

PART II

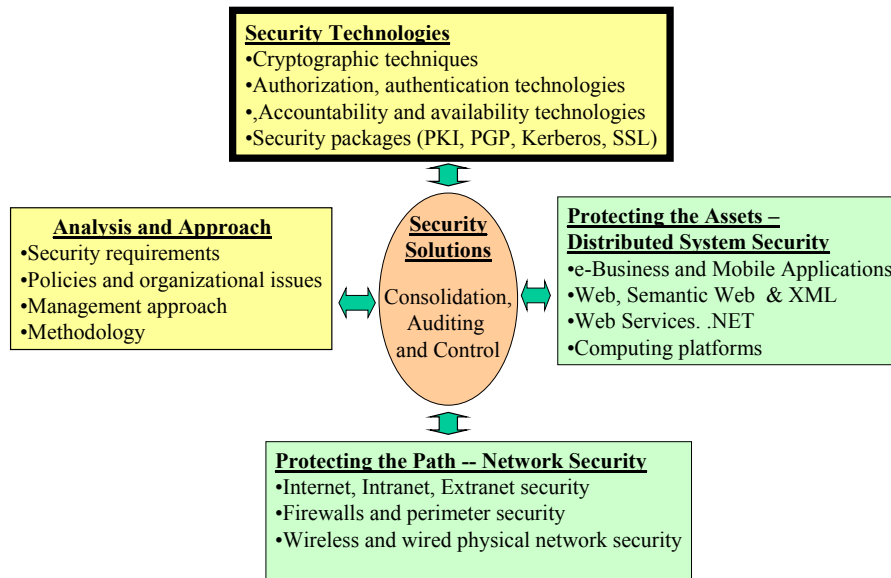
Security Technologies and Packages

This part of the book concentrates on the security technologies and packages (box with dark borders in the framework shown below).

Chapter 4: Cryptography and Encryption

Chapter 5: Authorization, Authentication, Accountability, and Availability Technologies – The 4As

Chapter 6: Common Security Packages: PKI, VPN, SSL, PGP and Kerberos



4 Cryptography and Encryption

| | | |
|--------|---|------|
| 4.1 | CRYPTOGRAPHY OVERVIEW..... | 4-2 |
| 4.2 | SYMMETRIC KEY ENCRYPTION – THE CONVENTIONAL APPROACH | 4-5 |
| 4.2.1 | Overview..... | 4-5 |
| 4.2.2 | Modern Cryptography Foundations..... | 4-7 |
| 4.2.3 | Common Encryption Algorithms..... | 4-8 |
| 4.2.4 | Issues with Symmetric (Secret) Key Cryptography..... | 4-8 |
| 4.3 | PUBLIC KEY (ASYMMETRIC) CRYPTOGRAPHY | 4-9 |
| 4.3.1 | Overview..... | 4-9 |
| 4.3.2 | Asymmetric Key Cryptography – A Closer Look | 4-10 |
| 4.3.3 | Symmetric Versus Asymmetric Cryptography..... | 4-11 |
| 4.3.4 | Hybrid Encryption Systems | 4-11 |
| 4.4 | SECURITY KEY CONSIDERATIONS: PERFORMANCE AND PROTECTION..... | 4-13 |
| 4.5 | DIGITAL SIGNATURES | 4-14 |
| 4.5.1 | Overview..... | 4-14 |
| 4.5.2 | A Closer Look..... | 4-15 |
| 4.5.3 | e-Signature Law..... | 4-15 |
| 4.6 | MESSAGE DIGESTS FOR INFORMATION INTEGRITY | 4-16 |
| 4.7 | DIGITAL ENVELOPES – COMBINING SYMMETRIC AND ASYMMETRIC KEY SYSTEMS..... | 4-17 |
| 4.8 | MAN IN THE MIDDLE – SECURITY WEAKNESS OF PUBLIC KEY SYSTEMS..... | 4-18 |
| 4.9 | EXAMPLES OF ENCRYPTION SYSTEMS | 4-19 |
| 4.10 | CONCLUDING COMMENTS: HOW SYMMETRIC AND ASYMMETRIC SYSTEMS ARE REALLY BEING USED IN PRACTICE | 4-20 |
| 4.11 | SHORT CASE STUDIES AND EXAMPLES | 4-20 |
| 4.11.1 | Legal Issues Concerning Publication of Encryption Algorithms | 4-20 |
| 4.11.2 | Digital Signatures Support Mobile Home Banking..... | 4-21 |
| 4.11.3 | Security in Health Care | 4-22 |
| 4.12 | SUGGESTED REVIEW QUESTIONS | 4-23 |

4.1 Cryptography Overview

Cryptography has been used for a number of years to mask messages so that interveners cannot see or modify the messages. In fact, cryptography is a Greek word that means “secret message” and was used by Julius Caesar to mask messages he sent to his generals. Cryptography over the years has become a science of using mathematics to encrypt and decrypt data. It enables you to store sensitive information in files or transmit it across a network so that it cannot be read by anyone except the intended recipient.

The main ingredients of cryptography, as illustrated in Figure 4-1, are:

- **Plaintext**, also known as clear text, is the original message or data that the sender wants to send.
- **Encryption algorithm**, or **cipher**, performs various transformations and substitutions on the plaintext, i.e., scrambles the plaintext. These algorithms are mathematical functions used in the encryption and decryption process.
- **Key**, used by the encryption algorithm to scramble (encrypt) the plaintext. The exact transformations and substitutions depend on the key. A key can be a word, number, or a phrase. The same plaintext encrypts to different ciphertext with different keys.
- **Ciphertext** is the scrambled message produced as an output. This message depends on the encryption algorithm and the key.
- **Decryption algorithm** unscrambles the ciphertext and is effectively a reverse of the encryption algorithm. The decryption algorithm may use the same or a different key for decryption. As we will see later, this is the difference between symmetric and asymmetric encryption.

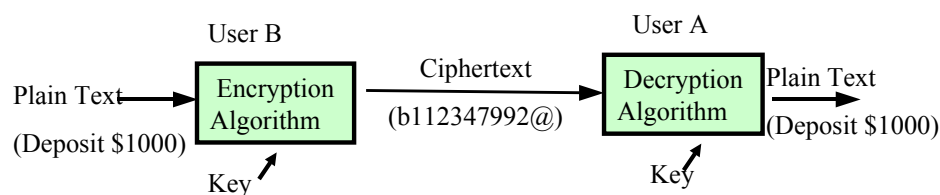


Figure 4-1: Cryptography Components

To illustrate the key ideas, let us look at the “Caesar Cipher” – an old but well known example of cryptography. The **Caesar Cipher** was used by Julius Caesar to send encrypted messages to his generals. The main idea was to replace each input letter with the third letter to the right as shown below:

a b c d e f g h i j k l m n o p q r s t (input – plaintext)
 d e f g h i j k l m n o p q r s t u v w (output – cipher)

In this case the key is 3 and the encryption algorithm is “shift right.” To decrypt, the receiver reverses it (i.e., shifts the ciphertext 3 letters to the left). For example, the plaintext “come back” would produce a ciphertext of “frph edfn.” No wonder the enemies of Caesar were so confused!

While cryptography is used to secure data, **cryptanalysis** is used to analyze and break secure communication. Cryptanalysts, also called attackers, use a combination of analytical reasoning, mathematical tools, pattern finding, patience, determination, and luck. Cryptology includes both cryptography and cryptanalysis.

Cryptography can be *strong* or *weak* depending on the time and resources needed to recover the plaintext from a ciphertext. *Strong cryptography* produces a ciphertext that is virtually impossible to decipher. The main idea is to make this task so difficult that given today’s computing power and available time, it is not possible to decipher the result of very strong cryptography for thousands of years. This does not mean that an extremely

determined cryptanalyst does not get lucky or just uses other means to access secret keys (greed and money still work!), In addition, newer computing systems with massively parallel processors could have some impact on difficulties in breaking strong cryptography. Basically, the security of encrypted data is entirely dependent on the strength of the cryptographic algorithm, length of the key, and the secrecy of the key.

Due to e-commerce, encryption/decryption has become a major area of active work. In addition, the events of September 2001 have heightened the need for security. In most cases, data is transformed into an encrypted message. The encrypted message is then transmitted and decrypted on the other side by using the *same* key. This type of cryptography, known as **Conventional** or **Symmetric Encryption**, is discussed below. Encryption/decryption can be performed by hardware and/or software. Modern computing systems have the ability to implement very sophisticated encryption/decryption techniques. The same encryption can be used on all data in a system or encryption keys can be more “personalized.” For example, instead of using the same encryption/decryption key on all data from all stations in a network, each user can have his or her own encryption/decryption key card, which is inserted into a workstation before the user logs on. This card encrypts the data before sending it across the network. The encrypted data can be read only by those users or programs with access to an appropriate key. Encryption techniques generally fall into two broad categories: symmetric key and asymmetric key.

Chapter Highlights

- Encryption includes symmetric and asymmetric key cryptography.
- Symmetric, also known as private key encryption, is simple and efficient:
 - It can be broken by brute force techniques.
 - Larger keys are harder to break by brute force.
 - If a private key is found, intruders can decrypt messages because the same key is used for encryption and decryption.
 - DES is a common symmetric key algorithm.
- Asymmetric key cryptography uses two keys, one private and the other public:
 - It's much safer because encryption and decryption keys are different
 - It's not very efficient – adds a great deal of overhead.
 - RSA is the most common asymmetric key algorithm.
- Consider the tradeoff between symmetric and asymmetric key encryption: safety versus efficiency
- Digital envelope is a good compromise between the two encryptions.
 - It uses symmetric key P for encrypting/decrypting message.
 - It uses asymmetric key for sending private key P.
- Message digests (hashes) are used to assure that the messages are not changed in transit.
- Digital signatures are used to authenticate a user, very much like a handwritten signature.
 - Uses asymmetric key algorithm.
 - There are many issues with widespread use of digital signatures.
- Encryption is used widely in systems such as PGP, SSL, SET, and others.

4.2 Symmetric Key Encryption – The Conventional Approach

4.2.1 Overview

In a symmetric key encryption scheme, the *same key* is used by the sender to encrypt the message, and by the receiver to decrypt it. For example, as shown in Figure 4-2, the same key (e) is used to encrypt as well as decrypt. To use this cryptographic system, you encrypt the plaintext by using receiver's secret key e . Thus when the receiver receives the ciphertext, it can be decrypted by using e . The main issue in this widely used encryption scheme is that the senders and receivers have to agree on a secret key e . The main problem, as we will see, is that this scheme does not scale well to a large number of users because all senders and receivers have to agree on secret keys before transmission. In addition, how secure can this key be if it has to be exchanged between many senders and receivers?

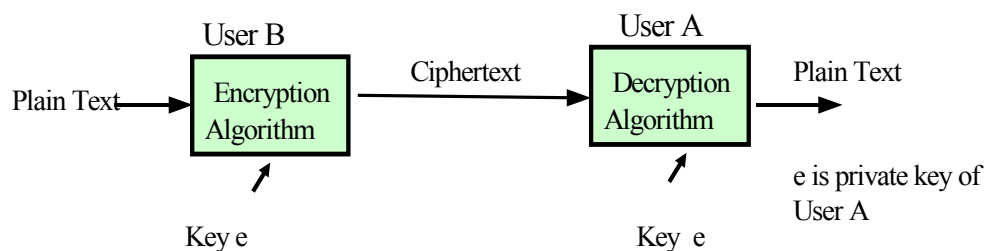


Figure 4-2: Symmetric (Secret) Key Encryption – the Conventional Approach

The idea of symmetric key encryption is very old – that is why it is known as conventional encryption. For example, the Caesar Cipher mentioned previously is an early example of symmetric key encryption. Many old classical approaches were variants of Caesar's cipher where:

- Key can be n instead of 3
- The shift can be left or right
- The alphabets can be replaced with numbers and special characters
- The shifts and the directions can change periodically (e.g., the first 10 characters are shifted by 3 letters to the right and the next 10 letters are shifted 6 letters to the left, etc.).

The main limitation of the old character-based cryptography is that these encryption schemes are not very strong – you can shift characters left and right just so long. You also cannot perform mathematical functions on characters (you cannot add or multiply alphabetic characters). Modern cryptographic techniques have changed all this by using translation of plaintext into bits instead of characters. The characters are translated into bits by computer systems by using ASCII or EBCDIC codes, so no extra effort is needed

here. Once everything is in bits, random numbers can be generated for keys and then “applied” to the plaintext bits by using a variety of mathematical functions.

Table 4-1: Average Time Estimates for a Hardware Brute Force Attack on Symmetric Keys with Different Lengths (based on 1995 Technologies)

| | 40-bit Key | 64-bit key | 80-bit key | 128-bit key |
|--------------------------------|-------------------|-------------------|-------------------|--------------------|
| Slowest Machine | 2 Seconds | 1 year | 70,000 years | 10^{19} years |
| Reasonably Fast Machine | 0.02 seconds | 5.4 minutes | 245 days | 10^{14} years |
| Fastest Machine | 0.02 microseconds | 0.3 seconds | 6 hours | 10^{11} years |

The main strength of a symmetric key is the key length. The longer the key, the harder it is to break in. For example, the numbers shown in Table 4-1 are based on a study published in *Applied Cryptography* by Bruce Schneier (the numbers are based on 1995 technology). Thus it is better to choose higher-bit encryption. However, in many cases, the key length chosen depends on the key lengths supported by the client and the server in a session (see the sidebar “Key Sizes in SSL Sessions – A Practical Issue”).

Key Sizes in SSL Sessions – A Practical Issue

SSL (Secure Socket layer) is a common protocol used in Web communications. For example, whenever you log on to a secure site, SSL is used between your Web browser and the Web server (indicated by https instead of http). The SSL client and server negotiate an encryption scheme and key sizes before establishing a session. In reality, SSL gives the users many cryptographic choices, known as “*cipher suites*.” Each cipher suite has a different security strength. For example, the cipher suite (DES-RSA-MD5) in SSL 3.0 represents a security option with very high strength and long key sizes. Each Web browser and server supports several cipher suites. When an SSL client connects to a server, they both negotiate a cipher suite that is strongest but available on both sides. A common problem is that international web sites have smaller key lengths (e.g., 40 bits). Thus the SSL session uses 40-bit keys even though higher key lengths are available on the Web browser. Consider, for example, the situation where your browser supports 40- and 80-bit encryption keys but you need to communicate with the following servers:

- A site in New York that supports 64-, 80- and 128-bit key encryption
- A site in London that supports 40-, 64-, and 80-bit key encryption
- A site in Malaysia that supports 40-bit key encryption

Then, your session with the New York and London sites will be encrypted by using an 80-bit key (the largest common key is 80-bit), but a 40-bit key will be used for Malaysia. Thus the security of your session depends on the largest key size both participants can agree on.

4.2.2 Modern Cryptography Foundations

The basic idea of modern cryptography is that all data (plaintext, cipher text) is in bits instead of alphanumeric characters. A random number K is generated to serve as a key. For encryption, complex functions are applied (i.e., added, subtracted, shifted, “orred,”¹ “exclusive orred”²) to the key and the message bits to produce the ciphertext (see Figure 4-3).

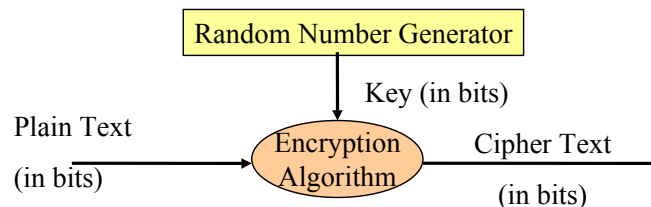


Figure 4-3: Modern Cryptography

Modern cryptography uses two types of cryptographic algorithms: stream ciphers that operate on continuous bits, and block ciphers that operate on blocks of data.

Stream ciphers accept a random number as a bit stream and perform operations between these bits and the bits of the plaintext. The operations can be shift rights, shift lefts, additions, or other bit operations. These ciphers are usually implemented in hardware devices.

Block ciphers are usually implemented in software and are much more popular than stream cryptos. This scheme operates on blocks of text – usually in multiples of 8 bits (character length). DES (Data Encryption Standard) developed by NSA (National Security Administration) is the best-known example of this type of encryption. The key, also known as a cipher variable, is 64 bits long. The encryption algorithm works as follows (see Figure 4-4):

- Plaintext is divided into 64-bit blocks (T_1, T_2, \dots). These blocks are processed in several iterations and produce corresponding ciphertext blocks (C_1, C_2, \dots).
- Each block is divided into 2 parts: L_0 and R_0 (left and right bits)
- The two halves are processed as follows for $I=0,1,\dots, 15$ by using a complex function f :

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} + f(R_{i-1}, K_i)$$

- Each block (T_1, T_2, \dots) is processed several times and new R is computed from L by using f to generate the ciphertext blocks C_1, C_2, \dots .



¹ Or operation is a binary arithmetic operation which indicates that the resulting bit is on if any of the input bits is on.

² Exclusive or is a binary arithmetic operation which indicates that the resulting bit is on if the input bits are dissimilar

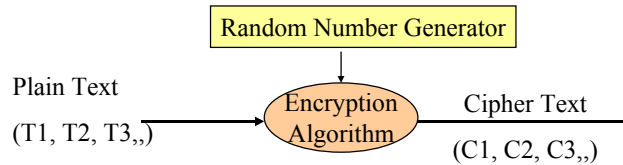


Figure 4-4: Block Crypto Algorithms

4.2.3 Common Encryption Algorithms

A variety of symmetric key encryption algorithms have been introduced over the years. **DES (Data Encryption Standard)** is one of the most common publicly available encryption algorithms. Initially developed by IBM in the 70s, DES was adopted by the US government as a national standard. Despite its popularity, DES has been controversial for quite a while because some people feel that it could be cracked by the National Security Administration (NSA) whenever needed [Andress 2002]. DES was initially designed for a 64-bit key, but NSA reduced it to 56 bits before making it a standard (this naturally raised more suspicions). Although DES is used heavily, it has been cracked a few times – so it is not flawless (see www.rsasecurity.com).

In several cases, DES is applied multiple times for stronger cryptography. TripleDES is an example. In this case, the message is encrypted and then decrypted by using three keys; DES is used three times with an effective key length of 168 bits. This makes it much safer than the simple 56-bit DES algorithm. Other variants of DES such as DESX, GDES, and RDEX have also been developed.

Additional examples of private key encryptions are RCx, Rijndael, IDEA (International Data Encryption Algorithm), and Blowfish. See Stallings [2001] for details of these and other algorithms.

4.2.4 Issues with Symmetric (Secret) Key Cryptography

Private key encryption is very fast and efficient. It is especially useful for large data transmissions such as file transfers. However, it suffers from several limitations, especially in key management. In other words, since the sender and receiver have to agree on the same key, sending the key from one side to the other might compromise it. Basically, if someone finds out the key, then he or she can decrypt the ciphertext. So how can the sender and receiver know about the key safely? Naturally, it is not a good idea to send the key plus the ciphertext in the same message – they should be separate messages. But how to send the key? The key could be sent over a secure channel. But if a secure channel exists, why use encryption?

Another problem with symmetric keys is that in large systems with thousands of users, it is difficult to assign unique keys. Symmetric key encryption is particularly difficult for businesses that deal with millions of one-time customers on a daily basis. Special techniques are needed to use this method of encryption for situations involving more than a modest number of permanent users.

One such technique is to generate the key by software or hardware on both sides, instead of exchanging it. A typical technology used in many corporations is the **secure card**. A

secure card is given to each employee. The card has a chip which generates a random number R every minute. The number generated is unique to the employee. To access a corporate system, the employee types in his/her employee ID plus R – this becomes the key K . On the receiving system, a similar routine generates R for the employee. Messages sent are encrypted by using the key K that the user types, and are decrypted by using the key K generated by the system. If these keys do not match, the message cannot be decrypted.

What do hackers do? Typical hacker practice consists of the following steps:

- Trap the messages between senders and receivers.
- Try to understand the messages. In many cases, the messages are not encrypted and are in clear text. This is hacker delight.
- If messages are encrypted, try to guess the key. Sophisticated un-scramblers can be devised that run forever and try to guess the keys by using different combinations.

The main idea of strong encryption is to make sure that the key must be very difficult to guess. For example, the secure ID does generate keys that are quite difficult to guess. Key lengths and sophisticated key processing improves security but adds significant overhead.

4.3 Public Key (Asymmetric) Cryptography

4.3.1 Overview

In a public key (asymmetric) system, the encryption key e and the decryption key d are different – hence the name “asymmetric” (see Figure 4-5). Each user has a pair of keys, a private key d that he keeps secret and a public key e that he publishes. When Pat needs to send a message to Joe, she encrypts the message with Joe’s public key e_j . This encrypted message can only be decrypted with Joe’s private key d_j . Therefore, if this encrypted message is delivered to a user (Sam) who does not know d_j , then Sam cannot decrypt it. Even when the message is broadcast over the Internet, only Joe can decrypt the message because he is the only user who knows d_j . The main point of public key systems is that the decryption key is completely private and is not transmitted over the network. While public key systems solve the problem of key management, they are usually significantly slower than private key systems. The RSA (Rivest, Shamir and Adleman) algorithm, developed in 1976, is by far the most widely used public key encryption algorithm.

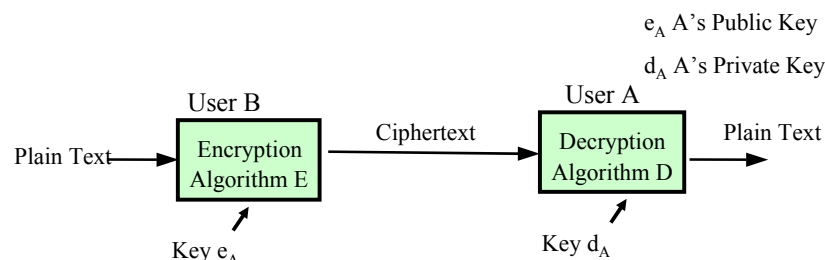


Figure 4-5: Public (Asymmetric) Key Encryption

4.3.2 Asymmetric Key Cryptography – A Closer Look

The concept of public key cryptography was introduced by Whitfield Diffie and Martin Hellman in 1975. However, there is now evidence that the British Secret Service invented this technique a few years before Diffie and Hellman [Ellis 1970]. The reason why this was not known by Diffie and Hellman is that this technique was kept a military secret with no thought towards commercialization.

In an asymmetric (public) key system every user has two keys: a public key e that everyone knows (it is used for encryption) and private key d that only the receiver knows (it is used for decryption). Even if someone knows the key used for encryption, this key cannot be used for decryption. This scheme is very suitable for the Internet because anyone can send a message, but only the intended user can decrypt and read the message. Given:

M: Message (plaintext)

E: algorithm uses public key (e) – everyone knows it (used for encryption)

D: algorithm uses private key (d) – only receiver knows (used for decryption)

The following relationships hold

1. $D(E(M)) = M$
2. $E(D(M)) = M$
3. Given E , it is not possible to determine D
4. Given D , it is not possible to determine E

Each person (A, B, \dots) has two keys (e_A, d_A), (e_B, d_B). When the sender B wants to send a message M to A , it first encrypts M by using e_A , and sends it to A . After receiving the encrypted message, A decrypts it by using d_B . This process uses principle 1. As we will see later, a digital signature uses principle 2.

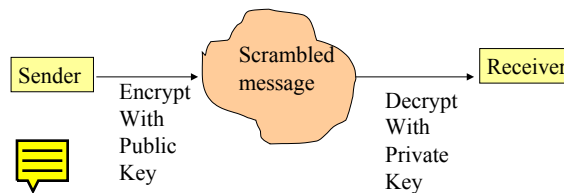


Figure 4-6; Public Key Infrastructure

The main idea is that you publish your public key to the world while keeping your private key secret (Figure 4-6). Anyone with a copy of your public key – even people you have never met – can then encrypt information that only you can read. It is computationally infeasible to deduce the private key from the public key. Anyone who has a public key can encrypt information but cannot decrypt it. Only the person who has the corresponding private key can decrypt the information. RSA (named for its inventors, Ron Rivest, Adi Shamir, and Leonard Adleman) is the best-known public key cryptosystem. Other examples of public-key cryptosystems are Elgamal (named for its

inventor, Taher Elgamal), Diffie-Hellman (named for its inventors), and DSA, the Digital Signature Algorithm (invented by David Kravitz).

4.3.3 Symmetric Versus Asymmetric Cryptography

Conventional symmetric encryption has several benefits. It is very fast and is especially useful for encrypting data that does not need to be sent over a network. Thus you can encrypt a sensitive file and only the people knowing the private key can read it. However, the private key must be shared between senders and receivers if the data is to be sent and shared by multiple users. As the population of users grows, some (dishonest) users can tell others about the private key or send the encryption card to an accomplice. Remotely located users must trust a courier or some other secure communication medium to prevent the disclosure of the secret key during transmission. Anyone who intercepts the key in transit can later read, modify, and forge all information encrypted or authenticated with that key. The persistent problem with conventional encryption is *key distribution*: how do you get the key to the recipient without someone intercepting it?

The primary benefit of public key cryptography is that it allows people who have no preexisting security arrangement to exchange messages securely. The need for sender and receiver to share secret keys via some secure channel is eliminated; all communications involve only public keys, and no private key is ever transmitted or shared. The need for expensive secure channels and key distribution is eliminated. These factors have limited private key systems to government and large banks. Public key encryption provides strong cryptography to the general public.

While public key systems solve the problem of key management, they are usually significantly slower than private key systems. The RSA (Rivest, Shamir and Adleman) algorithm, described briefly in the following sidebar, is between one hundred to one thousand times slower than DES [Stein 1998]. Due to this limitation, public key encryption is not typically used on the entire message. Instead, just the key is encrypted by using RSA. This compromise, known as a digital envelope, is discussed later in section 4.7.

4.3.4 Hybrid Encryption Systems

Hybrid encryption systems use a mixture of symmetric and asymmetric encryption. These systems take advantage of the security of asymmetric and the performance of symmetric key systems. These systems are especially useful for highly secure transmission of large amounts of data. In this case, asymmetric encryption is too slow, but symmetric encryption is not very secure due to the key transmission problem. Hybrid encryption systems use a public key only to encrypt the keys of a symmetric system, instead of the entire message. You start by using a secret key (called a “session key”) that is used to encrypt the message. But before sending the session key, you encrypt this key by using the asymmetric method. Thus, the key is transmitted safely along with the data, with minimum overhead. Digital envelopes, discussed later, are used in many security packages like SSL and are based on this approach.

RSA – A Very Brief Description

1. Start with two large prime numbers: p and q . The numbers are large enough so that they contain hundreds of digits. These two numbers must be prime numbers; i.e., they do not have divisors other than the number itself and 1. Examples of prime numbers are 2, 3, 7, 11, 13, 17, 19, etc. The number 2 is the only even prime number; all others are odd.

2. Define $n = p \cdot q$ and $t = (p-1) \cdot (q-1)$. Thus for $p = 3$ and $q = 7$, we get $n = 21$ and $t = 12$.

3. Now choose another number e that has no common divisor with t . We can choose 5.

4. Next choose d such that $e \cdot d = 1 \pmod{t}$. This means that $(e \cdot d - 1)$ is divisible by t with no remainder. We can choose $d = 5$ (d and e are usually different, but in this example, they are the same).

The numbers d , p , q are kept secret because they are used to build D . The numbers e and n are public and are used to build E .

5. To encrypt a message m , convert the message to a string of integers m_1, m_2, \dots , all smaller than n

7. Compute $c_i = E(m_i) = (m_i)^e \pmod{n}$.

This means that we raise m_i to the power e and then divide by n . The remainder is $c_i = E(m_i)$. In our example, for $m_1 = 9$, $c_i = (9)^5 \pmod{21}$. I.e., $c_i = 59049 \pmod{21}$. After carrying out the division, we get a remainder of 18, thus $c_i = 18$.

7. To decrypt, the decryption function D is defined by $D(c_i) = (c_i)^d \pmod{n}$.

This means that we raise c_i to the power d and then divide by n . The remainder is $D(c_i)$. In our example, $c_i = 18$ and $d = 5$. Thus we get $(18)^5 \pmod{21}$. I.e., $D(c_i) = 1889668 \pmod{21}$. After carrying out the division, we get a remainder of 9.

Thus we conclude that $D(18) = 9$, the message we started with.

To verify this process further, you should try the following combinations out:

a) Two prime numbers ($p=7$, $q=11$) and message text = 9. After calculating n and t , we can get $e=11$ and $d=11$. Working through E and D , we get, $c_i = 53$

b) Two prime numbers ($p=7$, $q=11$) and message text = 8. After calculating n and t , we can get $e=11$ and $d=11$. Working through E and D , we get, $c_i = 8$

c) Two prime numbers ($p=3$, $q=11$) and message text = 9. After calculating n and t , we can get $e=9$ and $d=9$. Working through E and D , we get, $c_i = 27$

4.4 Security Key Considerations: Performance and Protection

A key is a value that is used by a cryptographic algorithm to produce a ciphertext. In modern cryptography, keys are very large numbers, usually measured in bits. The larger the key (bits in the key), the more secure the ciphertext. However, the algorithms used for symmetric versus asymmetric cryptography are very different, and thus a key size of n bits in symmetric crypto is not equivalent to an n -bit key in asymmetric crypto. For example, a conventional 80-bit key has the equivalent strength of a 1024-bit public key.

Although the public and private keys of asymmetric systems are mathematically related, it is very difficult to derive the private key given only the public key; however, deriving the private key is always possible given enough time and computing power. Thus it is very important to pick keys of the right size so that they are large enough to be secure but small enough to be applied quickly. Larger keys are cryptographically more secure for a longer period of time because they are harder to guess. This does depend of course on the determination of attackers and the faster computer systems of the future. For a long time, a 56-bit symmetric key was considered extremely safe, but now larger key sizes (128-bit) are being preferred.

Keys are stored in encrypted form. Systems such as PGP (Pretty Good Privacy) store the keys in two files, called *keyrings*, on your hard disk. The public keyring is used to store the public keys of your recipients, and the private keyring is used to store your private keys. These keyrings are important – if you lose your private keyring, it is time to close your tent and go home because you cannot decrypt any information encrypted to keys on that ring.

Protection of the keys that in turn are used to protect the assets is extremely important. Private keys and shared secrets, once acquired, must be protected. End-to-end security must include consideration of the security of the end-user device. Private keys stored on a personal computer disk file may be stolen via access to the file system or outright theft of the device. Security can be enhanced by the use of smart cards. Another approach is to use a security chip embedded in end-user systems. In addition, server-side hardware devices can provide tamper-resistant key storage as well as assistance for encrypting and decrypting messages and public/private key operations that require a heavy computational load.

Sharing of private keys, although not a recommended practice, is necessary at times. For example, Corporate Signing Keys are private keys used by a company to sign legal documents or press releases. In such a case, it may be better for multiple members of the company to have access to the private key. It may be worthwhile to split the key among multiple people who must present a piece of the key in order to reconstitute it to a usable condition. We have all seen examples of such “*key splitting*” in movies where each person knows only part of a secret code. If a secure network connection is used during the reconstitution process, the partial key holders need not be physically present in order to rejoin the key.

4.5 Digital Signatures

4.5.1 Overview

A major appeal of public key cryptography is that it provides a method for employing *digital signatures*. A digital signature serves the same purpose as a handwritten (also known as “wet”) signature – it enables the recipient of information to verify the authenticity of the information’s origin, and also verify that the information is intact. Thus, public key digital signatures provide *authentication* and data *integrity*. A digital signature also provides *non-repudiation*; i.e., it prevents the sender from claiming that he or she did not actually send the information. These features are as important to cryptography as privacy.

Digital signature technology is used to authenticate the source of a message. It is essentially the same as a public key system except that the order in which the keys are applied is reversed. A sender “signs” the message by applying his private key to it. The sender sends the message and the signature to the receiver. The receiver checks the signature by applying the sender’s public key to it. If the receiver gets the original message back, he is sure that the message was signed by the sender’s private key, and therefore, was sent by the sender himself. In essence, a digital signature is a block of data created by applying a cryptographic signing algorithm to some data using the signer’s private key. Digital signatures may be used to authenticate the source of the message and to assure message recipients that no one has tampered with a message since the time it was sent by the signer.

Let us consider a scenario in which Joe wants to send his signature to Pat, as shown in Figure 4-7. Joe creates a short message (“I am Joe”) as a signature S and encrypts his signature with his private key (encrypts by using d_j). The created ciphertext $d_j(S)$ is sent to Pat. On arrival, Pat decrypts it by using Joe’s public key e_j . This means that the signature is Joe’s signature because the ciphertext can only be decrypted if sent from the right person. This uses Principle 2 as discussed in section 4.3.2, i.e., $E(D(M)) = M$. The main idea is that the receiver knows that the message could not be decrypted unless it was encrypted by D .

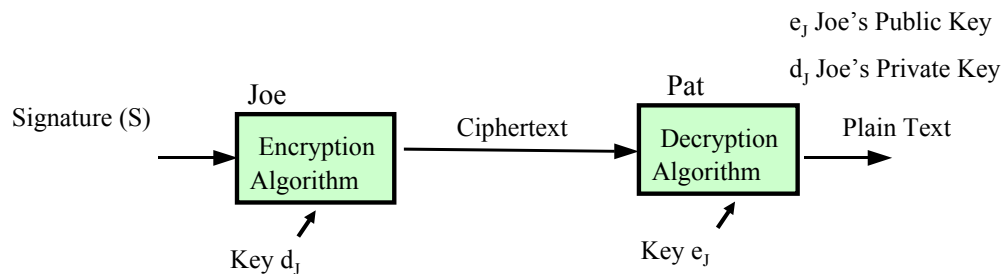


Figure 4-7: Digital Signature

4.5.2 A Closer Look

Let us now consider a more complicated, albeit more practical scenario. Let us assume that Joe's lawyer sends a document (Joe's will W , stating "Pat will inherit my estate") and asks Joe to sign it and send it to Pat. Let us assume that Joe has the private-public key pair (d_j, e_j) and Pat has the private-public key pair (d_p, e_p) . Once again, Joe creates a short message ("I am Joe") as a signature (S) and encrypts his signature with his private key d_j . Joe then attaches the encrypted signature to the will and creates a message M where:

$$M = W + d_j(S)$$

To send M securely to Pat, Joe encrypts the whole message M by using Pat's public key and generating $e_p(M)$. Pat receives this cipher and decrypts it by using Pat's private key d_p ; i.e., she obtains M by using $D(E(M)) = M$. But the message $M = W + d_j(S)$; i.e., it has the will plus the signature. The will is in plaintext now but the signature is still encrypted. The ciphertext $d_j(S)$ is decrypted by Pat by using Joe's public key e_j . This means that the signature is Joe's signature because the ciphertext can only be decrypted if sent from the right person.

In real life, digital signature protocols are more complicated than this to make sure that the signatures cannot be forged, modified, or attached to different documents. Consider, for example, if a digitally signed document was sent but a malicious individual detached the signature from the document and attached it to another document or modified the document in some way. In other words, it is extremely important to make sure that the message is not modified. This is accomplished through message digests (presented in the next section). Another way to assure the identity of the remote user in prolonged conversations is to issue a "challenge" (a random number) that is sent to the remote user. The remote user encrypts the challenge with her private key and sends it back to you. Now, if you can decrypt the challenge by using the remote users' public key, then you can continue. If not, you should disconnect (and perhaps, send some impolite message to the intruder!).

4.5.3 e-Signature Law

Technically speaking, digital signatures provide greater reliability for identification of a signer than conventional signatures. However, there are several issues related to associating a key with a person. How do we know that an adversary is not using Joe's key that he is using for digital signatures? This is similar to the question with conventional signatures: how do we know if someone learned how Joe signs his name and forged his signature? A solution to this problem is to bring in a third party – a certificate authority (CA) – to electronically verify the identity of the key-holder.

Due to the potential issues with digital signatures, the federal Electronics Signatures Act, commonly known as the e-signature law, took effect in October 2000. This law accords electronic signatures the same legal status as conventional signatures. The law states that a signature cannot be turned down just because it takes electronic form. The bill does not, however, specify the technology to be used for the e-signature. It also does not specify the exact form of an e-signature. Thus an electronic certificate or an encrypted key could be possibly used as e-signature. This vagueness is expected to help new technologies to be developed and used in this area. But on the other hand, it creates more litigation

opportunities for the lawyers to exactly define what an e-signature is. Doesn't it seem that some lawyers wrote this law!

A great deal of information on this topic is accumulating quickly. See, for example, the site (www.chadbourn.com/Publications/Memo/Esignature.htm) for discussion.

4.6 Message Digests for Information Integrity

Both symmetric and asymmetric key cryptography provide some built-in integrity checking to make sure that the information is not modified in transit. Modified messages do not decrypt correctly. However, this is not a strong integrity check because messages are typically encrypted in small blocks of text. It is possible for a portion of an encrypted message to be deleted or duplicated without any problems. In addition, it is possible to take someone's signature from one document and attach it to another document without causing a decryption problem. In many secure communications, it is extremely important that the slightest change in a document causes the receiving process to detect the change and produce an error. In many cases, integrity of information is more important than privacy. Encryption schemes handle privacy but do not guarantee integrity. This is extremely important in wireless systems because wireless communications are very prone to errors due to rain, thunderstorms, and God's other creation known as hackers.

Message digesting is used to make sure that a certain message was not changed along the way between the sender and the receiver. A message digest algorithm $f()$, as shown in Figure 4-8, produces a fingerprint of the message, by applying a hashing function to it. The receiver can check for the integrity of the message by reapplying the hash function and comparing it with the original fingerprint. The hash functions used in these schemes are such that the fingerprint changes dramatically if a single bit of the message changes. Two messages cannot generate the same hash.

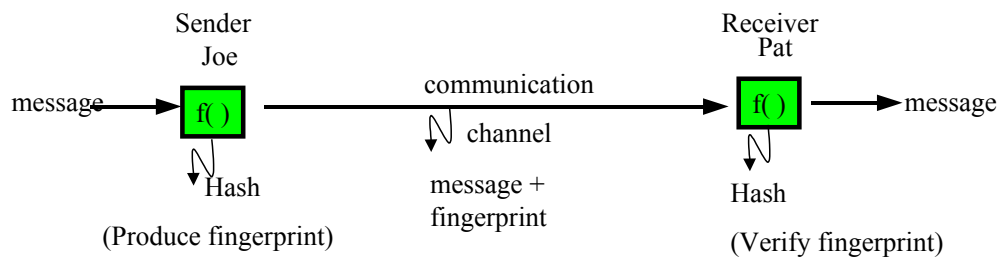


Figure 4-8: Message Digest

The message digest algorithms produce the fingerprint of a message using hashing schemes to verify its integrity in a manner that is similar to encryption. However, no decryption is required. This is why it is also called one-way encryption. The process works as follows:

- Sender creates a hash h_1 and sends it with message
- Receiver gets the message and creates a hash h_2

- If $h_1 = h_2$, then everything is okay; otherwise loss of data is indicated

A one-way hash function takes variable-length input, a message of even thousands or millions of bits, and produces a fixed-length output; say, 128 bits. The hash function ensures that if the information is changed in any way, an entirely different output value is produced.

Several systems such as PGP (Pretty Good Privacy) use a cryptographically strong hash function on plaintext. This generates a fixed-length data item known as a *message digest*. Once again, any change to the information results in a totally different digest. Let us take the example of PGP a little further. After creating the digest, PGP uses the digest and the private key to create the “signature.” PGP transmits the signature and the plaintext together. Upon receipt of the message, the recipient uses PGP to recompute the digest, thus verifying the signature. By using a secure hash function, you cannot take someone’s signature from one document and attach it to another, or alter a signed message in any way. The slightest change in a signed document will cause the digital signature verification process to fail.

Many hashing algorithms are in use (no surprise!). Most popular algorithms are a) MD5, the most widely used message digest function, producing a 128-bit hash, and b) SHA, a secure hash algorithm developed by NIST for a digital signature standard – it uses a 160-bit hash.

4.7 Digital Envelopes – Combining Symmetric and Asymmetric Key Systems

Public key systems are quite secure but are quite slow due to their complexity. For example, the RSA algorithm is very complex because it performs intricate operations on large prime numbers. Performance of RSA is very slow (can be up to 1000 times slower than private key algorithms [Stein 1998]). The performance degrades with message size, thus RSA is not suitable for large documents.

A good compromise is the digital envelope that uses a mixture of symmetric and asymmetric encryption. Digital envelopes use public keys only to encrypt the keys of a symmetric system instead of the entire message. The operation is similar to that of a digital signature:

- Generate a secret key (called a “session key” because it is discarded after the communication session is done).
- Encrypt the message by using the session key and *symmetric algorithm* (e.g., DES).
- Encrypt the session key with the receiver’s public key. This becomes the **digital envelope**.
- Send the digital envelope plus the encrypted message to the receiver.

Digital envelopes combine symmetric and asymmetric key systems and are the most commonly used encryption technique at present. They are used widely in many commercially available security systems such as PGP, SSL, SET, and others.

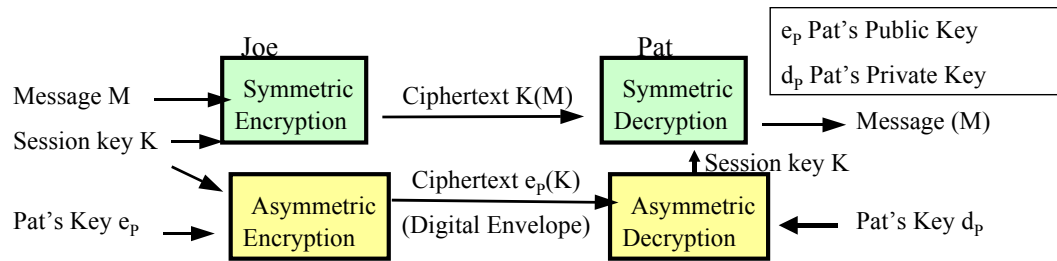


Figure 4-9: Digital Envelope

Figure 4-9 illustrates this process using our friends Joe and Pat. In essence, two encryption processes take place on the sender side as well as on the receiver side. On the sender (Joe) side, the session key K is used to encrypt the message M by using symmetric encryption. The key K is also encrypted by using Pat's public key, and a digital envelope is created that contains the session key. On the receiver (Pat) side, first the digital envelope is opened using Pat's private key. This produces the session key K that is then used to decrypt the message M .

Diffie-Hellman: Encryption Without Authentication

Public key cryptography usually requires authentication. However, in some cases, it may be desirable for two parties to communicate with each other securely but with anonymity. An algorithm known as Diffie-Hellman allows two parties to negotiate a session key without ever sending the key itself across the network. This algorithm works by having the two parties pick a partial key independently. Then they exchange enough information with each other so that each can independently build a session key but not enough so that an eavesdropper can reconstruct the session key. This key is now used in a symmetric key fashion to encrypt and exchange documents. When the exchange is done, the key is discarded.

Diffie-Hellman suffers from a problem known as “man in the middle.” In this case, a malicious individual C could intercept the traffic between the two parties (A and B) and masquerade as B to A and vice versa. See the following section for a discussion of this problem. This type of attack is more realistic in wireless networks.

4.8 Man in the Middle – Security Weakness of Public Key Systems

Public key systems, in general, provide very strong protection for information storage and transmission. However, there is a well-known problem with public key systems, called “man in the middle,” that should be mentioned here. This problem is associated with a special class of security scenarios, known as Diffie-Hellman, that involves encryption without authentication (see the sidebar “Diffie-Hellman: Encryption Without Authentication”). However, it is serious enough, especially in wireless networks, that it

should be noted generally. The main problem is that a malicious individual (“man in the middle”) could intercept the traffic between a sender and receiver and then send his own public key instead of the public key of the receiver; he could later decrypt the information by using his own private key. Let us go through some details using Figure 4-10.

Let us assume that B is the sender and A is the receiver, but C is a malicious man in the middle. Since we are using public key cryptography, each person (A, B, C) has two keys (e_A, d_A), (e_B, d_B), (e_C, d_C) where, according to our convention, E is the public and D is the private key. The following scenario is plausible:

- B wants to send message M to A, so B needs A’s public key e_A .
- C is dishonest (man in the middle) and intercepts this message.
- C sends its own public key e_C to B.
- B encrypts the message and sends it out.
- C intercepts the encrypted message and decrypts it by using his own private key d_C .

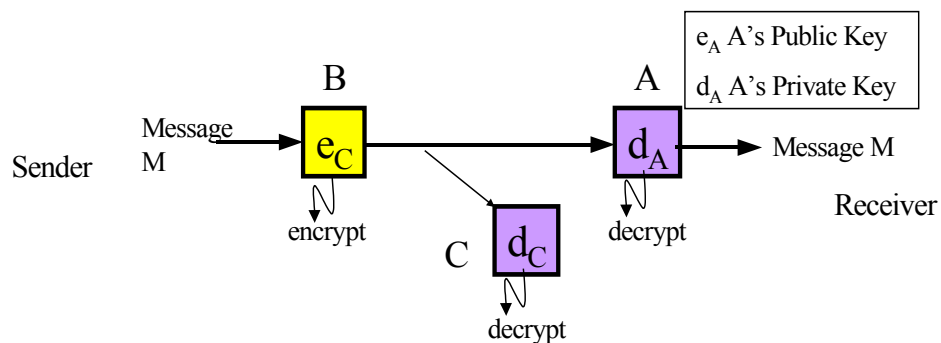


Figure 4-10: Man in the Middle – C is the Malicious person

To protect against this type of problem, the public key must also be protected, and mechanisms must exist to verify that if you ask for A’s public key, you in fact are getting A’s public key. This is done through Certificate Authorities (CAs) that maintain a user’s public key as a certificate. CAs are part of Public Key Infrastructure (PKI), discussed in the next section.

4.9 Examples of Encryption Systems

Numerous systems in the market today employ encryption technologies discussed in this chapter. Examples of these systems include (we will discuss them in a later chapter):

- SSL (Secure Socket Layer) is the de facto standard for securing the traffic between Web clients and Web servers. SSL uses encryption, digital certificates, and digital signatures.
- IPSec is a standard used in Virtual Private Networks (VPNs) to encrypt and decrypt IP packets in the Internet. By using VPNs, customers can get a secure path over the public Internet by encrypting the IP traffic.

- PGP (Pretty Good Privacy) provides encryption and digital signature services. PGP is used commonly to protect email. Free and commercial versions of PGP are available.
- PKI (Public Key Infrastructure) is an extensive collection of security technologies (encryption, digital signatures, digital certificates) for a wide range of applications. However, due to its extensive capabilities, PKI is somewhat difficult to use.

4.10 Concluding Comments: How Symmetric and Asymmetric Systems are Really Being Used in Practice

Most current systems use a mixture of symmetric and asymmetric key cryptography. The reason is simple: Asymmetric key systems are quite secure but are quite slow due to their complexity; while symmetric key systems are the other way around (weak in security but strong in performance). Ideally, you would like to use the RSA algorithm or other asymmetric key algorithm on the entire plaintext, but they are very very slow (up to 1000 times slower than symmetric). In addition, the performance of RSA degrades with message size; thus RSA is not suitable for large documents.

A good compromise is the digital envelope approach that uses a mixture of symmetric and asymmetric encryption. Digital envelopes use a public key only to encrypt the *keys* of a symmetric system instead of the entire message. The operation works like this

- Generate a secret key by using a random number generator (called a “session key” because it is discarded after the communication session is done).
- Encrypt the message by using the session key and *symmetric algorithm* (e.g., DES).
- Encrypt the session key with the receiver’s public key. This creates a digital envelope.
- Send the digital envelope plus the encrypted message to the receiver.

This approach is used widely in systems such as PGP, SSL, SET, and others.

4.11 Short Case Studies and Examples

4.11.1 Legal Issues Concerning Publication of Encryption Algorithms

Several legal issues surround publication of encryption algorithms and shipment of encryption code overseas. The situation is not clear, but the following cases give some insights.

One of the first cases is the one filed in September 1995 by cryptographer Philip Karn, who was denied permission to export a computer disk containing the encryption software source code in the book *Applied Cryptography*. In another case, an Ohio professor (Case Western Reserve University law professor Peter Junger) was told by a federal court that he does not have a First Amendment right to post encryption code on his course Web

site. Junger challenged federal restrictions on strong encryption. The technology requires an export license because it is considered a potential weapon under the law. Junger waged the court fight in 1996 to ensure his right to teach foreign and local students about data security technology by posting material on his Web site. The Ohio court ruled that although books containing encryption code can be shipped overseas without a license, websites containing the code are not protected to the same extent by the First Amendment. Specifically, the U.S. District Court Judge James Gwin of the Northern District of Ohio ruled that encryption software code does not warrant the same constitutional protection as other speech.

The ruling of Judge Gwin flew in the face of another federal ruling by the U.S. government in California. The California case was filed by University of Illinois math professor Daniel Bernstein, who wanted to post online the code of an encryption program he wrote. Bernstein filed suit in 1995 after spending three years fighting with the federal government over whether a simple encryption program could be freely distributed on the Internet. U.S. law at the time deemed online publication an “export” that could be punished with severe prison terms. In the Bernstein case, California federal jurist Marilyn Hall Patel called software a “language” that held the same constitutional protection as books or other forms of public speech. Judge Patel said the government’s rules were unconstitutional. But Gwin disputed the Bernstein ruling. “The court in Bernstein misunderstood the significance of source code’s functionality,” he ruled.

The Bernstein and Junger cases – along with Karn’s case – created a great deal of discussion in the late 1990s. The software industry tried to move Congress and the Clinton administration to throw out the rules, which they said are bureaucratic and prohibit them from competing with foreign manufacturers that can ship stronger products without restrictions. Those who believe academia has a right to post encryption code online say the government’s licensing policy is stifling academic freedom and free speech.

After years of legal wranglings, U.S. District Judge Patel in San Francisco threw out the Bernstein case in May 2003, after the Bush administration said it would no longer try to enforce portions of the regulations. The cases of Bernstein-Junger-Karn have been credited with forcing the federal government to drastically scale back its attempts to regulate the privacy-protecting encryption technology. At one point such encryption was regulated by the State Department and treated as a weapon like tanks and fighter jets, but the Clinton and Bush administrations have relaxed the rules. But the issues are not completely settled. Stay tuned.

Sources:

- McCullagh, D., “Cold War Encryption Laws Stand, But not as Firmly,” CNET News.com, May 2003
- Macavinta, C., “Professor Loses Crypto Case,” CNET News.com, July 6, 1998
- Goodin, D., “Crypto ruling impact unclear,” CNET News.com, August 26, 1997

4.11.2 Digital Signatures Support Mobile Home Banking

More than one million customers in Danish savings banks can do secure banking from Web browsers anywhere in the world. The new Internet banking solution from the Savings Banks Data Center (SDC) is based on mobile digital signatures. The SDC

provides IT and related services to Danish banks (80 banks with 506 branch offices and 1.1 million customers).

Suppose that you wanted to pay a bill while on vacation (and you do not have your laptop with you, since a vacation should be “laptopless”!). You can go to the nearest computer with Internet access (hotels provide that) and start home banking by using SDC’s Café Bank. To use Café Bank, you only need a mobile phone and a computer with Internet access. You access the bank’s website as usual and log on by entering your password as usual. A few seconds later you receive an SMS (Short Message Service) message with a Café id-code. You enter the id-code and are now able to use the banking application as usual.

How does it work? SDC’s traditional home banking solution requires a digital signature. To apply a digital signature to a transaction, the user needs a signature key. This key is stored on his or her home PC, but a copy is kept on a central server. When traveling, the customer uses the Café id-code to identify herself and access the key stored centrally. The signature server is delivered by Cryptomathic and handles generation of digital signatures. To access Café Bank and perform transactions, the users have to authenticate themselves to the signature server. The signature server provides strong two-factor authentication using a static password and a one-time password – the Café id-code sent to the user’s mobile phone via SMS. The keys stored on the signature server are protected by a Hardware Security Module, where the signatures are created. Firewalls are set up to form a demilitarized zone around the Web server.

Safe storage of signature keys can be a problem in digital signature applications. Software key stores on client machines are common, but they offer only limited security and mobility. Hardware key stores, like chip cards, offer higher security and some mobility, but smart card readers are not yet included in standard PCs. Cryptomathic’s signature server attempts to combine the best of the two worlds by offering protection, physical security and user mobility.

Source http://www.cryptomathic.com/pdf/home_banking_leaves_home.pdf

4.11.3 Security in Health Care

The healthcare industry presents numerous security challenges. The goal of healthcare information systems is to provide open but secure access to information. The systems are old – one of the oldest in the industry – and not well designed. In addition, the designers are not sure how to deal with the Internet because no clear policies for Internet use have been developed. To complicate matters further, the industry is heavily regulated, with new privacy laws being introduced regularly. The designers have to conform systems to cope with the regulations. Perhaps the most difficult problem is that consumer distrust is quite high – thus any mistake can have serious side effects.

Development of secure healthcare information systems (HIS) requires a systematic management approach that includes:

- Development of clear policies about who can access the information, how the information can be used, how account IDs are assigned, how the Internet can be used, etc.
- Establishing security requirements about the most valuable assets. In case of healthcare, the most valuable asset is the patient information. This information must

be protected for privacy, integrity, authorization, authentication, accountability, and availability (PIA4) attacks.

- Detailed risk analysis to understand the vulnerabilities of the HIS. The vulnerabilities must pinpoint the resources plus the paths (networks) to the resources. Another vulnerability is that adversaries can pose as doctors or researchers to access unauthorized information. Thus vulnerabilities in authentication must be seriously looked at.
- Development of circumventions and policies to mitigate risks. Policies of punishment can serve as a deterrent (see the results of the 1999 Consumer's Union survey, discussed below). In addition, security technologies of encryption, password protection, digital signatures, and audit trails must be used heavily.

Since the main risk of a security breach is serious consumer impact, the healthcare industry tries to gauge consumer attitudes on a regular basis. To develop an understanding of consumer attitudes, an interesting survey was conducted by the Consumer's Union in 1999. The survey found the following:

- Willingness to share: consumers were largely unwilling to share information with anyone. There was some willingness to share information only with researchers (50% of the respondents were willing to share information for research).
- Perceived threat: The largest threat was hackers (70%).
- Effective Safeguards: Punishments and fines were rated very high as deterrents. Requiring specific permission from owners was highly recommended, and use of technology for better security was considered most effective.

The key point is that the consumers do not trust health plans or providers. Consumers also do not trust computers. The most surprising finding was that the consumers will compromise quality of healthcare for privacy.

There are several issues that need more investigation. The main question is: can the opinions of consumers be turned around? In particular, can the increased use of the Internet be used as an example? Consumers now feel comfortable giving credit card information over the Internet, but they took quite a while to develop this type of confidence. Finally, can improved technical solutions win consumer confidence? It is difficult to predict this because more experience with technology can have positive as well as negative impact on consumers.

Source:

M. Krause, "Information Security Management in the Healthcare Industry," in *Information Security Management Handbook*, ed. H. Tipton and M. Kraus, 4th and 5th ed. (Auerbach, 2000).

4.12 Suggested Review Questions

1. What is cryptography and what are its main ingredients?
2. What are the key differences between symmetric and asymmetric key cryptography?
3. List the main security technologies and discuss how are they related to cryptography.
4. What are digital signatures and how do they differ from message digests?
5. How do digital envelopes combine symmetric and asymmetric keys?

6. Explain the “man in the middle” problem through an example.