

ITPro™
SERIES

WindowsITPro eBooks

By Jan De Clercq

Understanding and Leveraging

SSL-TLS for Secure Communications



Contents

Chapter 1: Introducing SSL/TLS and Its Cryptographic Roots	1
The SSL/TLS Protocols	1
Cryptographic Roots	3
Cryptographic Terminology	3
Symmetric vs. Asymmetric Ciphers	5
Symmetric Ciphers	5
Asymmetric Ciphers	7
Comparing Symmetric and Asymmetric Ciphers	8
Hash Functions and Digital Signatures	8
Hybrid Cryptographic Solutions	10
S/MIME Operation	11
SSL/TLS Operation	12
Certificate and Public Key Infrastructure (PKI) Concepts	13
Certificates	13
PKI	16
Certification Authorities	16
Directories	18
PKI Policies	18
PKI Standards	19
Conclusion	20

Chapter 1:

Introducing SSL/TLS and Its Cryptographic Roots

This chapter provides a general introduction to the Secure Sockets Layer/Transport Layer Security (SSL/TLS) protocols and the security building blocks upon which these protocols are rooted. The chapter discusses the cryptographic roots of the protocols and related concepts, such as certificates and public key infrastructure (PKI).

The following chapters will provide more detail about the operation of the SSL/TLS protocols and how to implement and configure them to secure your applications. Here's a short overview of what's coming up in chapters 2, 3, and 4:

- Chapter 2 focuses on leveraging SSL/TLS for secure Web communications (HTTPS). The chapter explains how to set up SSL/TLS on a Web server and how to enable HTTP clients for SSL/TLS. It also compares SSL/TLS to other Web-authentication protocols.
- Chapter 3 covers advanced SSL/TLS topics for secure Web communications (HTTPS). These topics include the SSL/TLS certificate validation process, how to optimize SSL/TLS server-side performance, and how to deal with SSL/TLS in load-balancing setups. This chapter also provides best practices for implementing SSL/TLS for secure Web communications.
- Chapter 4 focuses on leveraging SSL for secure Lightweight Directory Access Protocol (LDAP), Network News Transfer Protocol (NNTP), and SMTP (LDAPS, NNTPS, and SMTPS). The chapter discusses how to set up SSL/TLS for secure LDAP, secure NNTP, and secure SMTP, and provides best practices for implementing SSL/TLS for these applications.

The SSL/TLS Protocols

The Secure Sockets Layer (SSL) protocol is a security protocol that sits in between the application and the transport layers of the TCP/IP networking stack (as Figure 1-1 illustrates). SSL can provide the following security services:

- *Server authentication.* SSL-enabled applications can use the X.509 server certificate to authenticate the server.
- *Data confidentiality and integrity services.* SSL provides channel encryption services, also referred to as secure channel services.
- *Client authentication.* SSL can use X.509 client certificates to authenticate the clients.

Figure 1-1

Positioning the SSL/TLS protocols in the TCP/IP networking stack

Application SMTP, HTTP
SSL/TLS
Transport TCP, UDP
Packet IP

SSL can provide these authentication, confidentiality, and integrity services to a wide range of application-level protocols; for example, HTTP, for secure Web communications; SMTP, for secure mail transfer operations; Network News Transfer Protocol (NNTP), for secure news operations. These services are also referred to as secure channel services. With its positioning in the TCP/IP stack, SSL can provide these secure channel services to application-layer protocols transparently, which means it does so without modifying the application-layer data.

Netscape initially developed SSL. You can download the latest SSL specification (3.0) from <http://wp.netscape.com/eng/ssl3>. In 1999, in RFC2246, the IETF standardized SSL under the name Transport Layer Security (TLS) protocol (you can find more information about this at <http://www.ietf.org/rfc/rfc2246.txt>). TLS is also referred to as SSL version 3.1. In its role as a new version of SSL, TLS also doubles the number of SSL error messages and includes some cryptographic optimizations.

The SSL/TLS protocols are built upon symmetric and asymmetric cryptographic protocols (also known as public key crypto) and X.509 certificates, concepts that we explain in detail later in this chapter.

From an operational point of view, SSL/TLS is a client-server handshake protocol. In short, this protocol is how an SSL/TLS connection is set up between, for example, a browser and a Web server. When SSL/TLS is used to secure the HTTP traffic between the browser and the Web server,

- The browser connects to the Web server using a secure URL (<https://>).
- The Web server sends the browser the server certificate (which contains the Web server's public key) for server authentication. The browser logic checks the server name in the certificate against the name provided in the URL.
- Optionally, the browser sends the Web server the client certificate (which contains the client's public key) for client authentication.
- The browser and Web server negotiate the encryption level to be used.

- The browser encrypts a session key (to be used for SSL channel encryption) with the Web server's public key, and then sends the encrypted blob to the Web server.
- The Web server uses its private key to decrypt the encrypted blob.
- The browser and Web server exchange data; the data are secured using the session key.

A more detailed explanation of the SSL/TLS operation follows in the “Cryptographic Roots” section.

SSL/TLS protocol support comes bundled with today's most popular Web servers, Microsoft Internet Information Services (IIS) and Apache. A popular open-source SSL/TLS implementation is the OpenSSL SSL/TLS implementation, which you can download free of charge from <http://www.openssl.org>.

Several software vendors sell C/C++ and Java SSL/TLS toolkits that you can use to SSL/TLS-enable your applications. Certicom (<http://www.certicom.com>), for example, sells both a C/C++ and Java SSL/TLS toolkit.

Cryptographic Roots

After you read this section, you should have a clear understanding of the basic cryptographic building blocks of the SSL/TLS protocols. These building blocks include symmetric crypto, asymmetric crypto, hashing, digital signatures, and hybrid cryptographic solutions.

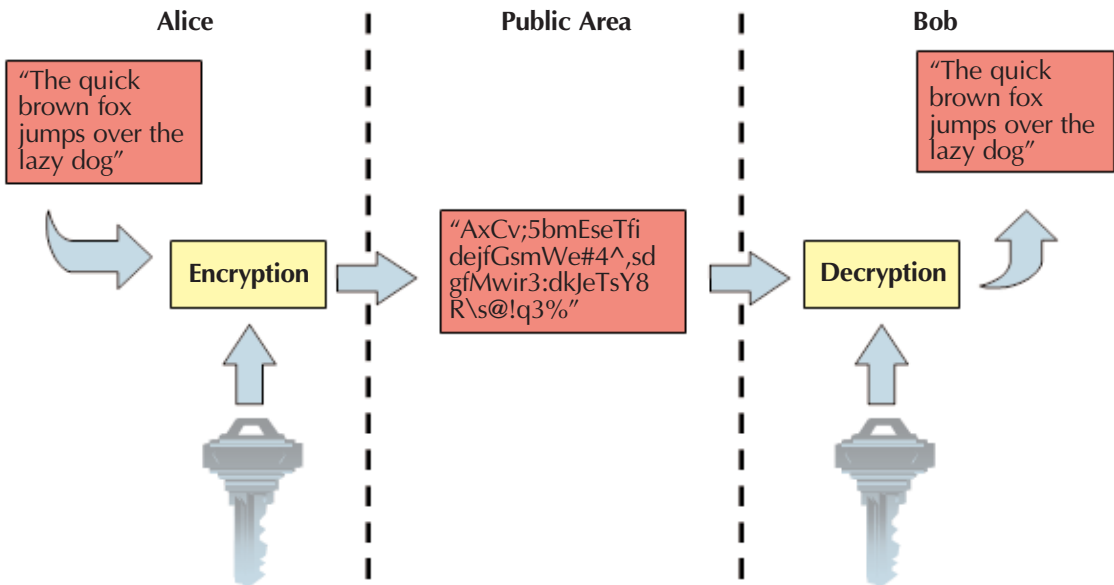
Cryptographic Terminology

On the most basic level, all security solutions, including certificates and PKI, rely on cryptography. Literally translated from Greek, *cryptography* is the science of “hidden writing.” This section does not provide all the details behind cryptography; instead, it gives enough information to help you understand the references to cryptography in the rest of this eBook. If you want to know all the details, read one of the following excellent books:

- *Handbook of Applied Cryptography* by Alfred Menezes, Scott A. Vanstone, and Paul C. Van Oorschot
- *Applied Cryptography* by Bruce Schneier
- *Secrets and Lies* by Bruce Schneier (Chapter 6 provides a good high-level overview of cryptography.)

Cryptography is based upon two processes: encryption and decryption (as Figure 1-2 illustrates). *Encryption* transforms the cleartext (or plaintext) into ciphertext. To get back to the plaintext from the ciphertext, the inverse transformation, *decryption*, is applied to the ciphertext.

Figure 1-2
Cryptographic terminology



In modern cryptography, both the encryption and decryption processes are based upon mathematics. The three basic cryptographic primitives, all of which are derived from mathematical functions, are symmetric ciphers, asymmetric ciphers, and hash functions.

- *Symmetric ciphers* use mathematical transposition and substitution functions. *Transposition* means that characters in the cleartext are moved to a different position in the ciphertext. For example, if "jan" is the cleartext, a possible ciphertext resulting from a transposition of the cleartext is "anj." *Substitution* means that characters are replaced by other characters; for example, the "j" by the "z," the "a" by the "r," and the "n" by the "j." In this example, the resulting ciphertext is "zrj."
- *Asymmetric ciphers* are built upon difficult or hard-to-resolve mathematical problems, such as the factorization problem, the discrete logarithm problem, or the elliptic curve theory. We explain the operation of asymmetric ciphers in more detail in the following sections.
- *Hash functions* use mathematical one-way functions. We explain the operation of hash functions in more detail in the following sections.

The three primitives listed above all have different characteristics; thus, all serve different security goals. Most security solutions combine these three primitives in what is known as *hybrid cryptographic solutions*, which we explain in the "Hybrid Cryptographic Solutions" section later in the chapter.

In most cryptographic solutions, the algorithms behind the primitives are public, which makes it possible for us to analyze them and develop more efficient implementations. Because the algorithms are public, however, we need something else to provide secrecy, and that's the goal of a cryptographic key. This key, which is based on a public algorithm, is a secret input parameter for a cryptographic process (as Figure 1-2 illustrates).

The use of keys promotes the reuse of the same algorithm for different secure communications. Private algorithms provide secrecy because only the users of the algorithms know them. And although public algorithms are available to everyone, they can be reused for secret communications because of the secret-key concept.

When you are evaluating different cryptographic solutions, always consider the following criteria:

- The cost of breaking the cipher should exceed the value of the encrypted information.
- The time required to break the cipher should exceed the useful lifetime of the information.

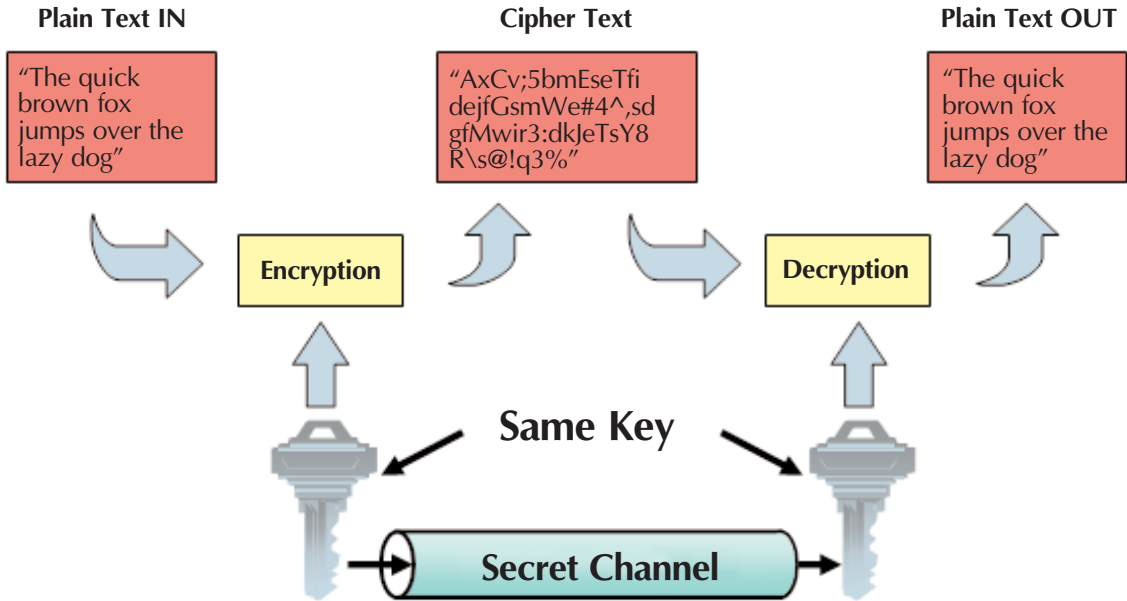
Symmetric vs. Asymmetric Ciphers

In this section, we differentiate between two basic ciphers—symmetric and asymmetric—based on the key material that is used. We also discuss some of the advantages and disadvantages of these two ciphers.

Symmetric Ciphers

Symmetric ciphers are the oldest and most-used cryptographic ciphers. In a symmetric cipher, the key used to decipher the ciphertext is the same as (or can be easily derived from) the key that is used to encipher the cleartext (as Figure 1-3 illustrates). This key is called the *secret key*. This cipher provides secrecy by sharing the secret key only between the participants to the secure communication process. In practice, retaining this secrecy requires the key generated at one side of the communication channel to be sent to the other side of the channel using a secret channel. “Secret channel” in this context means protecting the confidentiality of the key, or keeping the key secret. The most widely used symmetric cipher is the *Data Encryption Standard (DES)*.

Figure 1-3
Symmetric cipher



The strength of a symmetric cipher depends on the following three variables:

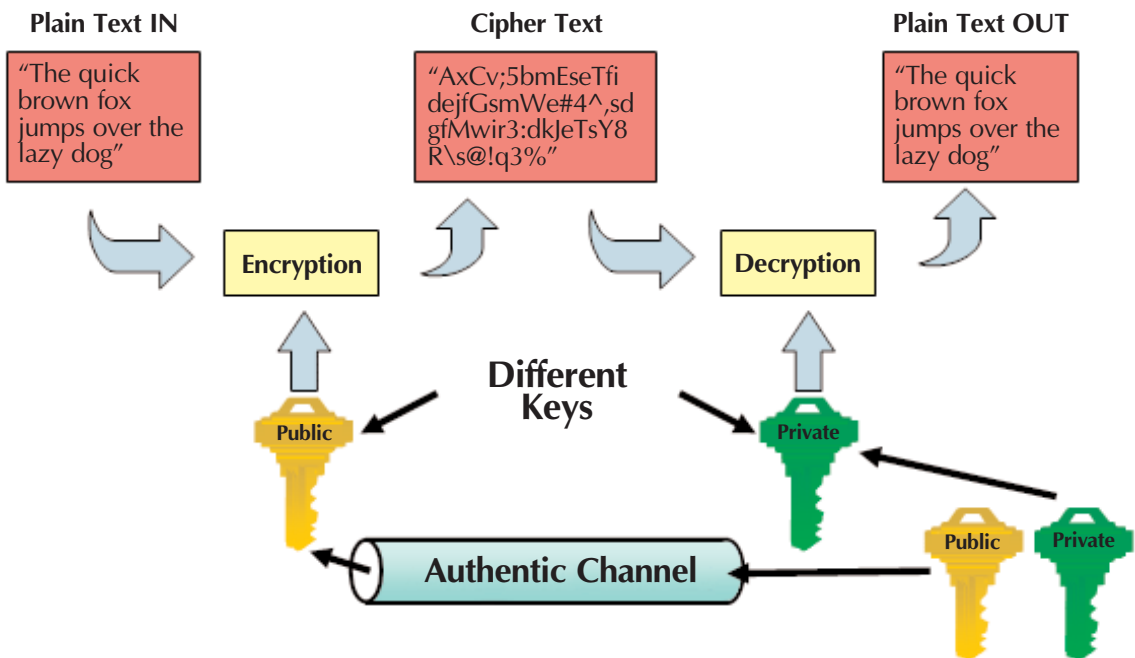
1. The block length, or the number of bits, upon which the symmetric cipher operates during one round. DES operates on 64-bit blocks.
2. The key length. The longer the key, the bigger the keyspace, and the more difficult it becomes for a hacker to guess the right key. A typical DES key is 56 bits long.
3. The random-number generator used to generate the symmetric key. If the same key is generated several times, it becomes easier to guess the key used for the encryption. This variable is an often overlooked critical parameter.

The above parameters and the constantly increasing processing power of computer systems explain why the original implementation of the DES is not safe anymore, and why it is being replaced by stronger algorithms derived from the original DES, such as DESX and 3DES. A new U.S. Federal standard for symmetric encryption, the *Advanced Encryption Standard (AES)*, will replace DES. AES supports block sizes of 128 bits and key sizes of 128 bits, 192 bits, and 256 bits. AES operates on 128-bit blocks and can use variable key sizes (128 bits, 192 bits, or 256 bits). In October 2000, the National Institute of Standards and Technology (NIST) selected Rijndael as the cipher for the AES. You can find more information about Rijndael at <http://www.esat.kuleuven.ac.be/cosic>. To get the latest news on the AES development, surf to <http://src.nist.gov/CryptoToolkit/aes/rijndael/>.

Asymmetric Ciphers

Asymmetric ciphers are the most recent cryptographic ciphers. Contrary to a symmetric cipher, an asymmetric cipher uses two keys: one key that is kept secret and is known to only one person (the private key), and another key that is public and available to everyone (the public key). The two keys are mathematically interrelated, but it is impossible to derive one key from the other. An important advantage of an asymmetric cipher is that its secrecy depends not on the secrecy of the key that is exchanged between the communicating entities (the public key), but rather on the secrecy of the key that is never exchanged (the private key), as Figure 1-4 illustrates. Consequently, an asymmetric cipher does not require a secret channel for public key exchange—only an authentic channel. “Authentic channel” in this context means the recipient of the public key is assured of the origin of the key. Well-known asymmetric ciphers are the Diffie-Hellman algorithm, RSA, and the directory service agent (DSA).

Figure 1-4
Asymmetric cipher



The strength of an asymmetric cipher depends on the difficulty of resolving the mathematical problem behind it, and on the key length. Let's illustrate this relationship using the best-known asymmetric cipher, RSA.

RSA is based on the mathematical problem of factorization. This problem tells us that it is easy to generate two large prime numbers and to multiply them, but doing the reverse—that is, finding the two prime factors of a large number—requires a huge amount of computation.

8 Understanding and Leveraging SSL-TLS for Secure Communications

The biggest factorable number is growing constantly because of the ever-increasing processing power; currently, a key length of 1,024 bits is recommended for personal use of RSA. If you want military-grade security, you will need 4,096-bit keys.

Comparing Symmetric and Asymmetric Ciphers

There are both advantages and disadvantages to using an asymmetric cipher rather than symmetric ciphers. Here are some advantages:

- An important advantage is that no secret channel is needed for the exchange of the public key. The receiver only needs to be assured of the authenticity of the public key. Symmetric ciphers require a secret channel, generated at one side of the communication channel, to send the secret key to the other side.
- Asymmetric ciphers create fewer key management problems than symmetric ciphers: Only $2n$ keys are needed for n entities to communicate securely with one another. In a system based on symmetric ciphers, we need $n(n-1)/2$ secret keys. In a 5,000-employee organization, for example, the deployment companywide of a symmetric crypto-based security solution requires more than 12 million keys. The deployment of an asymmetric solution requires only 10,000 keys.
- Asymmetric ciphers, because of their unique mathematical background, are the only ciphers that can provide nonrepudiation services. In short, if the signature on a digital document can be validated using a valid public key, the user cannot deny having signed the message using his or her private key. (True nonrepudiation services not only require the use of asymmetric ciphers but also the presence of a trusted time-stamping service and a legal or policy framework that fixes the liabilities and responsibilities of all parties involved.)
- Symmetric ciphers can be cracked by doing a “brute-force attack,” in which all possible keys are tried out until the right key is found.

A disadvantage of asymmetric ciphers compared to symmetric ciphers is that they tend to be about 1,000 times slower than symmetric ciphers. “Slower” means that it takes about 1,000 times more CPU time to process an asymmetric encryption or decryption than to process a symmetric encryption or decryption.

Because of all these characteristics, asymmetric ciphers are typically used for data authentication (through digital signatures), for the distribution of a symmetric bulk encryption key (this technique is also known as a digital envelope), nonrepudiation services, and key agreement. Symmetric ciphers are used for bulk encryption.

Hash Functions and Digital Signatures

A hash function is a cryptographic primitive that can be used for integrity, confidentiality, and authentication services. The hash function takes an input of an arbitrary length and reduces it to a fixed-size string. The output of a hash function is also known as a *digital fingerprint*, or a “hash.” Hash functions are one-way functions, which means that, given the unique fingerprint, it is impossible to obtain the original text. Well-known examples of hash functions are MD5 (MD stands for Message Digest) and the Secure Hash Algorithm (SHA). MD5 generates a 128-bit hash; SHA generates a 160-bit hash.

The primary task of hash functions is to offer a detection mechanism against message tampering. The sender first calculates the hash of the message and attaches it to the message. If the original message is changed during transmission, reapplying the same hash function will result in another hash. This system offers the receiver a way to check whether the integrity of the original message is still valid after it has been transmitted over a public medium.

Hash functions are often used to protect the confidentiality of passwords. Storing the hash of a password in a centralized database guarantees that nobody can get to the password; at the same time, this approach provides a way to check whether a given password is valid. Hash functions are also a building block for cryptographic authentication solutions such as Message Authentication Codes (MACs) and digital signatures.

MACs can provide data-origin authentication. MACs are the result of a hash function and a symmetric cipher. Before a message is signed with a secret key (known only by the two communicating parties), it is compressed using a hash function.

Digital signatures can provide data origin authentication and identification. Identification is also known as the authentication of persons. A digital signature combines an asymmetric cipher with a hash function. Figures 1-5 and 1-6 illustrate the process of digital signature generation (sender side) and verification (receiver side).

Figure 1-5

Digital signature: creation on sender side

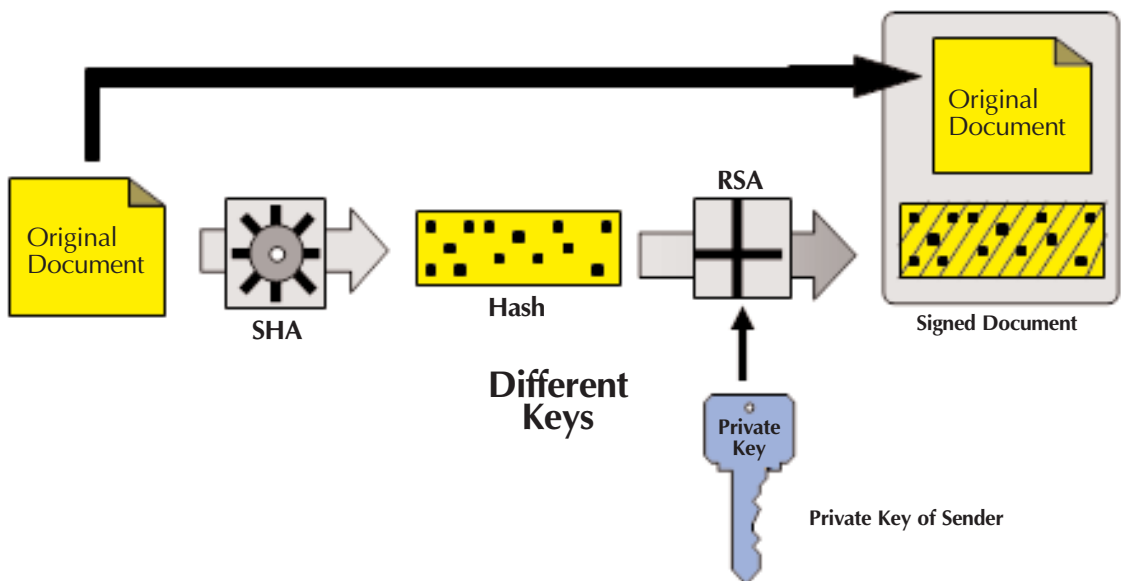
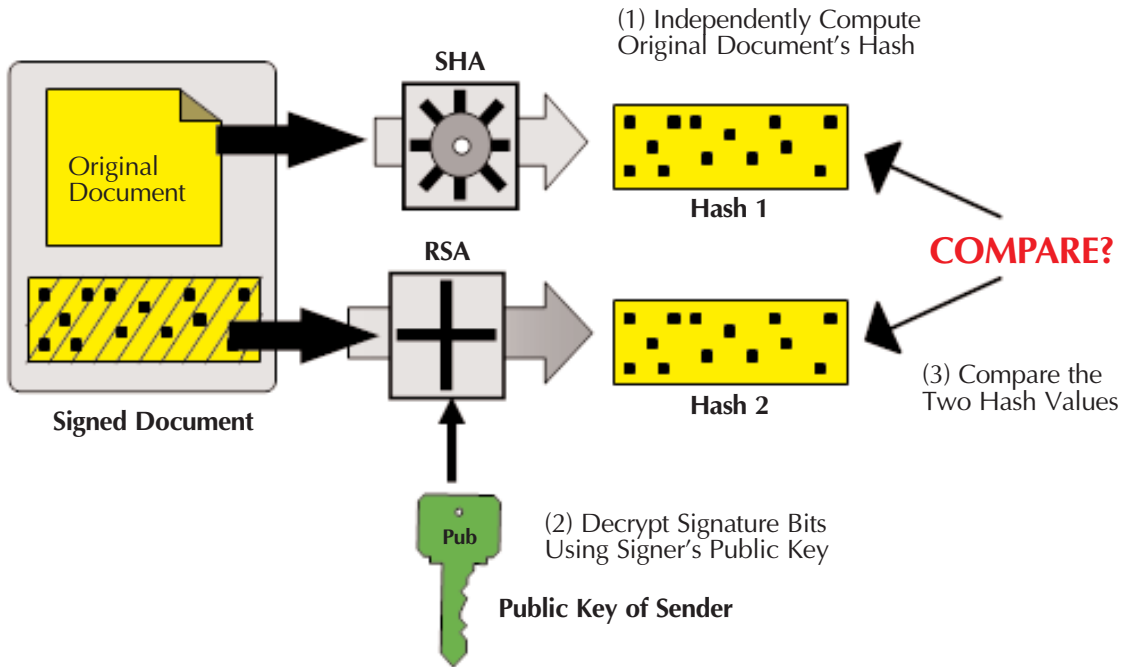


Figure 1-6*Digital signature: verification on receiver side*

Using a hash function, the process computes a unique hash value from the original (unencrypted) message, which is then encrypted using the sender's private key and added to the end of the message (the message digest), as Figure 1-5 illustrates. The recipient opens both the message and the digital signature, and uses the sender's public key to decrypt the hash value (step 2 in Figure 1-6). The same hash function is used to recompute the hash on the message (step 1). This hash will differ from the original (the one coming with the message) if the message contents are off by even 1 bit (step 3). If they are different, the recipient is notified that the contents of the message might have changed.

Hybrid Cryptographic Solutions

Most real-world security problems that require a high-quality security solution are solved by combining basic cryptographic primitives (symmetric ciphers, asymmetric ciphers, and hashing algorithms) into a hybrid cryptographic system.

A hybrid system uses a symmetric key and algorithm to encrypt the plaintext. It then uses the recipient's public key to encrypt the symmetric key (into what is called a *lockbox*) and attaches to the message both the lockbox and the encrypted information being sent to the recipient. This technique is also known as the creation of a *digital envelope*. The recipient uses the private key to decode the lockbox and retrieve the symmetric key with which the message can be decrypted. This method uses fast symmetric key encryption on the bulk of the message and retains the benefits of more-secure

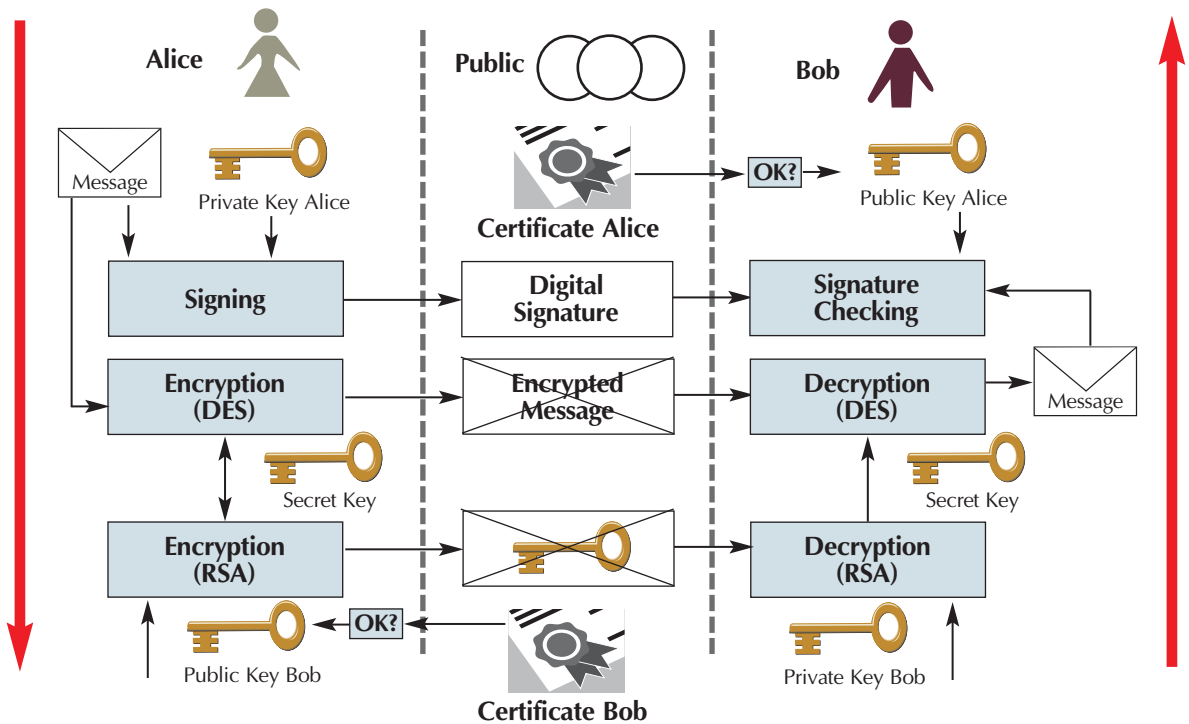
public-key encryption. By using the recipient's public key to encrypt the lockbox it also ensures that only the intended recipient (who has a private key) can get to the symmetric key. Two examples of hybrid cryptographic solutions are the SSL/TLS protocols and S/MIME, a standard you can use to create secure email messages.

S/MIME Operation

Figure 1-7 illustrates S/MIME, a protocol for secure messaging and an excellent example of a hybrid cryptographic solution. In a typical S/MIME scenario, Alice wants to send a secure message to Bob. "Secure" means guaranteeing confidentiality, integrity, authentication, and nonrepudiation.

Figure 1-7

Hybrid cryptographic solution: S/MIME



We can split the S/MIME exchange illustrated in Figure 1-7 into six steps, as follows:

1. Using her private key, Alice creates a digital signature for the message.
2. Using a bulk encryption key, Alice encrypts the message.
3. To create a secure channel that protects the confidentiality of the encryption key, Alice encrypts the encryption key with Bob's public key. This process results in the lockbox.
4. Bob decrypts the lockbox using his private key, which results in the bulk encryption key.

12 Understanding and Leveraging SSL-TLS for Secure Communications

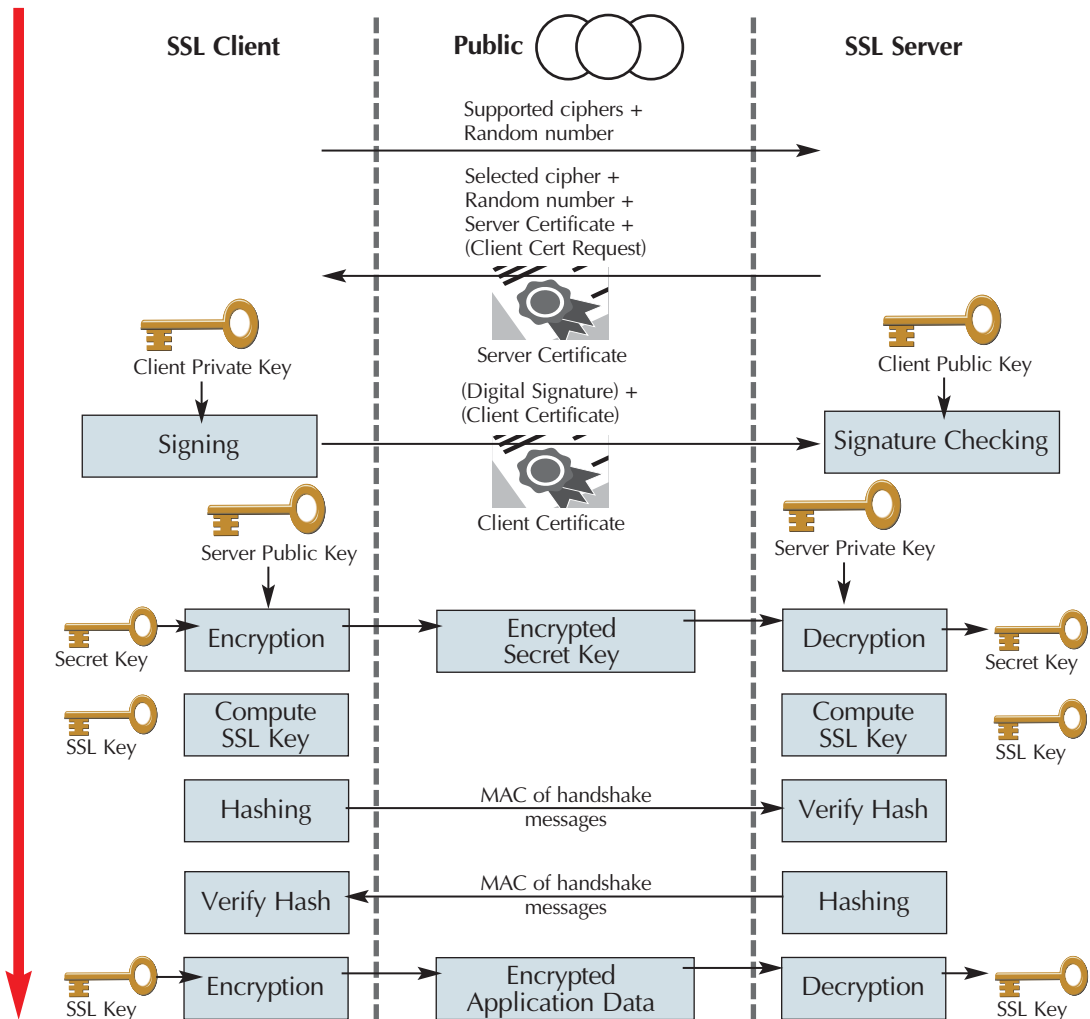
- Using the bulk encryption key, Bob decrypts the message. This action gives Bob the readable message.
- Bob verifies the authenticity and integrity of the message using Alice's public key to verify the digital signature.

SSL/TLS Operation

Figure 8 illustrates the SSL/TLS protocol operation, which is another good example of a hybrid cryptographic solution.

Figure 1-8

Hybrid cryptographic solution: SSL/TLS



We can split the SSL/TLS exchange illustrated in Figure 1-8 into eight steps, as follows:

1. The SSL client informs the SSL server about the cryptographic ciphers it supports, and then forwards a random number (which was generated on the client side).
2. The SSL server selects a cryptographic cipher, forwards a random number (generated on the server side), forwards its SSL server certificate, and optionally requests the client SSL certificate.
3. The SSL client optionally authenticates to the SSL server using a digital signature. The SSL client certificate is attached to the digital signature. The SSL server validates the client's digital signature.
4. The SSL client generates a secret key (based on the previously exchanged random numbers) and securely sends that key to the SSL server. The secure channel is created by encryption of the secret key with the SSL server's public key.
5. Both the SSL client and the SSL server derive the SSL key from the secret key. The SSL key will be used in step 8 to provide secure channel services for the application-layer data that are transmitted using SSL/TLS.
6. The SSL client calculates a hash value of the handshake messages previously exchanged. This hash provides a mechanism to check the integrity of the handshake messages.
7. The SSL server calculates a hash value of the handshake messages (for the same reason as that stated in step 6).
8. The application-layer data are securely transmitted using the previously generated SSL key.

Certificate and Public Key Infrastructure (PKI) Concepts

To use the cryptographic building blocks in security applications such as the SSL/TLS protocols requires a supporting security infrastructure. The following sections introduce the key concepts associated with the security infrastructure used in an SSL/TLS setup: certificates and public key infrastructure (PKI).

Certificates

A certificate is a digital document that confirms the binding of a public key to an entity. The certificate creates confidence in the legitimacy of an entity's public key. In other words, a certificate provides a tool to verify the claim that a specific public key belongs to a specific entity. A digital signature is added to the certificate for this reason. The issuer of the certificate, a trusted certification authority (CA), applies the digital signature using its private key.

In addition to a public key and the entity's identity, a certificate contains other data: the certificate serial number, the issuing CA's identity, and so on. A standard format for certificates is defined in the International Telecommunications Union Telecommunication Standardization Sector (ITU-T) X.509 standard. The International Engineering Task Force (IETF) later adopted this standard in RFC 2459. You can download a copy of the ITU standard from the ITU-T Website at <http://www.itu.int/ITU-T/>. The RFC is available from <http://www.ietf.org>.

So far, three versions of the X.509 standard have been published. Most PKI software products support X.509 V3. The most important characteristic of X.509 Version 3 is its support for certificate extensions. Using extensions extra fields can be added to the certificate. Some extensions have been predefined in the X.509 standard, but an organization might as well decide to add its own custom extensions.

14 Understanding and Leveraging SSL-TLS for Secure Communications

Certificates have a limited lifetime. An X.509 certificate's Validity field contains a start and an end date that delimit the lifecycle of the certificate. This validity limitation helps the users and administrators of certificates cope with advances in cryptography and computer science.

Certificates can be revoked. Revocation is required when a certificate's associated private key is compromised. For example, PKI users might lose his or her laptop, which contains all his or her private keys. When this occurs the administrator of the CA will add the certificate (that's linked to the compromised private key) to a blacklist. This blacklist is called the *certificate revocation list (CRL)*. The CRL must be checked every time the certificate is used. Not to doing so may lead to dangerous situations. One might, for instance, encrypt a message with a public key from which a hacker has stolen the corresponding private key. The X.509 standard also defines a format for CRLs. Another solution for providing certificate revocation-checking services is the Online Certificate Status Protocol (OCSP). We discuss both CRLs and OCSP in more detail later in this eBook.

When you are dealing with certificates in a Windows environment, an interesting tool for viewing the content and formatting of X.509 certificates is the built-in Windows certificate viewer. The certificate viewer displays a certificate's content (as Figure 1-9 illustrates). This is the Windows default certificate viewer that is automatically started when you double-click a certificate file. You can view certificate files from the Certificates MMC snap-in. The first tab of the viewer shows general information such as the applications the certificate can be used for, the subject name, the issuer name, the validity period of the certificate, and whether the certificate has a corresponding private key stored in the user's profile. The second tab shows all X.509 certificate entries: the standard fields, and the critical and noncritical certificate extensions. The third tab shows the certification path and the certificate status.

Figure 1-9
The Windows certificate viewer

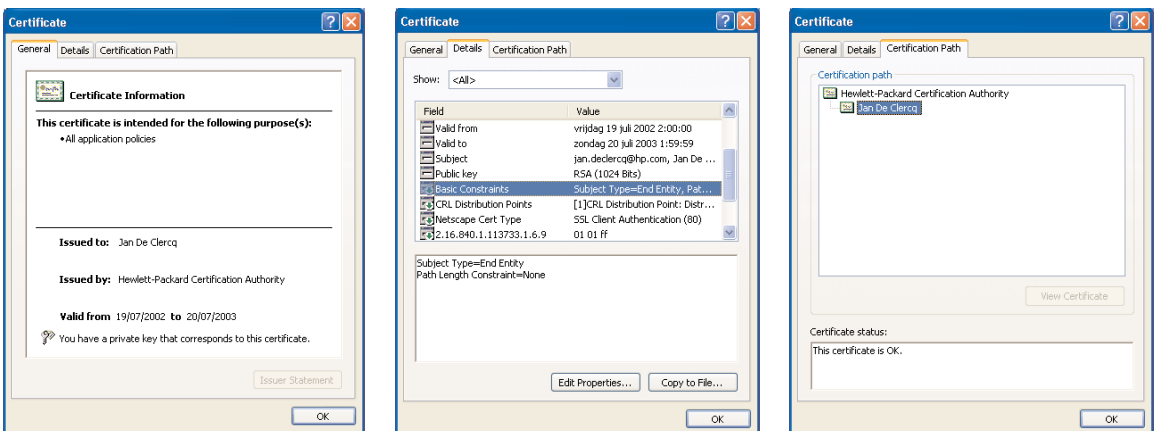


Table 1-1 lists and explains the different fields that make up an X.509 certificate and CRL. You can find more detailed information in the ITU-T X.509 Recommendation (version 03/2000), which you can download from the ITU Website at <http://www.itu.int>.

Table 1-1 X.509 formatting

X.509 Certificate Format		
X.509 Field Name	Field Meaning	X.509 Version/Optional-Required/Criticality for Extensions
Version	Identifies the X.509 version of the encoded certificate.	V1 Required
SerialNumber	Identifies the unique serial number of the certificate. The serial number together with the issuer name identify a unique certificate.	V1 Required
Signature	Contains the algorithm identifier for the algorithm and hash function used by the CA when it signs the certificate.	V1 Required
Issuer	Identifies the entity that has signed and issued the certificate.	V1 Required
Validity	Identifies the start and end date of the certificate, or the time interval during which the CA warrants that it will maintain status information of the certificate.	V1 Required
Subject	Identifies the entity associated with the public key found in the subject public key field.	V1 Required
SubjectPublicKeyInfo	Carries public key being certified and identifies the algorithm of which the public key is an instance.	V1 Required
IssuerUniqueIdIdentifier	Uniquely identifies an issuer in case of a name reuse.	V2 Optional
SubjectUniqueIdIdentifier	Uniquely identifies a subject in case of a name reuse.	V2 Optional
Extensions	Allows the addition of new fields to the certificate structure.	V3 Optional
<i>AuthorityKeyIdentifier</i>	Identifies the public key to be used for certificate signature verification.	V3 Optional—Always Noncritical
<i>SubjectKeyIdentifier</i>	Identifies the public key being certified.	V3 Optional —Always Noncritical
<i>KeyUsage</i>	Identifies the purpose for which the certified public key is used.	V3 Optional—Critical or Noncritical
<i>PrivateKeyUsagePeriod</i>	Indicates the period of use of a private key corresponding to certified public key.	V3 Optional—Always Noncritical
<i>Certificate Policies</i>	Identifies the certificate policies, recognized by the issuing CA, that apply to this certificate.	V3 Optional—Critical or Noncritical
<i>PolicyMappings</i>	For CA certificates only. Maps the certificate policy defined in one domain to the policy in another domain.	V3 Optional—Always Noncritical
<i>SubjectAltName</i>	Identifies alternative names for the certificate subject. Critical or Noncritical	V3 Optional—
<i>IssuerAltName</i>	Identifies alternative names for the certificate issuer.	V3 Optional—Critical or Noncritical
<i>SubjectDirectoryAttributes</i>	Lists directory attributes for the certificate subject.	V3 Optional—Always Noncritical
<i>BasicConstraints</i>	“CA” field: Answers the question, Can the public key listed in this certificate be used to verify other certificates? “PathLengthConstraint” field: Designates the maximum number of certificates that can follow this certificate in certification path.	V3 Optional—Critical or Noncritical
<i>NameConstraints</i>	Indicates name space within which all subject names in subsequent certificates in a certification path shall be located.	V3 Optional—Critical or Noncritical

Table 1-1 X.509 formatting

X.509 Certificate Format <i>continued</i>		
X.509 Field Name	Field Meaning	X.509 Version/Optional-Required/Criticality for Extensions
<i>PolicyConstraints</i>	Specifies constraints that might require explicit certificate policy identification or inhibit policy mapping for the remainder of the certification path.	V3 Optional—Critical or Noncritical
<i>InhibitAnyPolicy</i>	Specifies that “any-policy” is not considered an explicit match for other certificate policies	V3 Optional—Critical or Noncritical
<i>CRLDistributionPoints</i>	Identifies the CRL Distribution Point to which a certificate user should refer to ascertain whether the certificate has been revoked.	V3 Optional—Critical or Noncritical
Signature	Designates a digital signature on certificate content.	V1 Required

PKI

A Public Key Infrastructure (PKI) provides a set of security building blocks that distributed applications can use to provide strong security services to their users. Among the security building blocks that a PKI can offer are identification, data authentication, confidentiality, integrity and nonrepudiation.

- *Identification* gives an entity a way to check another entity’s identity.
- *Data authentication* provides a way to ensure that the alleged sender of a message is the one from whom the message originates.
- *Confidentiality* is a service that protects against unauthorized disclosure of information.
- *Integrity* is a security service that protects against undetected modification of a message.
- *Nonrepudiation* protects against denial, by one of the entities involved in a communication, of having participated in all or part of the communication. In a paper-and-pencil world, a manual signature provides nonrepudiation.

It’s important to stress the last character of the PKI acronym: A PKI is an *infrastructure*, which means many applications can build on it to provide strong security.

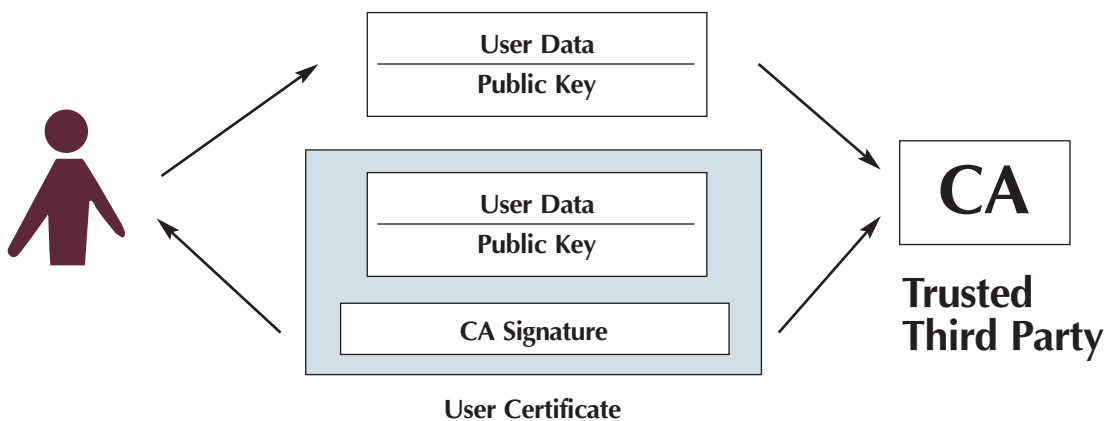
As we discussed earlier in the chapter, at the heart of a PKI is cryptography, and more specifically, asymmetric cryptographic ciphers. Asymmetric ciphers deal with public keys and private keys. A PKI provides services to manage these keys and their entire lifecycle. Among these services are certification, user registration, key generation, key update, certificate publishing, certificate renewal, certificate reissuing, and certificate revocation.

Certification Authorities

Certification Authorities (CAs) are trusted third parties that issue public key certificates to entities that trust them and that the CA can authenticate. These entities can be users, machines, and applications. When a CA issues a certificate, that action establishes and vouches for the authenticity of the entity’s public key. To do so, the CA applies a digital signature on the certificate’s content (including the entity’s public key) and adds the signature to the certificate. To verify a certificate issued by a CA, one needs the CA’s public key, by which the digital signature can be verified.

In most PKI-enabled applications, a key pair, which consists of a private and a public key, is generated on the machine of the requesting entity. Once generated, the private key is securely stored locally. The public key is presented to a CA for certification. “Certification” means that CA will sign the content of the certificate using his or her own private key. An often-overlooked process is identification of the certificate requestor. Before the CA actually generates the certificate, he or she must be sure that the public key (and with it the private key) belongs to the entity requesting the certificate. Figure 1-10 illustrates the role of the CA.

Figure 1-10
CA role



Because CAs have a public key and a private key, they also have their own certificate that certifies their public key. CAs either sign their own certificates or have another CA sign them.

A critical component of a PKI is the CA’s private key. If this key is compromised the complete PKI and, with it, the trust infrastructure fall down. You can use any of a number of different strategies to protect against, or at least to minimize the risks of, possible CA private-key compromise:

- Keep the CA and its private key offline, disconnected from the network, when it is off duty.
- Store the CA private key on special hardware device such as a Hardware Security Module (HSM).
- Provide adequate logical, physical, organizational, and communication security measures for the CA. [Production: The following are sub-bulleted items under this item]
 - *Logical security*: Provide the highest possible level of platform (operating system) security settings in the areas authentication, access control, and auditing on the CA server.
 - *Physical security*: Provide tamper-resistant and fault-tolerant racks to host the CA hardware. Implement strict access-control procedures on the level of the CA rack, the remote access software (console), and the computer rooms where the CAs are housed.
 - *Organizational security*: Document all security procedures, and provide awareness of the importance of CA private-key security within the IT community.
 - *Communications security*: Keep the CA servers disconnected from the network, or install a firewall to protect the subnet that hosts the CA servers.

18 Understanding and Leveraging SSL-TLS for Secure Communications

In large PKI setups, registration authorities are the PKI users' primary point of contact for a CA. A registration authority typically deals with PKI user identification and enrollment.

Organizations can build their proper in-house CAs and PKI, or they can rely upon commercial CAs to provide certificates to their applications. Often, CA-provided certificates are the simplest and most cost-effective approach for small organizations and those organizations that require few certificates. Examples of products you use to build your own PKI are the Entrust and Microsoft PKI solutions. Examples of commercial CAs are Thawte and Verisign.

Directories

A PKI uses a directory to store certificates and CRLs. Most PKIs require the directory to be Lightweight Directory Access Protocol (LDAP) and X.500 compliant. The most critical requirement for a directory from a PKI point of view is its availability. PKI users should have permanent access to other users' certificates and revocation lists. In most environments, availability of the directory is even much more critical than the availability of the certification authorities.

PKI Policies

An important nontechnical aspect technology-oriented PKI planners often forget is the creation of the following PKI-related documents. These documents include the security policy, certificate policy, and certification practice statement (CPS).

The security policy is a high-level document created by the corporate IT group. The security policy defines a set of rules regarding the use and provision of security services within the organization, and it should reflect your organization's business- and IT strategies. The security policy should answer the following PKI-related questions:

- What applications should be secured with certificates?
- What kind of security services should be offered using certificates?

Below is a short list of resources you can consult for defining your security policy:

- RFC 2196—the *Site Security Handbook*—available from <http://www.ietf.org/rfc/rfc2196.txt?number=2196>.
- The ISO 17799 / BS 7799 standard—*The Code of Practice for Information Security Management*, which you can purchase from <https://www.bspsl.com/secure/17799/cvm.cfm>.

A Certificate Policy is linked to a certificate type issued by a CA. The certificate policy focuses on certificate characteristics such as usage, enrollment procedure, and liability issues. It should answer questions such as

- What type of applications can the certificate be used for?
- How can a user enroll for the certificate?
- How are users identified when they request a certificate?
- What is a certificate's lifetime?
- How is renewal defined? Is a new key pair generated at every certificate renewal?
- What key lengths and ciphers are used to generate the certificate?
- Where is the private key stored?
- What is the liability when the issuing CA is compromised (e.g., when a malicious person gets access to the CA's private key) or when users lose their private keys?

Below is a short list of resources you can consult to learn more about certificate policies:

- RFC 2527—*Internet X.509 PKI Certificate Policy and Certification Practices Framework*—available from <http://www.ietf.org/rfc/rfc2527.txt?number=2527>
- *X.509 Certificate Policy of the United States Department of Defense*—available from http://www.defenselink.mil/nii/org/sio/ia/pki/DOD_CP_V7.0_18Dec2002_R.pdf

The certification practice statement (CPS) translates certificate policies into operational procedures on the CA level. The certificate policy focuses on a certificate; the CPS focuses on a CA. A CPS answers questions such as

- What policy is linked to the CA?
- How are certificates issued? Are they issued directly to users, or into a directory?
- Who can administer the CA? What subtasks are delegated to the different administrators?
- How is certificate revocation handled? When is a certificate revoked? What are the conditions for revocation? Where are CRLs published? How often are the CRLs updated?
- How is access to the CA physically and logically secured?
- Who is responsible for backing up the CA?
- How is the quality of the CA certificate and private key determined? What is the lifetime of the keys and the certificate? Where is the private key stored?

Below is a short list of resources you can consult to learn more about CPSs:

- The Thawte CPS—available from <http://www.thawte.com/cps>
- RFC 2527—*Internet X.509 PKI Certificate Policy and Certification Practices Framework*—available from <http://www.ietf.org/rfc/rfc2527.txt?number=2527>

PKI Standards

Certainly, standards for security solutions are very important for interoperability reasons. We've already mentioned the use of the X.509, LDAP, and X.500 standards in PKI environments. Two other PKI-related standards we have not yet discussed are Public-Key Cryptography Standards (PKCS) and Public Key Infrastructure for X.509 Certificates (PKIX)

The PKCS are a set of PKI-related standards (15 so far) that define message formats for certificate requests, certificate transport, Diffie-Hellman key agreement, RSA encryption and signing, and so on. A complete list of the PKCS standards is given below. More information on the PKCS standards is available from the Web site of RSA security at <http://www.rsasecurity.com/rsalabs/node.asp?id=2124>.

- PKCS #1: RSA Encryption Standard
- PKCS #2: Integrated in PKCS#1
- PKCS #3: Diffie-Hellman Key-Agreement Standard
- PKCS #4: Integrated in PKCS#1
- PKCS #5: Password-Based Encryption Standard (PBE)
- PKCS #6: Extended-Certificate Syntax Standard
- PKCS #7: Cryptographic Message Syntax Standard
- PKCS #8: Private-Key Information Syntax Standard
- PKCS #9: Selected Attribute Types for Use in Other PKCS Standards
- PKCS #10: Certification Request Syntax Standard
- PKCS #11: Cryptographic Token Interface Standard

20 Understanding and Leveraging SSL-TLS for Secure Communications

PKCS #12: Personal Information Exchange Syntax Standard

PKCS #13: Elliptic Curve Cryptography Standard

PKCS #14: Pseudo Random Number Generation

PKCS #15: Cryptographic Token Information Format Standard

PKIX is a PKI standardization effort that is driven by the IETF. The PKIX working group has already produced several important RFCs standardizing technical PKI aspects, such as PKI message formats and protocols. You can find an extensive list of the PKIX standards at <http://www.ietf.org/html.charters/pkix-charter.html>.

Conclusion

SSL/TLS has become a key security enabler for today's applications. An understanding of the cryptographic roots and underlying security infrastructure of SSL/TLS eases its adoption. The following chapters provide more detail about how to enable and configure your applications to use SSL/TLS.