RFC2015].

PGP specifies packet formats that convey messages and files from one entity to another. PGP also includes packet formats that convey *PGP keys* (sometimes referred to as *PGP certificates*) from one entity to another.

Significant differences exist between PGP keys (or certificates) and X.509 Version 3 public-key certificates, and the trust models they embody are also completely different. (Chapter 9 discusses the PGP trust model further.) The significant differences between PGP keys and the X.509 Version 3 public-key certificate have created interoperability barriers between the PGP-user community and other communities that base their certificate formats on X.509 (for example, the S/MIME-user community). This is much more than a protocol incompatibility issue because the very foundation for the underlying public-key-enabled security services is different and incompatible. One possible solution is for PGP (or OpenPGP) to adopt X.509 Version 3 public-key certificates in addition to (or perhaps in lieu of) the PGP certificate. In fact, Version 6.5 of OpenPGP has pursued this direction and is now capable of supporting X.509 certificates. Although allowing OpenPGP users to tap into X.509-based PKIs, this still does not solve the basic protocol incompatibilities between OpenPGP and S/MIME. Another possibility might be for PKI vendors to offer products that support both PGP and X.509 Version 3 public-key certificates, but this can lead to other difficulties due to the significant differences in the trust models. (For example, it would introduce significant administrative and control issues.)

Although PGP enjoys a significant amount of use over the Internet, it does not make a good candidate for the corporate intranet (that is, the enterprise domain) because all trust decisions rest with individuals rather than with the enterprise. Because many CA and PKI vendors seem to have concentrated their product development efforts on the enterprise domain, it is unclear that they have any motivation to offer PGP-compatible (or OpenPGP-compatible) products.

Chapter 18 provides additional information regarding the role and status of the PGP/OpenPGP work.

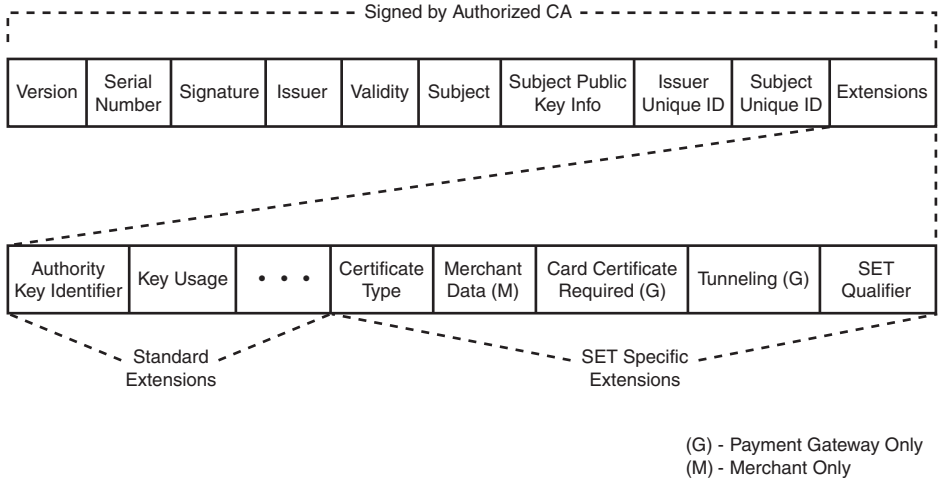
SET

The *Secure Electronic Transaction (SET)* specifications [SET1; SET2; SET3] define a standard to support credit card payment transactions over distributed communications networks such as the Internet. Essentially, SET defines a standard payment protocol and specifies requirements that the supporting PKI is expected to meet.

SET adopts the X.509 Version 3 public-key certificate format, and it defines specific private extensions that have meaning only in a SET context. SET also levies certain profile requirements on the standard extensions. Figure 6.2 illustrates a SET certificate. Note that Figure 6.2 does not represent all the possible extensions. (For example, it does not represent the Hashed Root Key extension present in a SET root CA certificate.)

Because non-SET applications will not understand the private extensions SET defines, one cannot expect a non-SET application (for example, S/MIME-based e-mail) to accept a SET certificate for use. This is true even though the SET certificate format is compliant with an X.509 Version 3 public-key certificate. Although one might suggest that a non-SET application could ignore the SET extensions, the *Certificate Type* extension is critical; therefore, by definition a non-SET application must reject a SET certificate. Note that it is accepted practice to intentionally mark certain extensions critical so that a particular type of certificate can be used only in the context of a specific application to minimize liability concerns.

Figure 6.2 SET certificate structure.



Although it is not necessarily the case that end users will require a separate certificate for each application, this example helps illustrate that multiple certificates per end entity will be required. (Chapter 10 provides additional discussion regarding the requirements for multiple certificates.)

Attribute Certificates

Attribute certificates were first standardized in the 1997 version of X.509 when the basic ASN.1 constructs for an attribute certificate were defined. The 2000 version of X.509 expands on the definition and use of attribute certificates significantly and even describes an attribute certificate framework that can be used as a foundation for building Privilege Management Infrastructures (PMIs). The subject of PMI is rather extensive and could very well be the topic of another book. For the purpose of this book, it is sufficient to say that—although attribute certificates are defined in the X.509 Recommendation—attribute certificates are *not* public-key certificates. Attribute certificates are designed to convey (potentially short-lived) attributes about a given subject to facilitate flexible and scalable privilege management. The attribute certificate may point to a public-key certificate that can be used to authenticate the identity of the attribute certificate holder. (Chapter 5 provides additional information regarding privilege management.)

Certificate Policies

As indicated earlier, a number of policy-related extensions may be present in a given certificate. The policy-related extensions are extremely important in the sense that they help govern the acceptable use of the certificate in terms of policy compliance, potentially across multiple PKI domains.

The policy-related extensions refer either directly or indirectly to a certificate policy. The X.509 Recommendation [X.509–00] defines a *Certificate Policy* as

> 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.

The Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework [RFC2527] also adopts this definition.

This can be contrasted with the definition of a Certification Practice Statement (CPS):

> a statement of the practices which a certification authority employs in issuing certificates [ABA].

In general, it is agreed that (1) a Certificate Policy is a high-level statement of requirements and restrictions associated with the intended use of the certificates issued under that policy and (2) a CPS is an extremely detailed (and potentially extremely sensitive) document that describes the internal operating procedures of the CA and/or PKI that issues those certificates. What is not universally accepted is the role that these documents have with respect to end-user notice and multidomain cross-certification arrangements.

Interesting sources of information related to Certificate Policies and CPSs include the following:

- "The Internet X.509 Public Key Infrastructure: Certificate Policy and Certification Practices Framework" [RFC2527]
- CARAT Guidelines [CARAT] sponsored by the National Automated Clearing House Association (NACHA)
- The American Bar Association (ABA), Digital Signature Guidelines: Legal Infrastructure for Certification Authorities and Electronic Commerce [ABA]
- The Automotive Network eXchange (ANX) Certificate Policy [ANX]
- The Government of Canada Certificate Policy [GOCCP]
- The U.S. Federal PKI "Model Certificate Policy" [MCP]

From a high-level perspective, most of this documentation reflects the common theme noted earlier. In particular, a Certificate Policy is expected to be a higher-level document than a CPS, and it is typically concerned with *what* will be supported rather than *how* it will be supported. (See Box 6.3.) Conversely, a CPS is expected to be a fairly detailed and comprehensive technical and procedural document regarding the operation of the supporting infrastructure. For example RFC2527 points out that CPSs

> may be quite comprehensive, robust documents providing a description of the precise service offerings, detailed procedures of the life-cycle management of certificates, and more—a level of detail which weds the CPS to a particular (proprietary) implementation of a service offering.

Object Identifiers

To easily distinguish one Certificate Policy from another, each Certificate Policy is assigned a globally unique OID. One or more OIDs can be specified in the Certificate Policies certificate extension, which can be further qualified as appropriate. For example, RFC3280 defines two optional Certificate Policy qualifiers: a User Notice and a pointer to a CPS. Certificate Policies can be placed in end-entity certificates as well as CA certificates. Cross-certificates (as described in Chapter 9) may also contain the Policy Mappings extension, which permits a policy OID in

Box 6.3 The Role of Certificate Policies and CPSs

The role that Certificate Policies and CPSs may have is not universally agreed upon and, to a large extent, depends on the type of PKI domain in question.

For example, the model we see in the Web environment demonstrates that suppliers of Web server and end-user certificates (such as Verisign) publish their CPS on their Web site, and the certificates that they issue point to their on-line CPS. The CPS contains a great deal of information, including warranties and obligations meant to be conveyed to the end user. Thus, the CPS is a public-domain document, subject to the scrutiny of many. On the other hand, an enterprise PKI domain typically considers the CPS to be extremely sensitive, usually reserved for internal audit purposes. The CPS is rarely a public-domain document in these cases, nor is it used to convey information to the end users in the enterprise domain.

The role these documents have in forging cross-certification arrangements (cross-certification is described in Chapter 9) can also vary. One school of thought suggests that the CPS of one domain should be scrutinized against the CPS of another. On the other hand, the CPS may not form a suitable basis for establishing cross-certification (or more generally, interoperability arrangements) for a number of reasons. First, as noted earlier, the CPS tends to be an extremely detailed and voluminous document, which would make CPS comparisons tedious and labor intensive (barring standardized markup languages and automated tools, of course). Second, terminology variations from one CPS to another that make equivalency comparisons problematic may be present [PAG]. Third, a given organization may consider their CPS to be too sensitive to be released externally, even for the purposes of establishing a cross-certification agreement with another enterprise. It can therefore be argued that Certificate Policies form a much more suitable basis for establishing cross-certification agreements. In the future, we may even see the use of PKI Disclosure Statements for establishing interoperability agreements between enterprise domains. See `http://www.verisign.com/repository/disclosure.html` for an example of a PKI Disclosure Statement.

Given the above, some would suggest that only a CPS is required for a CA service provider and only a Certificate Policy is required for an organization. However, it has been our experience that a CPS is typically used in an enterprise context to document the internal operating procedures of the organization's PKI and that the CPS is often used for internal audit purposes. It is also possible for an organization to adopt the CPS of a third-party service provider when that third party supplies some or all of that organization's PKI services.

one domain to be designated equivalent to a policy OID in another domain. Thus, if two PKI domains have each defined a Certificate Policy for the exchange of their own internal e-mail and the two policies are deemed to be equivalent by each domain, defining yet a third policy OID to allow e-mail exchanges between the two domains is not needed.

Policy Authorities

Policy authorities (sometimes referred to as *policy management authorities*) establish Certificate Policies. The policy authority itself may vary from one organization to another. For example, each organization may establish its own policies under the authority of the internal Information Technology Security (ITS) department or equivalent. Alternatively, this authority may emanate from a policy advisory board made up of members from each major department in an organization. In concert with the internal authority (or perhaps in lieu of such an authority), an external policy authority may establish the Certificate Policies for a number of PKI domains that belong to the same community of interest. In any event, the applicable policy authority is responsible for registering Certificate Policies with the appropriate registration authority (for example, a national registration authority) so that the Certificate Policy OIDs can be assigned appropriately.

Certification Authority

In the context of a PKI, *certification* is the act of binding a subject name (and potentially other attributes) with a public key. As discussed previously in this chapter, this binding occurs in the form of a signed data structure referred to as a *public-key certificate*. A *Certification Authority (CA)* is responsible for issuing these public-key certificates. (See Box 6.4.) These certificates are digitally signed with the private key of the issuing CA.

Box 6.4 Certificate Authority versus Certification Authority

A CA is sometimes referred to as a *certificate authority* rather than a *Certification Authority* in much of today's literature. Although it may be too late to stop the growing use of this term, we would like to point out that using this term to denote a CA is technically (and logically) incorrect. There is no such thing as a "certificate authority" in X.509, and the implication that a CA is an authority on certificates is somewhat misleading. Specifically, a policy authority (or policy management authority) is the authority on certificates; the CA is simply an instrument that issues certificates in accordance with the Certificate Policy dictated by the policy authority. The term *Certification Authority* is used throughout this book because a CA is an authority on the process of certification.

Because the issuing CA digitally signs certificates, they are self-protected from an integrity perspective. Thus, the certificates can be freely disseminated, assuming that they do not contain any sensitive information. (In Chapter 11, we discuss the difficulties associated with the dissemination of certificates that might be considered sensitive in nature.)

The CA can take on a number of different representations, depending on the trust model embodied by that CA. For example, in an enterprise domain, one can expect one or more CAs to be responsible for issuing certificates to the employees of the enterprise. The employees essentially place their "trust" in the enterprise CA(s).[4] A completely different architecture is reflected in the PGP "web of trust" model where individuals can act as their own CA, and all trust decisions lie with the individual rather than a remote CA. (Chapter 9 provides a more detailed discussion regarding trust models and the role a CA plays in relation to those trust models.)

Registration Authority

Although the registration function can be implemented directly with the CA component, it sometimes makes sense to off-load the registration function to a separate component referred to as a *Registration Authority (RA)*. For example, as the number of end entities in a given PKI domain increases and/or the end entities are widely dispersed geographically, the notion of centralized registration becomes problematic. Judicious deployment of multiple RAs (sometimes referred to as *Local Registration Authorities,* or *LRAs*) helps solve this problem. The primary purpose of the RA is to off-load certain functions from the CA to enhance scalability and decrease operational costs.

Although the functions implemented by the RA may vary, it can be designed to support one or more of the following:

- Establish and confirm the identity of an individual as part of the initialization process. (For example, the RA might verify the identify of an individual through a combination of physical presence and associated identification such as a driver's license, employee badge with picture, or a passport.)
- Distribute shared secrets to end users for subsequent authentication during an on-line initialization process.
- Initiate the certification process with a CA on behalf of individual end-users (including the registration of certain attributes to be associated with the end user).

[4]We recognize that the interpretation of the word *trust* is often the subject of lively debate. Chapter 9 provides the definition of *trust* as it applies in the context of this book.

- Generate keying material on behalf of an end user.
- Perform certain key/certificate life-cycle management functions, such as to initiate a revocation request or a key recovery operation on behalf of an end entity.

Regardless of the set of functions implemented in the RA, it should be noted that a RA is *never* allowed to issue certificates or CRLs. These functions rest solely with the CA.

End-entity registration requirements may vary significantly from one domain to another, between distinct applications in a given domain, or between distinct contexts in a given application in a given domain. (Chapter 7 discusses specific registration issues and procedures further.)[5]

Summary

The primary focus of this chapter has been the structure and semantics of the X.509 Version 3 public-key certificate and the need for certification in order to maintain the integrity and trustworthiness of the certificate itself. The Version 3 X.509 public-key certificate is by far the preferred choice for the enterprise domain, and it is quickly becoming widely accepted in other environments such as the Internet. This chapter also introduced a number of other certificate types (which may or may not be encountered in wide-scale implementation practice).

We also addressed the importance and role of the CA and RA components. A CA is responsible for issuing certificates in accordance with one or more Certificate Policies. The CA may also be responsible for end-entity registration, although one or more RAs can implement this function separately. Deploying one or more RAs reduces cost and enhances the overall scalability of a large-scale PKI.

A full understanding of certificates and certification requires familiarity with two related topics: the details regarding key/certificate life-cycle management (see Chapter 7) and the concepts associated with trust models and certification path processing. (See Chapter 9.)

[5]Chapter 7 presents in detail all aspects associated with key/certificate life-cycle management (for example, registration and initialization, revocation, key recovery, and so on).