Chapter 7
PKI and Cryptographic Applications

THE CISSP EXAM TOPICS COVERED IN THIS CHAPTER INCLUDE:

- ✓ **Domain 3: Security Architecture and Engineering**
  - 3.9 Apply cryptography
    - 3.9.1 Cryptographic lifecycle (e.g., key management, algorithm selection)
    - 3.9.2 Cryptographic methods
    - 3.9.3 Public Key Infrastructure (PKI)
    - 3.9.4 Key management practices
    - 3.9.5 Digital signatures
    - 3.9.6 Nonrepudiation
    - 3.9.7 Integrity
    - 3.9.8 Understand methods of cryptanalytic attacks
    - 3.9.9 Digital Rights Management (DRM)

In Chapter 6, “Cryptography and Symmetric Key Algorithms,” we introduced basic cryptography concepts and explored a variety of private key cryptosystems. These symmetric cryptosystems offer fast, secure communication but introduce the substantial challenge of key exchange between previously unrelated parties.

This chapter explores the world of asymmetric (or public key)

cryptography and the public-key infrastructure (PKI) that supports worldwide secure communication between parties that don't necessarily know each other prior to the communication. Asymmetric algorithms provide convenient key exchange mechanisms and are scalable to very large numbers of users, both challenges for users of symmetric cryptosystems.

This chapter also explores several practical applications of asymmetric cryptography: securing email, web communications, electronic commerce, digital rights management, and networking. The chapter concludes with an examination of a variety of attacks malicious individuals might use to compromise weak cryptosystems.

Asymmetric Cryptography

The section “Modern Cryptography” in Chapter 6 introduced the basic principles behind both private (symmetric) and public (asymmetric) key cryptography. You learned that symmetric key cryptosystems require both communicating parties to have the same shared secret key, creating the problem of secure key distribution. You also learned that asymmetric cryptosystems avoid this hurdle by using pairs of public and private keys to facilitate secure communication without the overhead of complex key distribution systems. The security of these systems relies on the difficulty of reversing a one-way function.

In the following sections, we’ll explore the concepts of public key cryptography in greater detail and look at three of the more common public key cryptosystems in use today: Rivest–Shamir–Adleman (RSA), El Gamal, and the elliptic curve cryptography (ECC).

Public and Private Keys

Recall from Chapter 6 that *public key cryptosystems* rely on pairs of keys assigned to each user of the cryptosystem. Every user maintains both a public key and a private key. As the names imply, public key cryptosystem users make their public keys freely available to anyone with whom they want to communicate. The mere possession of the public key by third parties does not introduce any weaknesses into the cryptosystem. The private key, on the other hand, is reserved for the sole use of the individual who owns the keys. It is never shared with any other cryptosystem user.

Normal communication between public key cryptosystem users is quite straightforward. Figure 7.1 shows the general process.

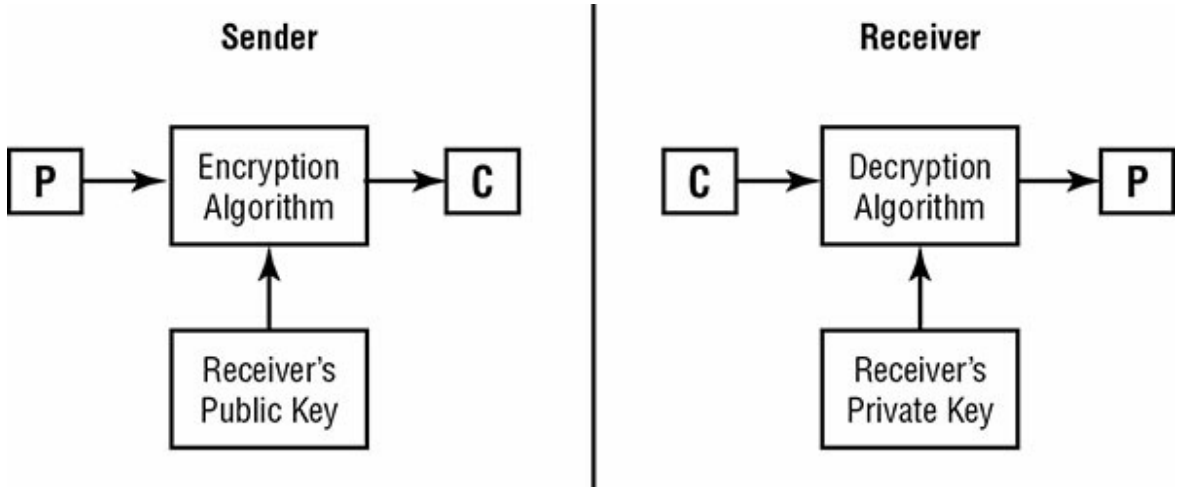

FIGURE 7.1 Asymmetric key cryptography

Notice that the process does not require the sharing of private keys. The sender encrypts the plaintext message (*P*) with the recipient's public key to create the ciphertext message (*C*). When the recipient opens the ciphertext message, they decrypt it using their private key to re-create the original plaintext message.

Once the sender encrypts the message with the recipient's public key, no user (including the sender) can decrypt that message without knowing the recipient's private key (the second half of the public-private key pair used to generate the message). This is the beauty of public key cryptography—public keys can be freely shared using unsecured communications and then used to create secure communications channels between users previously unknown to each other.

You also learned in the previous chapter that public key cryptography entails a higher degree of computational complexity. Keys used within public key systems must be longer than those used in private key systems to produce cryptosystems of equivalent strengths.

RSA

The most famous public key cryptosystem is named after its creators. In 1977, Ronald Rivest, Adi Shamir, and Leonard Adleman proposed the *RSA public key algorithm* that remains a worldwide standard today. They patented their algorithm and formed a commercial

venture known as RSA Security to develop mainstream implementations of their security technology. Today, the RSA algorithm has been released into the public domain and is widely used for secure communication.

The RSA algorithm depends on the computational difficulty inherent in factoring large prime numbers. Each user of the cryptosystem generates a pair of public and private keys using the algorithm described in the following steps:

1. Choose two large prime numbers (approximately 200 digits each), labeled p and q.
2. Compute the product of those two numbers: $n = p * q$.
3. Select a number, e, that satisfies the following two requirements:
   a. e is less than n.
   b. e and $(p - 1)(q - 1)$ are relatively prime—that is, the two numbers have no common factors other than 1.
4. Find a number, d, such that $(ed - 1) \bmod (p - 1)(q - 1) = 1$.
5. Distribute e and n as the public key to all cryptosystem users. Keep d secret as the private key.

If Alice wants to send an encrypted message to Bob, she generates the ciphertext (C) from the plain text (P) using the following formula (where e is Bob's public key and n is the product of p and q created during the key generation process):

$$C = P^e \bmod n$$

When Bob receives the message, he performs the following calculation to retrieve the plaintext message:

$$P = C^d \bmod n$$

Merkle-Hellman Knapsack

Another early asymmetric algorithm, the Merkle-Hellman Knapsack algorithm, was developed the year after RSA was

publicized. Like RSA, it's based on the difficulty of performing factoring operations, but it relies on a component of set theory known as *super-increasing sets* rather than on large prime numbers. Merkle-Hellman was proven ineffective when it was broken in 1984.

Importance of Key Length

The length of the cryptographic key is perhaps the most important security parameter that can be set at the discretion of the security administrator. It's important to understand the capabilities of your encryption algorithm and choose a key length that provides an appropriate level of protection. This judgment can be made by weighing the difficulty of defeating a given key length (measured in the amount of processing time required to defeat the cryptosystem) against the importance of the data.

Generally speaking, the more critical your data, the stronger the key you use to protect it should be. Timeliness of the data is also an important consideration. You must take into account the rapid growth of computing power—Moore's law suggests that computing power doubles approximately every two years. If it takes current computers one year of processing time to break your code, it will take only three months if the attempt is made with contemporary technology about four years down the road. If you expect that your data will still be sensitive at that time, you should choose a much longer cryptographic key that will remain secure well into the future.

Also, as attackers are now able to leverage cloud computing resources, they are able to more efficiently attack encrypted data. The cloud allows attackers to rent scalable computing power, including powerful graphic processing units (GPUs) on a per-hour basis, and offers significant discounts when using excess capacity during nonpeak hours. This brings powerful computing well within the reach of many attackers.

The strengths of various key lengths also vary greatly according to the cryptosystem you're using. The key lengths shown in the following table for three asymmetric cryptosystems all provide equal protection:

Cryptosystem	Key length
RSA	1,024 bits
DSA	1,024 bits
Elliptic curve	160 bits

El Gamal

In Chapter 6, you learned how the Diffie–Hellman algorithm uses large integers and modular arithmetic to facilitate the secure exchange of secret keys over insecure communications channels. In 1985, Dr. T. El Gamal published an article describing how the mathematical principles behind the Diffie–Hellman key exchange algorithm could be extended to support an entire public key cryptosystem used for encrypting and decrypting messages.

At the time of its release, one of the major advantages of El Gamal over the RSA algorithm was that it was released into the public domain. Dr. El Gamal did not obtain a patent on his extension of Diffie-Hellman, and it is freely available for use, unlike the then-patented RSA technology. (RSA released its algorithm into the public domain in 2000.)

However, El Gamal also has a major disadvantage—the algorithm doubles the length of any message it encrypts. This presents a major hardship when encrypting long messages or data that will be transmitted over a narrow bandwidth communications circuit.

Elliptic Curve

Also in 1985, two mathematicians, Neal Koblitz from the University of Washington and Victor Miller from IBM, independently proposed the application of *elliptic curve cryptography* (ECC) theory to develop

secure cryptographic systems.

The mathematical concepts behind elliptic curve cryptography are quite complex and well beyond the scope of this book. However, you should be generally familiar with the elliptic curve algorithm and its potential applications when preparing for the CISSP exam. If you are interested in learning the detailed mathematics behind elliptic curve cryptosystems, an excellent tutorial exists at https://www.certicom.com/content/certicom/en/ecc-tutorial.html.

Any elliptic curve can be defined by the following equation:

$$y^2 = x^3 + ax + b$$

In this equation, *x*, *y*, *a*, and *b* are all real numbers. Each elliptic curve has a corresponding *elliptic curve group* made up of the points on the elliptic curve along with the point *O*, located at infinity. Two points within the same elliptic curve group (*P* and *Q*) can be added together with an elliptic curve addition algorithm. This operation is expressed, quite simply, as follows:

$$P + Q$$

This problem can be extended to involve multiplication by assuming that *Q* is a multiple of *P*, meaning the following:

$$Q = xP$$

Computer scientists and mathematicians believe that it is extremely hard to find *x*, even if *P* and *Q* are already known. This difficult problem, known as the elliptic curve discrete logarithm problem, forms the basis of elliptic curve cryptography. It is widely believed that this problem is harder to solve than both the prime factorization problem that the RSA cryptosystem is based on and the standard discrete logarithm problem utilized by Diffie–Hellman and El Gamal. This is illustrated by the data shown in the table in the sidebar "Importance of Key Length," which noted that a 1,024-bit RSA key is

cryptographically equivalent to a 160-bit elliptic curve cryptosystem key.

Hash Functions

Later in this chapter, you'll learn how cryptosystems implement digital signatures to provide proof that a message originated from a particular user of the cryptosystem and to ensure that the message was not modified while in transit between the two parties. Before you can completely understand that concept, we must first explain the concept of *hash functions*. We will explore the basics of hash functions and look at several common hash functions used in modern digital signature algorithms.

Hash functions have a very simple purpose—they take a potentially long message and generate a unique output value derived from the content of the message. This value is commonly referred to as the *message digest*. Message digests can be generated by the sender of a message and transmitted to the recipient along with the full message for two reasons.

First, the recipient can use the same hash function to recompute the message digest from the full message. They can then compare the computed message digest to the transmitted one to ensure that the message sent by the originator is the same one received by the recipient. If the message digests do not match, that means the message was somehow modified while in transit. It is important to note that the messages must be *exactly* identical for the digests to match. If the messages have even a slight difference in spacing, punctuation, or content, the message digest values will be completely different. It is not possible to tell the degree of difference between two messages by comparing the digests. Even a slight difference will generate totally different digest values.

Second, the message digest can be used to implement a digital signature algorithm. This concept is covered in "Digital Signatures" later in this chapter.

The term *message digest* is used interchangeably with a

wide variety of synonyms, including *hash*, *hash value*, *hash total*, *CRC*, *fingerprint*, *checksum*, and *digital ID*.

In most cases, a message digest is 128 bits or larger. However, a single-digit value can be used to perform the function of parity, a low-level or single-digit checksum value used to provide a single individual point of verification. In most cases, the longer the message digest, the more reliable its verification of integrity.

According to RSA Security, there are five basic requirements for a cryptographic hash function:

- The input can be of any length.
- The output has a fixed length.
- The hash function is relatively easy to compute for any input.
- The hash function is one-way (meaning that it is extremely hard to determine the input when provided with the output). One-way functions and their usefulness in cryptography are described in Chapter 6.
- The hash function is collision free (meaning that it is extremely hard to find two messages that produce the same hash value).

In the following sections, we'll look at four common hashing algorithms: secure hash algorithm (SHA), message digest 2 (MD2), message digest 4 (MD4), and message digest 5 (MD5). Hash message authentication code (HMAC) is also discussed later in this chapter.

 .There are numerous hashing algorithms not addressed in this exam. But in addition to SHA, MD2, MD4, MD5, and HMAC, you should recognize HAVAL. Hash of Variable Length (HAVAL) is a modification of MD5. HAVAL uses 1,024-bit blocks and produces hash values of 128, 160, 192, 224, and 256 bits.

SHA

The Secure Hash Algorithm (SHA) and its successors, SHA-1, SHA-2, and SHA-3, are government standard hash functions promoted by the National Institute of Standards and Technology (NIST) and are specified in an official government publication—the Secure Hash Standard (SHS), also known as Federal Information Processing Standard (FIPS) 180.

SHA-1 takes an input of virtually any length (in reality, there is an upper bound of approximately 2,097,152 terabytes on the algorithm) and produces a 160-bit message digest. The SHA-1 algorithm processes a message in 512-bit blocks. Therefore, if the message length is not a multiple of 512, the SHA algorithm pads the message with additional data until the length reaches the next highest multiple of 512.

Cryptanalytic attacks demonstrated that there are weaknesses in the SHA-1 algorithm. This led to the creation of SHA-2, which has four variants:

- SHA-256 produces a 256-bit message digest using a 512-bit block size.
- SHA-224 uses a truncated version of the SHA-256 hash to produce a 224-bit message digest using a 512-bit block size.
- SHA-512 produces a 512-bit message digest using a 1,024-bit block size.
- SHA-384 uses a truncated version of the SHA-512 hash to produce a 384-bit digest using a 1,024-bit block size.

Although it might seem trivial, you should take the time to memorize the size of the message digests produced by each one of the hash algorithms described in this chapter.

The cryptographic community generally considers the SHA-2 algorithms secure, but they theoretically suffer from the same weakness as the SHA-1 algorithm. In 2015, the federal government

announced the release of the Keccak algorithm as the SHA-3 standard. The SHA-3 suite was developed to serve as drop-in replacement for the SHA-2 hash functions, offering the same variants and hash lengths using a more secure algorithm.

MD2

The Message Digest 2 (MD2) hash algorithm was developed by Ronald Rivest (the same Rivest of Rivest, Shamir, and Adleman fame) in 1989 to provide a secure hash function for 8-bit processors. MD2 pads the message so that its length is a multiple of 16 bytes. It then computes a 16-byte checksum and appends it to the end of the message. A 128-bit message digest is then generated by using the entire original message along with the appended checksum.

Cryptanalytic attacks exist against the MD2 algorithm. Specifically, Nathalie Rogier and Pascal Chauvaud discovered that if the checksum is not appended to the message before digest computation, collisions may occur. Frederic Mueller later proved that MD2 is not a one-way function. Therefore, it should no longer be used.

MD4

In 1990, Rivest enhanced his message digest algorithm to support 32-bit processors and increase the level of security. This enhanced algorithm is known as MD4. It first pads the message to ensure that the message length is 64 bits smaller than a multiple of 512 bits. For example, a 16-bit message would be padded with 432 additional bits of data to make it 448 bits, which is 64 bits smaller than a 512-bit message.

The MD4 algorithm then processes 512-bit blocks of the message in three rounds of computation. The final output is a 128-bit message digest.

The MD2, MD4, and MD5 algorithms are no longer accepted as suitable hashing functions. However, the details of the algorithms may still appear on the CISSP exam because they may

still be found in use today.

Several mathematicians have published papers documenting flaws in the full version of MD4 as well as improperly implemented versions of MD4. In particular, Hans Dobbertin published a paper in 1996 outlining how a modern personal computer could be used to find collisions for MD4 message digests in less than one minute. For this reason, MD4 is no longer considered to be a secure hashing algorithm, and its use should be avoided if at all possible.

MD5

In 1991, Rivest released the next version of his message digest algorithm, which he called MD5. It also processes 512-bit blocks of the message, but it uses four distinct rounds of computation to produce a digest of the same length as the MD2 and MD4 algorithms (128 bits). MD5 has the same padding requirements as MD4—the message length must be 64 bits less than a multiple of 512 bits.

MD5 implements additional security features that reduce the speed of message digest production significantly. Unfortunately, recent cryptanalytic attacks demonstrated that the MD5 protocol is subject to collisions, preventing its use for ensuring message integrity. Specifically, Arjen Lenstra and others demonstrated in 2005 that it is possible to create two digital certificates from different public keys that have the same MD5 hash.

Table 7.1 lists well-known hashing algorithms and their resultant hash value lengths in bits. Earmark this page for memorization.

TABLE 7.1 Hash algorithm memorization chart

Name	Hash value length
Hash of Variable Length (HAVAL)—an MD5 variant	128, 160, 192, 224, and 256 bits
Hash Message Authenticating Code (HMAC)	Variable
Message Digest 2 (MD2)	128

Message Digest 4 (MD4)	128
Message Digest 5 (MD5)	128
Secure Hash Algorithm (SHA-1)	160
SHA2-224/SHA3-224	224
SHA2-256/SHA3-256	256
SHA2-384/SHA3-384	384
SHA2-512/SHA3-512	512

Digital Signatures

Once you have chosen a cryptographically sound hashing algorithm, you can use it to implement a *digital signature* system. Digital signature infrastructures have two distinct goals:

- Digitally signed messages assure the recipient that the message truly came from the claimed sender. They enforce nonrepudiation (that is, they preclude the sender from later claiming that the message is a forgery).
- Digitally signed messages assure the recipient that the message was not altered while in transit between the sender and recipient. This protects against both malicious modification (a third party altering the meaning of the message) and unintentional modification (because of faults in the communications process, such as electrical interference).

Digital signature algorithms rely on a combination of the two major concepts already covered in this chapter—public key cryptography and hashing functions.

If Alice wants to digitally sign a message she's sending to Bob, she performs the following actions:

1. Alice generates a message digest of the original plaintext message using one of the cryptographically sound hashing algorithms, such as SHA3-512.
2. Alice then encrypts only the message digest using her private key. This encrypted message digest is the digital signature.
3. Alice appends the signed message digest to the plaintext message.
4. Alice transmits the appended message to Bob.

When Bob receives the digitally signed message, he reverses the procedure, as follows:

1. Bob decrypts the digital signature using Alice's public key.
2. Bob uses the same hashing function to create a message digest of

the full plaintext message received from Alice.

3. Bob then compares the decrypted message digest he received from Alice with the message digest he computed himself. If the two digests match, he can be assured that the message he received was sent by Alice. If they do not match, either the message was not sent by Alice or the message was modified while in transit.

Digital signatures are used for more than just messages. Software vendors often use digital signature technology to authenticate code distributions that you download from the internet, such as applets and software patches.

Note that the digital signature process does not provide any privacy in and of itself. It only ensures that the cryptographic goals of integrity, authentication, and nonrepudiation are met. However, if Alice wanted to ensure the privacy of her message to Bob, she could add a step to the message creation process. After appending the signed message digest to the plaintext message, Alice could encrypt the entire message with Bob's public key. When Bob received the message, he would decrypt it with his own private key before following the steps just outlined.

HMAC

The hashed message authentication code (HMAC) algorithm implements a partial digital signature—it guarantees the integrity of a message during transmission, but it does not provide for nonrepudiation.

Which Key Should I Use?

If you're new to public key cryptography, selecting the correct key for various applications can be quite confusing. Encryption, decryption, message signing, and signature verification all use the same algorithm with different key inputs. Here are a few simple

rules to help keep these concepts straight in your mind when preparing for the CISSP exam:

- If you want to encrypt a message, use the recipient's public key.
- If you want to decrypt a message sent to you, use your private key.
- If you want to digitally sign a message you are sending to someone else, use your private key.
- If you want to verify the signature on a message sent by someone else, use the sender's public key.

These four rules are the core principles of public key cryptography and digital signatures. If you understand each of them, you're off to a great start!

HMAC can be combined with any standard message digest generation algorithm, such as SHA-3, by using a shared secret key. Therefore, only communicating parties who know the key can generate or verify the digital signature. If the recipient decrypts the message digest but cannot successfully compare it to a message digest generated from the plaintext message, that means the message was altered in transit.

Because HMAC relies on a shared secret key, it does not provide any nonrepudiation functionality (as previously mentioned). However, it operates in a more efficient manner than the digital signature standard described in the following section and may be suitable for applications in which symmetric key cryptography is appropriate. In short, it represents a halfway point between unencrypted use of a message digest algorithm and computationally expensive digital signature algorithms based on public key cryptography.

Digital Signature Standard

The National Institute of Standards and Technology specifies the digital signature algorithms acceptable for federal government use in Federal Information Processing Standard (FIPS) 186-4, also known as the Digital Signature Standard (DSS). This document specifies that all federally approved digital signature algorithms must use the SHA-3

hashing functions.

DSS also specifies the encryption algorithms that can be used to support a digital signature infrastructure. There are three currently approved standard encryption algorithms:

- The Digital Signature Algorithm (DSA) as specified in FIPS 186-4
- The Rivest–Shamir–Adleman (RSA) algorithm as specified in ANSI X9.31
- The Elliptic Curve DSA (ECDSA) as specified in ANSI X9.62

.Two other digital signature algorithms you should recognize, at least by name, are Schnorr's signature algorithm and Nyberg-Rueppel's signature algorithm.

Public Key Infrastructure

The major strength of public key encryption is its ability to facilitate communication between parties previously unknown to each other. This is made possible by the *public key infrastructure (PKI)* hierarchy of trust relationships. These trusts permit combining asymmetric cryptography with symmetric cryptography along with hashing and digital certificates, giving us hybrid cryptography.

In the following sections, you'll learn the basic components of the public key infrastructure and the cryptographic concepts that make global secure communications possible. You'll learn the composition of a digital certificate, the role of certificate authorities, and the process used to generate and destroy certificates.

Certificates

Digital *certificates* provide communicating parties with the assurance that the people they are communicating with truly are who they claim to be. Digital certificates are essentially endorsed copies of an individual's public key. When users verify that a certificate was signed by a trusted certificate authority (CA), they know that the public key is legitimate.

Digital certificates contain specific identifying information, and their construction is governed by an international standard—X.509. Certificates that conform to X.509 contain the following data:

- Version of X.509 to which the certificate conforms
- Serial number (from the certificate creator)
- Signature algorithm identifier (specifies the technique used by the certificate authority to digitally sign the contents of the certificate)
- Issuer name (identification of the certificate authority that issued the certificate)
- Validity period (specifies the dates and times—a starting date and time and an ending date and time—during which the certificate is

valid)

- Subject's name (contains the distinguished name, or DN, of the entity that owns the public key contained in the certificate)
- Subject's public key (the meat of the certificate—the actual public key the certificate owner used to set up secure communications)

The current version of X.509 (version 3) supports certificate extensions—customized variables containing data inserted into the certificate by the certificate authority to support tracking of certificates or various applications.

If you're interested in building your own X.509 certificates or just want to explore the inner workings of the public key infrastructure, you can purchase the complete official X.509 standard from the International Telecommunications Union (ITU). It's part of the Open Systems Interconnection (OSI) series of communication standards and can be purchased electronically on the ITU website at www.itu.int.

Certificate Authorities

Certificate authorities (CAs) are the glue that binds the public key infrastructure together. These neutral organizations offer notarization services for digital certificates. To obtain a digital certificate from a reputable CA, you must prove your identity to the satisfaction of the CA. The following list includes some of the major CAs that provide widely accepted digital certificates:

- Symantec
- IdenTrust
- Amazon Web Services
- GlobalSign
- Comodo
- Certum

- GoDaddy
- DigiCert
- Secom
- Entrust
- Actalis
- Trustwave

Nothing is preventing any organization from simply setting up shop as a CA. However, the certificates issued by a CA are only as good as the trust placed in the CA that issued them. This is an important item to consider when receiving a digital certificate from a third party. If you don't recognize and trust the name of the CA that issued the certificate, you shouldn't place any trust in the certificate at all. PKI relies on a hierarchy of trust relationships. If you configure your browser to trust a CA, it will automatically trust all of the digital certificates issued by that CA. Browser developers preconfigure browsers to trust the major CAs to avoid placing this burden on users.

Registration authorities (RAs) assist CAs with the burden of verifying users' identities prior to issuing digital certificates. They do not directly issue certificates themselves, but they play an important role in the certification process, allowing CAs to remotely validate user identities.

Real World Scenario

Certificate Path Validation

You may have heard of *certificate path validation* (CPV) in your studies of certificate authorities. CPV means that each certificate in a certificate path from the original start or root of trust down to the server or client in question is valid and legitimate. CPV can be important if you need to verify that every link between "trusted" endpoints remains current, valid, and trustworthy.

This issue arises from time to time when intermediary systems' certificates expire or are replaced; this can break the chain of trust

or the verification path. By forcing a reverification of all stages of trust, you can reestablish all trust links and prove that the assumed trust remains assured.

Certificate Generation and Destruction

The technical concepts behind the public key infrastructure are relatively simple. In the following sections, we'll cover the processes used by certificate authorities to create, validate, and revoke client certificates.

Enrollment

When you want to obtain a digital certificate, you must first prove your identity to the CA in some manner; this process is called *enrollment*. As mentioned in the previous section, this sometimes involves physically appearing before an agent of the certification authority with the appropriate identification documents. Some certificate authorities provide other means of verification, including the use of credit report data and identity verification by trusted community leaders.

Once you've satisfied the certificate authority regarding your identity, you provide them with your public key. The CA next creates an X.509 digital certificate containing your identifying information and a copy of your public key. The CA then digitally signs the certificate using the CA's private key and provides you with a copy of your signed digital certificate. You may then safely distribute this certificate to anyone with whom you want to communicate securely.

Verification

When you receive a digital certificate from someone with whom you want to communicate, you *verify* the certificate by checking the CA's digital signature using the CA's public key. Next, you must check and ensure that the certificate was not revoked using a *certificate revocation list* (CRL) or the *Online Certificate Status Protocol (OCSP)*. At this point, you may assume that the public key listed in the certificate is authentic, provided that it satisfies the following requirements:

- The digital signature of the CA is authentic.
- You trust the CA.
- The certificate is not listed on a CRL.
- The certificate actually contains the data you are trusting.

The last point is a subtle but extremely important item. Before you trust an identifying piece of information about someone, be sure that it is actually contained within the certificate. If a certificate contains the email address (billjones@foo.com) but not the individual's name, you can be certain only that the public key contained therein is associated with that email address. The CA is not making any assertions about the actual identity of the billjones@foo.com email account. However, if the certificate contains the name Bill Jones along with an address and telephone number, the CA is vouching for that information as well.

Digital certificate verification algorithms are built in to a number of popular web browsing and email clients, so you won't often need to get involved in the particulars of the process. However, it's important to have a solid understanding of the technical details taking place behind the scenes to make appropriate security judgments for your organization. It's also the reason that, when purchasing a certificate, you choose a CA that is widely trusted. If a CA is not included in, or is later pulled from, the list of CAs trusted by a major browser, it will greatly limit the usefulness of your certificate.

In 2017, a significant security failure occurred in the digital certificate industry. Symantec, through a series of affiliated companies, issued several digital certificates that did not meet industry security standards. In response, Google announced that the Chrome browser would no longer trust Symantec certificates. As a result, Symantec wound up selling off its certificate-issuing business to DigiCert, which agreed to properly validate certificates prior to issuance. This demonstrates the importance of properly validating certificate requests. A series of seemingly small lapses in procedure can decimate a CA's business!

Revocation

Occasionally, a certificate authority needs to *revoke* a certificate. This might occur for one of the following reasons:

- The certificate was compromised (for example, the certificate owner accidentally gave away the private key).
- The certificate was erroneously issued (for example, the CA mistakenly issued a certificate without proper verification).
- The details of the certificate changed (for example, the subject's name changed).
- The security association changed (for example, the subject is no longer employed by the organization sponsoring the certificate).

The revocation request grace period is the maximum response time within which a CA will perform any requested revocation. This is defined in the *Certificate Practice Statement* (CPS). The CPS states the practices a CA employs when issuing or managing certificates.

You can use two techniques to verify the authenticity of certificates and identify revoked certificates:

Certificate Revocation Lists Certificate revocation lists (CRLs) are maintained by the various certificate authorities and contain the serial numbers of certificates that have been issued by a CA and have been revoked along with the date and time the revocation went into effect. The major disadvantage to certificate revocation lists is that they must be downloaded and cross-referenced periodically, introducing a period of latency between the time a certificate is revoked and the time end users are notified of the revocation. However, CRLs remain the most common method of checking certificate status in use today.

Online Certificate Status Protocol (OCSP) This protocol eliminates the latency inherent in the use of certificate revocation lists by providing a means for real-time certificate verification. When a client receives a certificate, it sends an OCSP request to the CA's OCSP

server. The server then responds with a status of valid, invalid, or unknown.

Asymmetric Key Management

When working within the public key infrastructure, it's important that you comply with several best practice requirements to maintain the security of your communications.

First, choose your encryption system wisely. As you learned earlier, "security through obscurity" is not an appropriate approach. Choose an encryption system with an algorithm in the public domain that has been thoroughly vetted by industry experts. Be wary of systems that use a "black-box" approach and maintain that the secrecy of their algorithm is critical to the integrity of the cryptosystem.

You must also select your keys in an appropriate manner. Use a key length that balances your security requirements with performance considerations. Also, ensure that your key is truly random. Any patterns within the key increase the likelihood that an attacker will be able to break your encryption and degrade the security of your cryptosystem.

When using public key encryption, keep your private key secret! Do not, under any circumstances, allow anyone else to gain access to your private key. Remember, allowing someone access even once permanently compromises all communications that take place (past, present, or future) using that key and allows the third party to successfully impersonate you.

Retire keys when they've served a useful life. Many organizations have mandatory key rotation requirements to protect against undetected key compromise. If you don't have a formal policy that you must follow, select an appropriate interval based on the frequency with which you use your key. You might want to change your key pair every few months, if practical.

Back up your key! If you lose the file containing your private key because of data corruption, disaster, or other circumstances, you'll certainly want to have a backup available. You may want to either create your own backup or use a key escrow service that maintains the backup for you. In either case, ensure that the backup is handled in a

secure manner. After all, it's just as important as your primary key file!

Hardware security modules (HSMs) also provide an effective way to manage encryption keys. These hardware devices store and manage encryption keys in a secure manner that prevents humans from ever needing to work directly with the keys. HSMs range in scope and complexity from very simple devices, such as the YubiKey, that store encrypted keys on a USB drive for personal use to more complex enterprise products that reside in a data center. Cloud providers, such as Amazon and Microsoft, also offer cloud-based HSMs that provide secure key management for IaaS services.

Applied Cryptography

Up to this point, you've learned a great deal about the foundations of cryptography, the inner workings of various cryptographic algorithms, and the use of the public key infrastructure to distribute identity credentials using digital certificates. You should now feel comfortable with the basics of cryptography and be prepared to move on to higher-level applications of this technology to solve everyday communications problems.

In the following sections, we'll examine the use of cryptography to secure data at rest, such as that stored on portable devices, as well as data in transit, using techniques that include secure email, encrypted web communications, and networking.

Portable Devices

The now ubiquitous nature of notebook computers, netbooks, smartphones, and tablets brings new risks to the world of computing. Those devices often contain highly sensitive information that, if lost or stolen, could cause serious harm to an organization and its customers, employees, and affiliates. For this reason, many organizations turn to encryption to protect the data on these devices in the event they are misplaced.

Current versions of popular operating systems now include disk encryption capabilities that make it easy to apply and manage encryption on portable devices. For example, Microsoft Windows includes the BitLocker and Encrypting File System (EFS) technologies, Mac OS X includes FileVault encryption, and the VeraCrypt open-source package allows the encryption of disks on Linux, Windows, and Mac systems.

Trusted Platform Module

Modern computers often include a specialized cryptographic component known as a Trusted Platform Module (TPM). The TPM

is a chip that resides on the motherboard of the device. The TPM serves a number of purposes, including the storage and management of keys used for full disk encryption (FDE) solutions. The TPM provides the operating system with access to the keys, preventing someone from removing the drive from one device and inserting it into another device to access the drive's data.

A wide variety of commercial tools are available that provide added features and management capability. The major differentiators between these tools are how they protect keys stored in memory, whether they provide full disk or volume-only encryption, and whether they integrate with hardware-based Trusted Platform Modules (TPMs) to provide added security. Any effort to select encryption software should include an analysis of how well the alternatives compete on these characteristics.

Don't forget about smartphones when developing your portable device encryption policy. Most major smartphone and tablet platforms include enterprise-level functionality that supports encryption of data stored on the phone.

Email

We have mentioned several times that security should be cost effective. When it comes to email, simplicity is the most cost-effective option, but sometimes cryptography functions provide specific security services that you can't avoid using. Since ensuring security is also cost effective, here are some simple rules about encrypting email:

- If you need confidentiality when sending an email message, encrypt the message.
- If your message must maintain integrity, you must hash the message.
- If your message needs authentication, integrity and/or nonrepudiation, you should digitally sign the message.

- If your message requires confidentiality, integrity, authentication, and nonrepudiation, you should encrypt and digitally sign the message.

It is always the responsibility of the sender to put proper mechanisms in place to ensure that the security (that is, confidentiality, integrity, authenticity, and nonrepudiation) of a message or transmission is maintained.

One of the most in-demand applications of cryptography is encrypting and signing email messages. Until recently, encrypted email required the use of complex, awkward software that in turn required manual intervention and complicated key exchange procedures. An increased emphasis on security in recent years resulted in the implementation of strong encryption technology in mainstream email packages. Next, we'll look at some of the secure email standards in widespread use today.

Pretty Good Privacy

Phil Zimmerman's Pretty Good Privacy (PGP) secure email system appeared on the computer security scene in 1991. It combines the CA hierarchy described earlier in this chapter with the "web of trust" concept—that is, you must become trusted by one or more PGP users to begin using the system. You then accept their judgment regarding the validity of additional users and, by extension, trust a multilevel "web" of users descending from your initial trust judgments.

PGP initially encountered a number of hurdles to widespread use. The most difficult obstruction was the U.S. government export regulations, which treated encryption technology as munitions and prohibited the distribution of strong encryption technology outside the United States. Fortunately, this restriction has since been repealed, and PGP may be freely distributed to most countries.

PGP is available in two versions. The commercial version uses RSA for key exchange, IDEA for encryption/decryption, and MD5 for message digest production. The freeware version (based on the extremely similar OpenPGP standard) uses Diffie-Hellman key exchange, the Carlisle Adams/Stafford Tavares (CAST) 128-bit

encryption/decryption algorithm, and the SHA-1 hashing function.

Many commercial providers also offer PGP-based email services as web-based cloud email offerings, mobile device applications, or webmail plug-ins. These services appeal to administrators and end users because they remove the complexity of configuring and maintaining encryption certificates and provide users with a managed secure email service. Some products in this category include StartMail, Mailvelope, SafeGmail, and Hushmail.

S/MIME

The Secure/Multipurpose Internet Mail Extensions (S/MIME) protocol has emerged as a de facto standard for encrypted email. S/MIME uses the RSA encryption algorithm and has received the backing of major industry players, including RSA Security. S/MIME has already been incorporated in a large number of commercial products, including these:

- Microsoft Outlook and Office 365
- Mozilla Thunderbird
- Mac OS X Mail
- GSuite Enterprise edition

S/MIME relies on the use of X.509 certificates for exchanging cryptographic keys. The public keys contained in these certificates are used for digital signatures and for the exchange of symmetric keys used for longer communications sessions. RSA is the only public key cryptographic protocol supported by S/MIME. The protocol supports the AES and 3DES symmetric encryption algorithms.

Despite strong industry support for the S/MIME standard, technical limitations have prevented its widespread adoption. Although major desktop mail applications support S/MIME email, mainstream web-based email systems do not support it out of the box (the use of browser extensions is required).

Web Applications

Encryption is widely used to protect web transactions. This is mainly because of the strong movement toward e-commerce and the desire of both e-commerce vendors and consumers to securely exchange financial information (such as credit card information) over the web. We'll look at the two technologies that are responsible for the small lock icon within web browsers—Secure Sockets Layer (SSL) and Transport Layer Security (TLS).

SSL was developed by Netscape to provide client/server encryption for web traffic. Hypertext Transfer Protocol Secure (HTTPS) uses port 443 to negotiate encrypted communications sessions between web servers and browser clients. Although SSL originated as a standard for Netscape browsers, Microsoft also adopted it as a security standard for its popular Internet Explorer browser. The incorporation of SSL into both of these products made it the de facto internet standard.

SSL relies on the exchange of server digital certificates to negotiate encryption/decryption parameters between the browser and the web server. SSL's goal is to create secure communications channels that remain open for an entire web browsing session. It depends on a combination of symmetric and asymmetric cryptography. The following steps are involved:

1. When a user accesses a website, the browser retrieves the web server's certificate and extracts the server's public key from it.
2. The browser then creates a random symmetric key, uses the server's public key to encrypt it, and then sends the encrypted symmetric key to the server.
3. The server then decrypts the symmetric key using its own private key, and the two systems exchange all future messages using the symmetric encryption key.

This approach allows SSL to leverage the advanced functionality of asymmetric cryptography while encrypting and decrypting the vast majority of the data exchanged using the faster symmetric algorithm.

In 1999, security engineers proposed TLS as a replacement for the SSL standard, which was at the time in its third version. As with SSL, TLS uses TCP port 443. Based on SSL technology, TLS incorporated many

security enhancements and was eventually adopted as a replacement for SSL in most applications. Early versions of TLS supported downgrading communications to SSL v3.0 when both parties did not support TLS. However, in 2011, TLS v1.2 dropped this backward compatibility.

In 2014, an attack known as the Padding Oracle On Downgraded Legacy Encryption (POODLE) demonstrated a significant flaw in the SSL 3.0 fallback mechanism of TLS. In an effort to remediate this vulnerability, many organizations completely dropped SSL support and now rely solely on TLS security.

Even though TLS has been in existence for more than a decade, many people still mistakenly call it SSL. For this reason, TLS has gained the nickname SSL 3.1.

Steganography and Watermarking

Steganography is the art of using cryptographic techniques to embed secret messages within another message. Steganographic algorithms work by making alterations to the least significant bits of the many bits that make up image files. The changes are so minor that there is no appreciable effect on the viewed image. This technique allows communicating parties to hide messages in plain sight—for example, they might embed a secret message within an illustration on an otherwise innocent web page.

Steganographers often embed their secret messages within images or WAV files because these files are often so large that the secret message would easily be missed by even the most observant inspector. Steganography techniques are often used for illegal or questionable activities, such as espionage and child pornography.

Steganography can also be used for legitimate purposes, however. Adding digital watermarks to documents to protect intellectual property is accomplished by means of steganography. The hidden information is known only to the file's creator. If someone later creates an unauthorized copy of the content, the watermark can be used to

detect the copy and (if uniquely watermarked files are provided to each original recipient) trace the offending copy back to the source.

Steganography is an extremely simple technology to use, with free tools openly available on the internet. Figure 7.2 shows the entire interface of one such tool, iSteg. It simply requires that you specify a text file containing your secret message and an image file that you wish to use to hide the message. Figure 7.3 shows an example of a picture with an embedded secret message; the message is impossible to detect with the human eye.

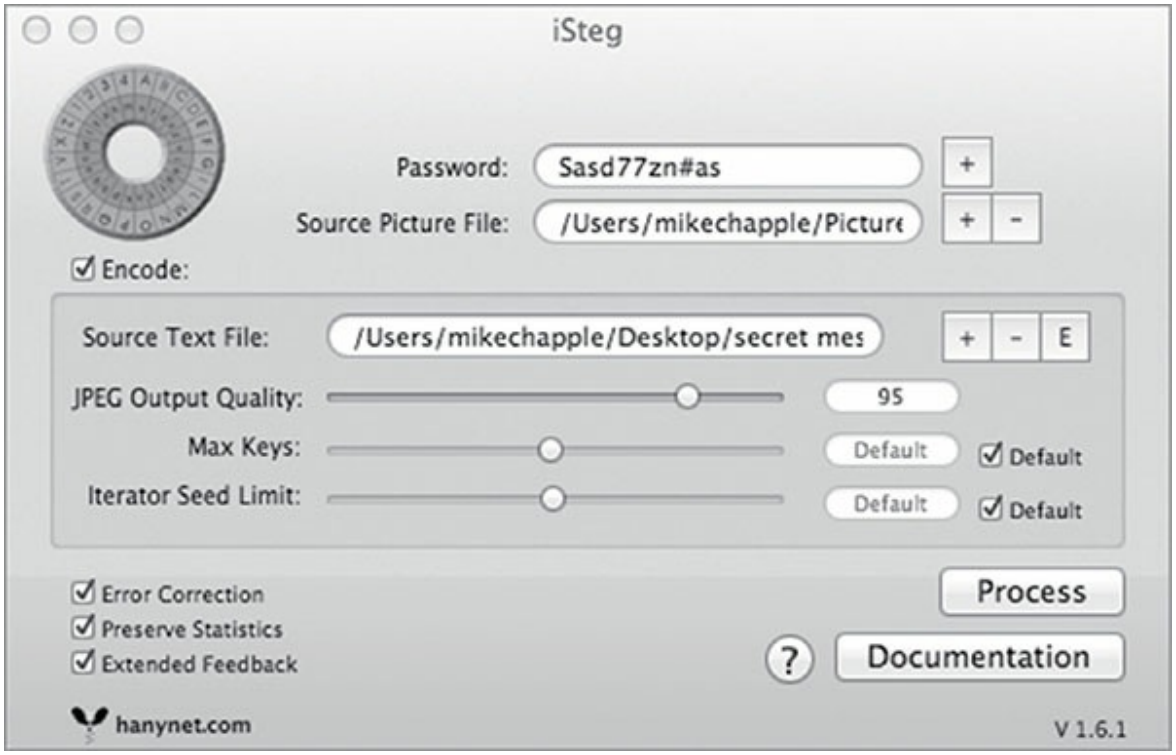

FIGURE 7.2 Steganography tool

FIGURE 7.3 Image with embedded message

Digital Rights Management

Digital rights management (DRM) software uses encryption to enforce copyright restrictions on digital media. Over the past decade, publishers attempted to deploy DRM schemes across a variety of media types, including music, movies, and books. In many cases, particularly with music, opponents met DRM deployment attempts with fierce opposition, arguing that the use of DRM violated their rights to freely enjoy and make backup copies of legitimately licensed media files.

As you will read in this section, many commercial attempts to deploy DRM on a widespread basis failed when users rejected the technology as intrusive and/or obstructive.

Music DRM

The music industry has battled pirates for years, dating back to the days of homemade cassette tape duplication and carrying through compact disc and digital formats. Music distribution companies attempted to use a variety of DRM schemes, but most backed away from the technology under pressure from consumers.

The use of DRM for purchased music slowed dramatically when, facing this opposition, Apple rolled back their use of FairPlay DRM for music sold through the iTunes Store. Apple co-founder Steve Jobs foreshadowed this move when, in 2007, he issued an open letter to the music industry calling on them to allow Apple to sell DRM-free music. That letter read, in part:

> The third alternative is to abolish DRMs entirely. Imagine a world where every online store sells DRM-free music encoded in open licensable formats. In such a world, any player can play music purchased from any store, and any store can sell music which is playable on all players. This is clearly the best alternative for consumers, and Apple would embrace it in a heartbeat. If the big four music companies would license Apple their music without the requirement that it be protected with a DRM, we would switch to selling only DRM-free music on our iTunes store. Every iPod ever made will play this DRM-free music.

The full essay is no longer available on Apple's website, but an archived copy may be found at http://bit.ly/1TyBm5e.

Currently, the major use of DRM technology in music is for subscription-based services such as Napster and Kazaa, which use DRM to revoke a user's access to downloaded music when their subscription period ends.

Do the descriptions of DRM technology in this section seem a little vague? There's a reason for that: manufacturers typically do not disclose the details of their DRM functionality due to fears that pirates will use that information to defeat the DRM

scheme.

Movie DRM

The movie industry has used a variety of DRM schemes over the years to stem the worldwide problem of movie piracy. Two of the major technologies used to protect mass-distributed media are as follows:

High-Bandwidth Digital Content Protection (HDCP) Provides DRM protection for content sent over digital connections including HDMI, DisplayPort, and DVI interfaces. While this technology is still found in many implementations, hackers released an HDCP master key in 2010, rendering the protection completely ineffective.

Advanced Access Content System (AACS) Protects the content stored on Blu-Ray and HD DVD media. Hackers have demonstrated attacks that retrieved AACS encryption keys and posted them on the internet.

Industry publishers and hackers continue the cat-and-mouse game today; media companies try to protect their content and hackers seek to gain continued access to unencrypted copies.

E-book DRM

Perhaps the most successful deployment of DRM technology is in the area of book and document publishing. Most e-books made available today use some form of DRM, and these technologies also protect sensitive documents produced by corporations with DRM capabilities.


All DRM schemes in use today share a fatal flaw: the device used to access the content must have access to the decryption key. If the decryption key is stored on a device possessed by the end user, there is always a chance that the user will manipulate the device to gain access to the key.

Adobe Systems offers the Adobe Digital Experience Protection Technology (ADEPT) to provide DRM technology for e-books sold in a

variety of formats. ADEPT uses a combination of AES technology to encrypt the media content and RSA encryption to protect the AES key. Many e-book readers, with the notable exception of the Amazon Kindle, use this technology to protect their content. Amazon's Kindle e-readers use a variety of formats for book distribution, and each contains its own encryption technology.

Video Game DRM

Many video games implement DRM technology that depends on consoles using an active internet connection to verify the game license with a cloud-based service. These technologies, such as Ubisoft's Uplay, once typically required a constant internet connection to facilitate gameplay. If a player lost connection, the game would cease functioning.

In March 2010, the Uplay system came under a denial-of-service attack and players of Uplay-enabled games around the world were unable to play games that previously functioned properly because their consoles were unable to access the Uplay servers. This led to public outcry, and Ubisoft later removed the always-on requirement, shifting to a DRM approach that only requires an initial activation of the game on the console and then allows unrestricted use.

Document DRM

Although the most common uses of DRM technology protect entertainment content, organizations may also use DRM to protect the security of sensitive information stored in PDF files, office productivity documents, and other formats. Commercial DRM products, such as Vitrium and FileOpen, use encryption to protect source content and then enable organizations to carefully control document rights.

Here are some of the common permissions restricted by document DRM solutions:

- Reading a file
- Modifying the contents of a file
- Removing watermarks from a file

- Downloading/saving a file
- Printing a file
- Taking screenshots of file content

DRM solutions allow organizations to control these rights by granting them when needed, revoking them when no longer necessary, and even automatically expiring rights after a specified period of time.

Networking

The final application of cryptography we'll explore in this chapter is the use of cryptographic algorithms to provide secure networking services. In the following sections, we'll take a brief look at two methods used to secure communications circuits. We'll also look at IPsec and Internet Security Association and Key Management Protocol (ISAKMP) as well as some of the security issues surrounding wireless networking.

Circuit Encryption

Security administrators use two types of encryption techniques to protect data traveling over networks:

- *Link encryption* protects entire communications circuits by creating a secure tunnel between two points using either a hardware solution or a software solution that encrypts all traffic entering one end of the tunnel and decrypts all traffic entering the other end of the tunnel. For example, a company with two offices connected via a data circuit might use link encryption to protect against attackers monitoring at a point in between the two offices.
- *End-to-end encryption* protects communications between two parties (for example, a client and a server) and is performed independently of link encryption. An example of end-to-end encryption would be the use of TLS to protect communications between a user and a web server. This protects against an intruder who might be monitoring traffic on the secure side of an encrypted link or traffic sent over an unencrypted link.

The critical difference between link and end-to-end encryption is that

in link encryption, all the data, including the header, trailer, address, and routing data, is also encrypted. Therefore, each packet has to be decrypted at each hop so it can be properly routed to the next hop and then re-encrypted before it can be sent along its way, which slows the routing. End-to-end encryption does not encrypt the header, trailer, address, and routing data, so it moves faster from point to point but is more susceptible to sniffers and eavesdroppers.

When encryption happens at the higher OSI layers, it is usually end-to-end encryption, and if encryption is done at the lower layers of the OSI model, it is usually link encryption.

Secure Shell (SSH) is a good example of an end-to-end encryption technique. This suite of programs provides encrypted alternatives to common internet applications such as File Transfer Protocol (FTP), Telnet, and rlogin. There are actually two versions of SSH. SSH1 (which is now considered insecure) supports the Data Encryption Standard (DES), Triple DES (3DES), and International Data Encryption Algorithm (IDEA), and Blowfish algorithms. SSH2 drops support for DES and IDEA but adds support for several other algorithms.

IPsec

Various security architectures are in use today, each one designed to address security issues in different environments. One such architecture that supports secure communications is the Internet Protocol Security (IPsec) standard. IPsec is a standard architecture set forth by the Internet Engineering Task Force (IETF) for setting up a secure channel to exchange information between two entities.

The entities communicating via IPsec could be two systems, two routers, two gateways, or any combination of entities. Although generally used to connect two networks, IPsec can be used to connect individual computers, such as a server and a workstation or a pair of workstations (sender and receiver, perhaps). IPsec does not dictate all implementation details but is an open, modular framework that allows many manufacturers and software developers to develop IPsec solutions that work well with products from other vendors.

IPsec uses public key cryptography to provide encryption, access control, nonrepudiation, and message authentication, all using IP-based protocols. The primary use of IPsec is for virtual private networks (VPNs), so IPsec can operate in either transport or tunnel mode. IPsec is commonly paired with the Layer 2 Tunneling Protocol (L2TP) as L2TP/IPsec.

The IP Security (IPsec) protocol provides a complete infrastructure for secured network communications. IPsec has gained widespread acceptance and is now offered in a number of commercial operating systems out of the box. IPsec relies on security associations, and there are two main components:

- The Authentication Header (AH) provides assurances of message integrity and nonrepudiation. AH also provides authentication and access control and prevents replay attacks.
- The Encapsulating Security Payload (ESP) provides confidentiality and integrity of packet contents. It provides encryption and limited authentication and prevents replay attacks.

ESP also provides some limited authentication, but not to the degree of the AH. Though ESP is sometimes used without AH, it's rare to see AH used without ESP.

IPsec provides for two discrete modes of operation. When IPsec is used in *transport mode*, only the packet payload is encrypted. This mode is designed for peer-to-peer communication. When it's used in *tunnel mode*, the entire packet, including the header, is encrypted. This mode is designed for gateway-to-gateway communication.

IPsec is an extremely important concept in modern computer security. Be certain that you're familiar with the component protocols and modes of IPsec operation.

At runtime, you set up an IPsec session by creating a *security association* (SA). The SA represents the communication session and records any configuration and status information about the connection. The SA represents a simplex connection. If you want a two-way channel, you need two SAs, one for each direction. Also, if you want to support a bidirectional channel using both AH and ESP, you will need to set up four SAs.

Some of IPsec's greatest strengths come from being able to filter or manage communications on a per-SA basis so that clients or gateways between which security associations exist can be rigorously managed in terms of what kinds of protocols or services can use an IPsec connection. Also, without a valid security association defined, pairs of users or gateways cannot establish IPsec links.

Further details of the IPsec algorithm are provided in Chapter 11, "Secure Network Architecture and Securing Network Components."

ISAKMP

The Internet Security Association and Key Management Protocol (ISAKMP) provides background security support services for IPsec by negotiating, establishing, modifying, and deleting security associations. As you learned in the previous section, IPsec relies on a system of security associations (SAs). These SAs are managed through the use of ISAKMP. There are four basic requirements for ISAKMP, as set forth in Internet RFC 2408:

- Authenticate communicating peers
- Create and manage security associations
- Provide key generation mechanisms
- Protect against threats (for example, replay and denial-of-service attacks)

Wireless Networking

The widespread rapid adoption of wireless networks poses a tremendous security risk. Many traditional networks do not implement encryption for routine communications between hosts on

the local network and rely on the assumption that it would be too difficult for an attacker to gain physical access to the network wire inside a secure location to eavesdrop on the network. However, wireless networks transmit data through the air, leaving them extremely vulnerable to interception. There are two main types of wireless security:

Wired Equivalent Privacy Wired Equivalent Privacy (WEP) provides 64- and 128-bit encryption options to protect communications within the wireless LAN. WEP is described in IEEE 802.11 as an optional component of the wireless networking standard.

Cryptanalysis has conclusively demonstrated that significant flaws exist in the WEP algorithm, making it possible to completely undermine the security of a WEP-protected network within seconds. You should never use WEP encryption to protect a wireless network. In fact, the use of WEP encryption on a store network was the root cause behind the TJX security breach that was widely publicized in 2007. Again, you should *never* use WEP encryption on a wireless network.

WiFi Protected Access WiFi Protected Access (WPA) improves on WEP encryption by implementing the Temporal Key Integrity Protocol (TKIP), eliminating the cryptographic weaknesses that undermined WEP. A further improvement to the technique, dubbed WPA2, adds AES cryptography. WPA2 provides secure algorithms appropriate for use on modern wireless networks.

Remember that WPA does not provide an end-to-end security solution. It encrypts traffic only between a mobile computer and the nearest wireless access point. Once the traffic hits the wired network, it's in the clear again.

Another commonly used security standard, IEEE 802.1x, provides a flexible framework for authentication and key management in wired

and wireless networks. To use 802.1x, the client runs a piece of software known as the *supplicant*. The supplicant communicates with the authentication server. After successful authentication, the network switch or wireless access point allows the client to access the network. WPA was designed to interact with 802.1x authentication servers.

Cryptographic Attacks

As with any security mechanism, malicious individuals have found a number of attacks to defeat cryptosystems. It's important that you understand the threats posed by various cryptographic attacks to minimize the risks posed to your systems:

Analytic Attack This is an algebraic manipulation that attempts to reduce the complexity of the algorithm. Analytic attacks focus on the logic of the algorithm itself.

Implementation Attack This is a type of attack that exploits weaknesses in the implementation of a cryptography system. It focuses on exploiting the software code, not just errors and flaws but the methodology employed to program the encryption system.

Statistical Attack A statistical attack exploits statistical weaknesses in a cryptosystem, such as floating-point errors and inability to produce truly random numbers. Statistical attacks attempt to find a vulnerability in the hardware or operating system hosting the cryptography application.

Brute Force Brute-force attacks are quite straightforward. Such an attack attempts every possible valid combination for a key or password. They involve using massive amounts of processing power to methodically guess the key used to secure cryptographic communications.

For a nonflawed protocol, the average amount of time required to discover the key through a brute-force attack is directly proportional to the length of the key. A brute-force attack will always be successful given enough time. Every additional bit of key length doubles the time to perform a brute-force attack because the number of potential keys doubles.

There are two modifications that attackers can make to enhance the effectiveness of a brute-force attack:

- Rainbow tables provide precomputed values for cryptographic hashes. These are commonly used for cracking passwords stored

on a system in hashed form.

- Specialized, scalable computing hardware designed specifically for the conduct of brute-force attacks may greatly increase the efficiency of this approach.

Salting Saves Passwords

Salt might be hazardous to your health, but it can save your password! To help combat the use of brute-force attacks, including those aided by dictionaries and rainbow tables, cryptographers make use of a technology known as *cryptographic salt*.

The cryptographic salt is a random value that is added to the end of the password before the operating system hashes the password. The salt is then stored in the password file along with the hash. When the operating system wishes to compare a user's proffered password to the password file, it first retrieves the salt and appends it to the password. It feeds the concatenated value to the hash function and compares the resulting hash with the one stored in the password file.

Specialized password hashing functions, such as PBKDF2, bcrypt, and scrypt, allow for the creation of hashes using salts and also incorporate a technique known as *key stretching* that makes it more computationally difficult to perform a single password guess.

The use of salting, especially when combined with key stretching, dramatically increases the difficulty of brute-force attacks. Anyone attempting to build a rainbow table must build a separate table for each possible value of the cryptographic salt.

Frequency Analysis and the Ciphertext Only Attack In many cases, the only information you have at your disposal is the encrypted ciphertext message, a scenario known as the *ciphertext only attack*. In this case, one technique that proves helpful against simple ciphers is frequency analysis—counting the number of times each letter appears in the ciphertext. Using your knowledge that the letters *E*, *T*, *A*, *O*, *I*, *N*

are the most common in the English language, you can then test several hypotheses:

- If these letters are also the most common in the ciphertext, the cipher was likely a transposition cipher, which rearranged the characters of the plain text without altering them.
- If other letters are the most common in the ciphertext, the cipher is probably some form of substitution cipher that replaced the plaintext characters.

This is a simple overview of frequency analysis, and many sophisticated variations on this technique can be used against polyalphabetic ciphers and other sophisticated cryptosystems.

Known Plaintext In the known plaintext attack, the attacker has a copy of the encrypted message along with the plaintext message used to generate the ciphertext (the copy). This knowledge greatly assists the attacker in breaking weaker codes. For example, imagine the ease with which you could break the Caesar cipher described in Chapter 6 if you had both a plaintext copy and a ciphertext copy of the same message.

Chosen Ciphertext In a chosen ciphertext attack, the attacker has the ability to decrypt chosen portions of the ciphertext message and use the decrypted portion of the message to discover the key.

Chosen Plaintext In a chosen plaintext attack, the attacker has the ability to encrypt plaintext messages of their choosing and can then analyze the ciphertext output of the encryption algorithm.

Meet in the Middle Attackers might use a meet-in-the-middle attack to defeat encryption algorithms that use two rounds of encryption. This attack is the reason that Double DES (2DES) was quickly discarded as a viable enhancement to the DES encryption (it was replaced by Triple DES, or 3DES).

In the meet-in-the-middle attack, the attacker uses a known plaintext message. The plain text is then encrypted using every possible key (k1), and the equivalent ciphertext is decrypted using all possible keys (k2). When a match is found, the corresponding pair (k1, k2) represents both portions of the double encryption. This type of attack

generally takes only double the time necessary to break a single round of encryption (or 2^n rather than the anticipated $2^n * 2^n$), offering minimal added protection.

Man in the Middle In the man-in-the-middle attack, a malicious individual sits between two communicating parties and intercepts all communications (including the setup of the cryptographic session). The attacker responds to the originator's initialization requests and sets up a secure session with the originator. The attacker then establishes a second secure session with the intended recipient using a different key and posing as the originator. The attacker can then "sit in the middle" of the communication and read all traffic as it passes between the two parties.

Be careful not to confuse the meet-in-the-middle attack with the man-in-the-middle attack. They may have similar names, but they are quite different!

Birthday The birthday attack, also known as a *collision attack* or *reverse hash matching* (see the discussion of brute-force and dictionary attacks in Chapter 14, "Controlling and Monitoring Access"), seeks to find flaws in the one-to-one nature of hashing functions. In this attack, the malicious individual seeks to substitute in a digitally signed communication a different message that produces the same message digest, thereby maintaining the validity of the original digital signature.

Don't forget that social engineering techniques can also be used in cryptanalysis. If you're able to obtain a decryption key by simply asking the sender for it, that's much easier than attempting to crack the cryptosystem!

Replay The replay attack is used against cryptographic algorithms that don't incorporate temporal protections. In this attack, the malicious individual intercepts an encrypted message between two

parties (often a request for authentication) and then later “replays” the captured message to open a new session. This attack can be defeated by incorporating a time stamp and expiration period into each message.

Summary

Asymmetric key cryptography, or public key encryption, provides an extremely flexible infrastructure, facilitating simple, secure communication between parties that do not necessarily know each other prior to initiating the communication. It also provides the framework for the digital signing of messages to ensure nonrepudiation and message integrity.

This chapter explored public key encryption, which provides a scalable cryptographic architecture for use by large numbers of users. We also described some popular cryptographic algorithms, such as link encryption and end-to-end encryption. Finally, we introduced you to the public key infrastructure, which uses certificate authorities (CAs) to generate digital certificates containing the public keys of system users and digital signatures, which rely on a combination of public key cryptography and hashing functions.

We also looked at some of the common applications of cryptographic technology in solving everyday problems. You learned how cryptography can be used to secure email (using PGP and S/MIME), web communications (using SSL and TLS), and both peer-to-peer and gateway-to-gateway networking (using IPsec and ISAKMP) as well as wireless communications (using WPA and WPA2).

Finally, we covered some of the more common attacks used by malicious individuals attempting to interfere with or intercept encrypted communications between two parties. Such attacks include birthday, cryptanalytic, replay, brute-force, known plaintext, chosen plaintext, chosen ciphertext, meet-in-the-middle, man-in-the-middle, and birthday attacks. It’s important for you to understand these attacks in order to provide adequate security against them.