CHAPTER 30

Cryptography

Network security is mostly achieved through the use of cryptography, a science based on abstract algebra. In this chapter, we briefly discuss the cryptography suitable for the scope of this book. We have tried to limit our discussion of abstract algebra as much as we could. Our goal is to give enough information about cryptography to make network security understandable. The chapter opens the door for studying network security in Chapter 31 and Internet security in Chapter 32.

30.1 INTRODUCTION

Let us introduce the issues involved in cryptography. First, we need to define some terms; then we give some taxonomies.

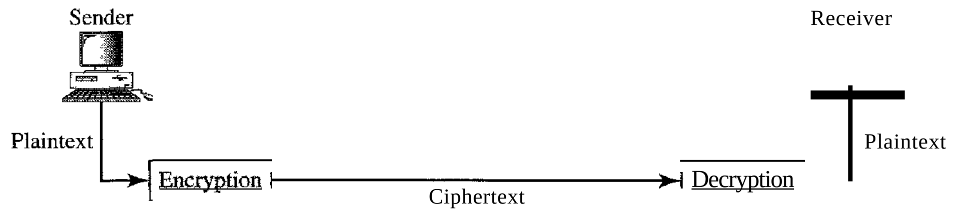
Definitions

We define some terms here that are used in the rest of the chapter.

Cryptography

Cryptography, a word with Greek origins, means "secret writing." However, we use the term to refer to the science and art of transforming messages to make them secure and immune to attacks. Figure 30.1 shows the components involved in cryptography.

Figure 30.1 *Cryptography components*



Plaintext and Ciphertext

The original message, before being transformed, is called plaintext. After the message is transformed, it is called ciphertext. An encryption algorithm transforms the plaintext into ciphertext; a decryption algorithm transforms the ciphertext back into plaintext. The sender uses an encryption algorithm, and the receiver uses a decryption algorithm.

Cipher

We refer to encryption and decryption algorithms as ciphers. The term *cipher* is also used to refer to different categories of algorithms in cryptography. This is not to say that every sender-receiver pair needs their very own unique cipher for a secure communication. On the contrary, one cipher can serve millions of communicating pairs.

Key

A key is a number (or a set of numbers) that the cipher, as an algorithm, operates on. To encrypt a message, we need an encryption algorithm, an encryption key, and the plaintext. These create the ciphertext. To decrypt a message, we need a decryption algorithm, a decryption key, and the ciphertext. These reveal the original plaintext.

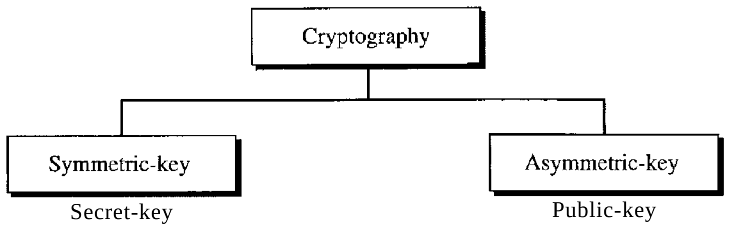
Alice, Bob, and Eve

In cryptography, it is customary to use three characters in an information exchange scenario; we use Alice, Bob, and Eve. Alice is the person who needs to send secure data. Bob is the recipient of the data. Eve is the person who somehow disturbs the communication between Alice and Bob by intercepting messages to uncover the data or by sending her own disguised messages. These three names represent computers or processes that actually send or receive data, or intercept or change data.

Two Categories

We can divide all the cryptography algorithms (ciphers) into two groups: symmetric-key (also called secret-key) cryptography algorithms and asymmetric (also called public-key) cryptography algorithms. Figure 30.2 shows the taxonomy.

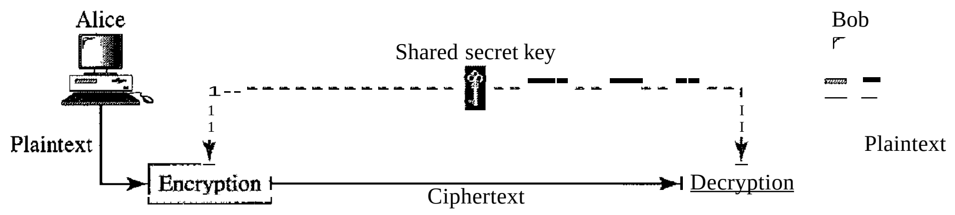
Figure 30.2 *Categories of cryptography*



Symmetric·Key Cryptography

In symmetric-key cryptography, the same key is used by both parties. The sender uses this key and an encryption algorithm to encrypt data; the receiver uses the same key and the corresponding decryption algorithm to decrypt the data (see Figure 30.3).

Figure 30.3 *Symmetric-key cryptography*

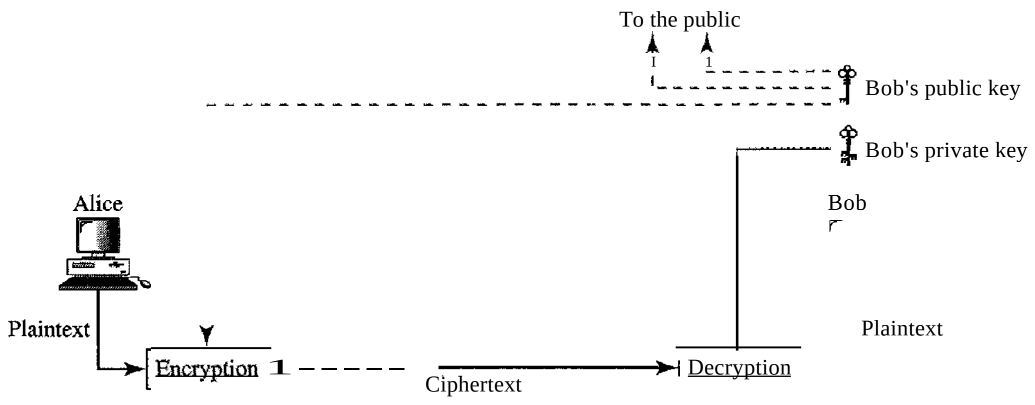


In symmetric·key cryptography, the same key is used by the sender (for encryption) and the receiver (for decryption). The key is shared.

Asymmetric-Key Cryptography

In asymmetric or public-key cryptography, there are two keys: a private key and a public key. The private key is kept by the receiver. The public key is announced to the public. In Figure 30.4, imagine Alice wants to send a message to Bob. Alice uses the public key to encrypt the message. When the message is received by Bob, the private key is used to decrypt the message.

Figure 30.4 *Asymmetric-key cryptography*

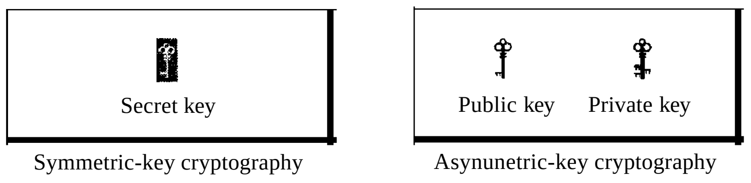


In public-key encryption/decryption, the public key that is used for encryption is different from the private key that is used for decryption. The public key is available to the public;' the private key is available only to an individual.

Three Types ofKeys

The reader may have noticed that we are dealing with three types of keys in cryptography: the secret key, the public key, and the private key. The first, the secret key, is the shared key used in symmetric-key cryptography. The second and the third are the public and private keys used in asymmetric-key cryptography. We will use three different icons for these keys throughout the book to distinguish one from the others, as shown in Figure 30.5.

Figure 30.5 *Keys used in cryptography*

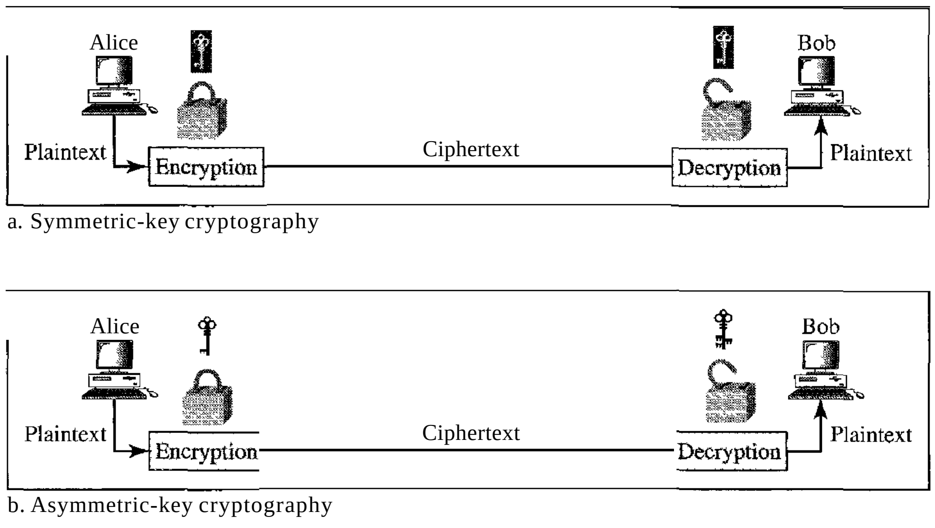


Comparison

Let us compare symmetric-key and asymmetric-key cryptography. Encryption can be thought of as electronic locking; decryption as electronic unlocking. The sender puts the message in a box and locks the box by using a key; the receiver unlocks the box with a key and takes out the message. The difference lies in the mechanism of the locking and unlocking and the type of keys used.

In symmetric-key cryptography, the same key locks and unlocks the box. In asymmetric-key cryptography, one key locks the box, but another key is needed to unlock it. Figure 30.6 shows the difference.

Figure 30.6 *Comparison between two categories ofcryptography*



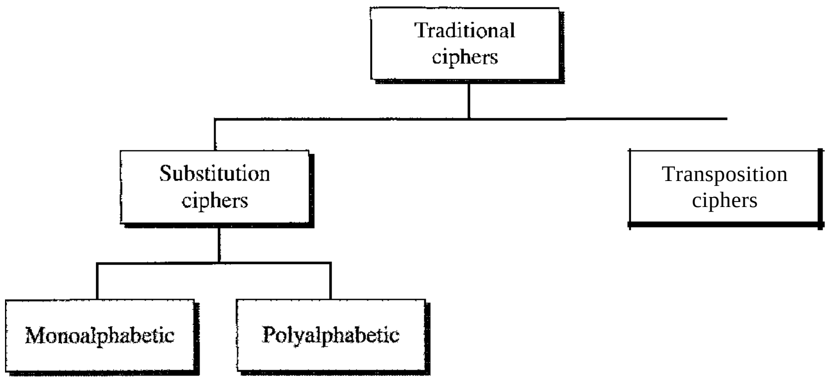
30.2 SYMMETRIC-KEY CRYPTOGRAPHY

Symmetric-key cryptography started thousands of years ago when people needed to exchange secrets (for exanlple, in a war). We still mainly use symmetric-key cryptography in our network security. However, today's ciphers are much more complex. Let us first discuss traditional algorithms, which were character-oriented. Then we discuss the modem ones, which are bit-oriented.

Traditional Ciphers

We briefly introduce some traditional ciphers, which are character-oriented. Although these are now obsolete, the goal is to show how modern ciphers evolved from them. We can divide traditional symmetric-key ciphers into two broad categories: substitution ciphers and transposition ciphers, as shown in Figure 30.7.

Figure 30.7 *Traditional ciphers*



Substitution Cipher

A substitution cipher substitutes one symbol with another. If the symbols in the plaintext are alphabetic characters, we replace one character with another. For example, we can replace character A with D, and character T with Z. If the symbols are digits (0 to 9), we can replace 3 with 7, and 2 with 6. Substitution ciphers can be categorized as either monoalphabetic or polyalphabetic ciphers.

A substitution cipher replaces one symbol with another.

In a monoalphabetic cipher, a character (or a symbol) in the plaintext is always changed to the same character (or symbol) in the ciphertext regardless of its position in the text. For example, if the algorithm says that character A in the plaintext is changed to character D, every character A is changed to character D. In other words, the relationship between characters in the plaintext and the ciphertext is a one-to-one relationship.

In a polyalphabetic cipher, each occurrence of a character can have a different substitute. The relationship between a character in the plaintext to a character in the ciphertext is a one-to-many relationship. For example, character A could be changed to D in the beginning of the text, but it could be changed to N at the middle. It is obvious that if the relationship between plaintext characters and ciphertext characters is one-to-many, the key must tell us which of the many possible characters can be chosen for encryption. To achieve this goal, we need to divide the text into groups of characters and use a set of keys. For example, we can divide the text "THISISANEASYTASK" into groups of 3 characters and then apply the encryption using a set of 3 keys. We then repeat the procedure for the next 3 characters.

Example 30.1

The following shows a plaintext and its corresponding ciphertext. Is the cipher monoalphabetic?

Plaintext: HELLO
Ciphertext: KHOOR

Solution

The cipher is probably monoalphabetic because both occurrences of L's are encrypted as O's.

Example 30.2

The following shows a plaintext and its corresponding ciphertext. Is the cipher monoalphabetic?

Plaintext: HELLO
Ciphertext: ABNZF

Solution

The cipher is not monoalphabetic because each occurrence of L is encrypted by a different character. The first L is encrypted as N; the second as Z.

Shift Cipher The simplest monoalphabetic cipher is probably the shift cipher. We assume that the plaintext and ciphertext consist of uppercase letters (A to Z) only. In this cipher, the encryption algorithm is "shift *key* characters down," with *key* equal to some number. The decryption algorithm is "shift *key* characters up." For example, if the key is 5, the encryption algorithm is "shift 5 characters down" (toward the end of the alphabet). The decryption algorithm is "shift 5 characters up" (toward the beginning of the alphabet). Of course, if we reach the end or beginning of the alphabet, we wrap around.

Julius Caesar used the shift cipher to communicate with his officers. For this reason, the shift cipher is sometimes referred to as the Caesar cipher. Caesar used a key of 3 for his communications.

The shift cipher is sometimes referred to as the Caesar cipher.

Example 30.3

Use the shift cipher with key = 15 to encrypt the message "HELLO."

Solution

We encrypt one character at a time. Each character is shifted 15 characters down. Letter H is encrypted to W. Letter E is encrypted to T. The first L is encrypted to A. The second L is also encrypted to A. And 0 is encrypted to D. The cipher text is WTAAD.

Example 30.4

Use the shift cipher with key = 15 to decrypt the message "WTAAD."

Solution

We decrypt one character at a time. Each character is shifted 15 characters up. Letter W is decrypted to H. Letter T is decrypted to E. The first A is decrypted to L. The second A is decrypted to L. And, finally, D is decrypted to O. The plaintext is HELLO.

Transposition Ciphers

In a transposition cipher, there is no substitution of characters; instead, their locations change. A character in the first position of the plaintext may appear in the tenth position of the ciphertext. A character in the eighth position may appear in the first position. In other words, a transposition cipher reorders the symbols in a block of symbols.

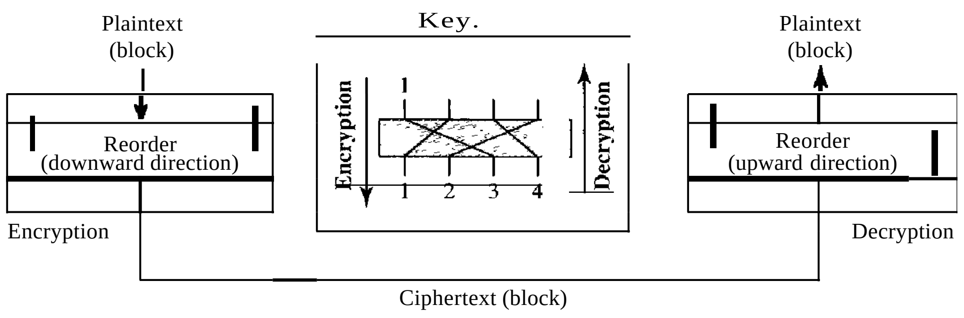
A transposition cipher reorders (permutes) symbols in a block of symbols.

Key In a transposition cipher, the key is a mapping between the position of the symbols in the plaintext and cipher text. For example, the following shows the key using a block of four characters:

Plaintext:	2	4	1	3
Ciphertext:	1	2	3	4

In encryption, we move the character at position 2 to position 1, the character at position 4 to position 2, and so on. In decryption, we do the reverse. Note that, to be more effective, the key should be long, which means encryption and decryption of long blocks of data. Figure 30.8 shows encryption and decryption for our four-character

Figure 30.8 *Transposition cipher*



block using the above key. The figure shows that the encryption and decryption use the same key. The encryption applies it from downward while decryption applies it upward.

Example 30.5

Encrypt the message "HELLO MY DEAR," using the above key.

Solution

We first remove the spaces in the message. We then divide the text into blocks of four characters. We add a bogus character Z at the end of the third block. The result is HELL OMYD EARZ. We create a three-block ciphertext ELHLMDOYAZER.

Example 30.6

Using Example 30.5, decrypt the message "ELHLMDOYAZER".

Solution

The result is HELL OMYD EARZ. After removing the bogus character and combining the characters, we get the original message "HELLO MY DEAR."

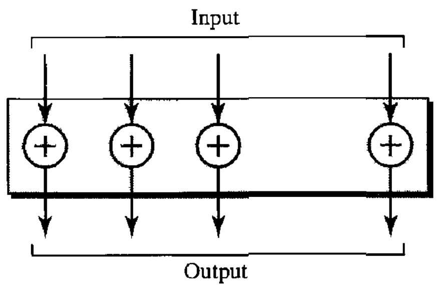
Simple Modern Ciphers

The traditional ciphers we have studied so far are character-oriented. With the advent of the computer, ciphers need to be bit-oriented. This is so because the information to be encrypted is not just text; it can also consist of numbers, graphics, audio, and video data. It is convenient to convert these types of data into a stream of bits, encrypt the stream, and then send the encrypted stream. In addition, when text is treated at the bit level, each character is replaced by 8 (or 16) bits, which means the number of symbols becomes 8 (or 16). Mingling and mangling bits provides more security than mingling and mangling characters. Modem ciphers use a different strategy than the traditional ones. A modern symmetric cipher is a combination of simple ciphers. In other words, a modern cipher uses several simple ciphers to achieve its goal. We first discuss these simple ciphers.

XOR Cipher

Modern ciphers today are normally made of a set of **simple ciphers,** which are simple predefined functions in mathematics or computer science. The first one discussed here is called the **XOR cipher** because it uses the exclusive-or operation as defined in computer science. Figure 30.9 shows an XOR cipher.

Figure 30.9 *XOR cipher*

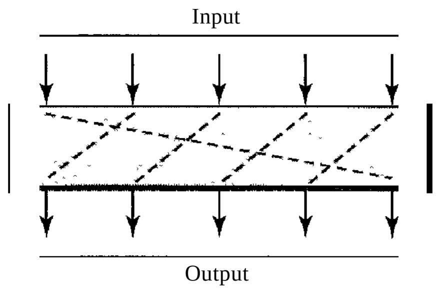


An XOR operation needs two data inputs plaintext, as the first and a key as the second. In other words, one of the inputs is the block to be the encrypted, the other input is a key; the result is the encrypted block. Note that in an XOR cipher, the size of the key, the plaintext, and the ciphertext are all the same. XOR ciphers have a very interesting property: the encryption and decryption are the same.

Rotation Cipher

Another common cipher is the rotation cipher, in which the input bits are rotated to the left or right. The rotation cipher can be keyed or keyless. In keyed rotation, the value of the key defines the number of rotations; in keyless rotation the number of rotations is fixed. Figure 30.10 shows an example of a rotation cipher. Note that the rotation cipher can be considered a special case of the transpositional cipher using bits instead of characters.

Figure 30.10 *Rotation cipher*



The rotation cipher has an interesting property. If the length of the original stream is *N*, after *N* rotations, we get the original input stream. This means that it is useless to apply more than $N - 1$ rotations. In other words, the number of rotations must be between 1 and $N-1$.

The decryption algorithm for the rotation cipher uses the same key and the opposite rotation direction. If we use a right rotation in the encryption, we use a left rotation in decryption and vice versa.

Substitution Cipher: S-box

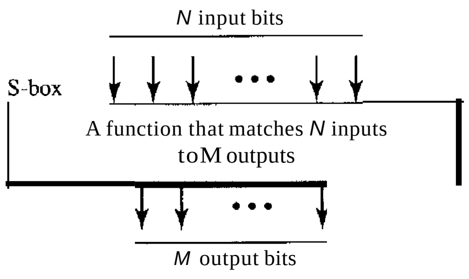
An S-box (substitution box) parallels the traditional substitution cipher for characters. The input to an S-box is a stream of bits with length *N;* the result is another stream of bits with length *M.* And Nand *M* are not necessarily the same. Figure 30.11 shows an S-box.

The S-box is normally keyless and is used as an intermediate stage of encryption or decryption. The function that matches the input to the output may be defined mathematically or by a table.

Transposition Cipher: P-box

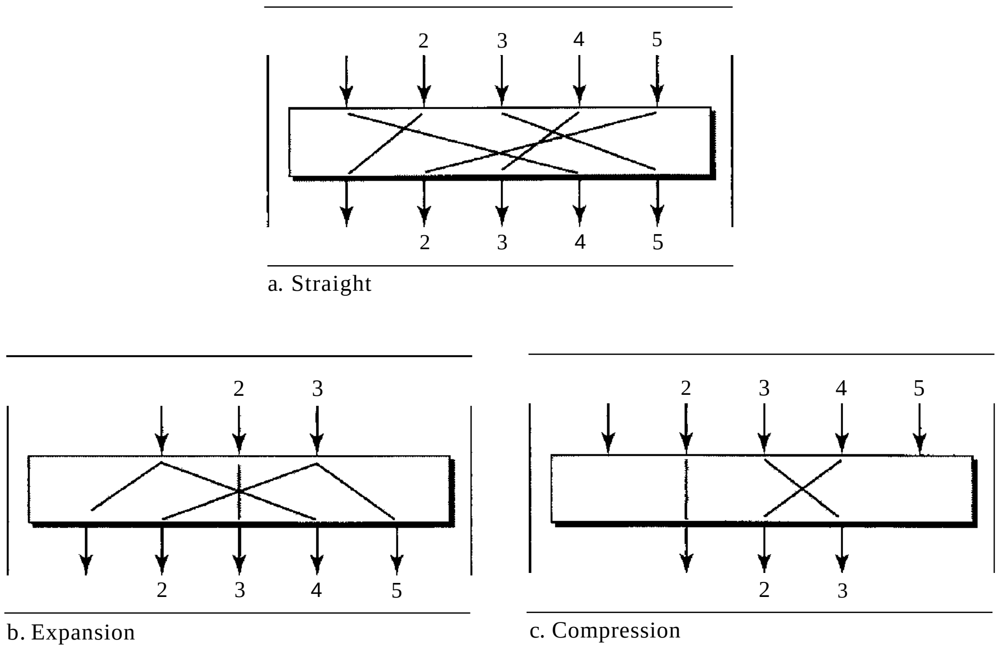
A P-box (permutation box) for bits parallels the traditional transposition cipher for characters. It performs a transposition at the bit level; it transposes bits. It can be implemented

Figure 30.11 *S-box*



in software or hardware, but hardware is faster. P-boxes, like S-boxes, are nonnally keyless. We can have three types of pennutations in P-boxes: the **straight permutation, expansion permutation, and compression permutation** as shown in Figure 30.12.

Figure 30.12 *P-boxes: straight, expansion, and compression*



A straight permutation cipher or a straight P-box has the same number of inputs as outputs. In other words, if the number of inputs is *N,* the number of outputs is also *N.* In an expansion pennutation cipher, the number of output ports is greater than the number of input ports. In a compression pennutation cipher, the number of output ports is less than the number of input ports.

Modern Round Ciphers

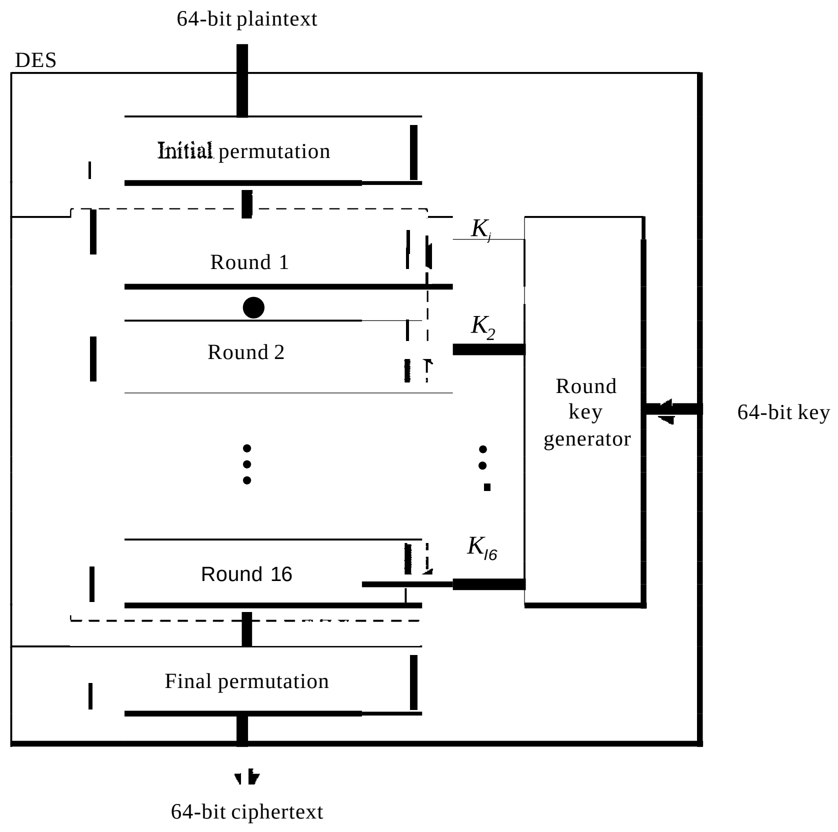
The ciphers of today are called **round ciphers** because they involve multiple **rounds,** where each round is a complex cipher made up of the simple ciphers that we previously described. The key used in each round is a subset or variation of the general key called the round key. If the cipher has N rounds, a key generator produces N keys, $K_1, K_2, \ldots, K_N$, where K_1 is used in round 1, K_2 in round 2, and so on.

In this section, we introduce two modem symmetric-key ciphers: DES and AES. These ciphers are referred to as block ciphers because they divide the plaintext into blocks and use the same key to encrypt and decrypt the blocks. DES has been the de facto standard until recently. AES is the formal standard now.

Data Encryption Standard (DES)

One example of a complex block cipher is the Data Encryption Standard (DES). DES was designed by IBM and adopted by the U.S. government as the standard encryption method for nonmilitary and nonclassified use. The algorithm encrypts a 64-bit plaintext block using a 64-bit key, as shown in Figure 30.13.

Figure 30.13 *DES*

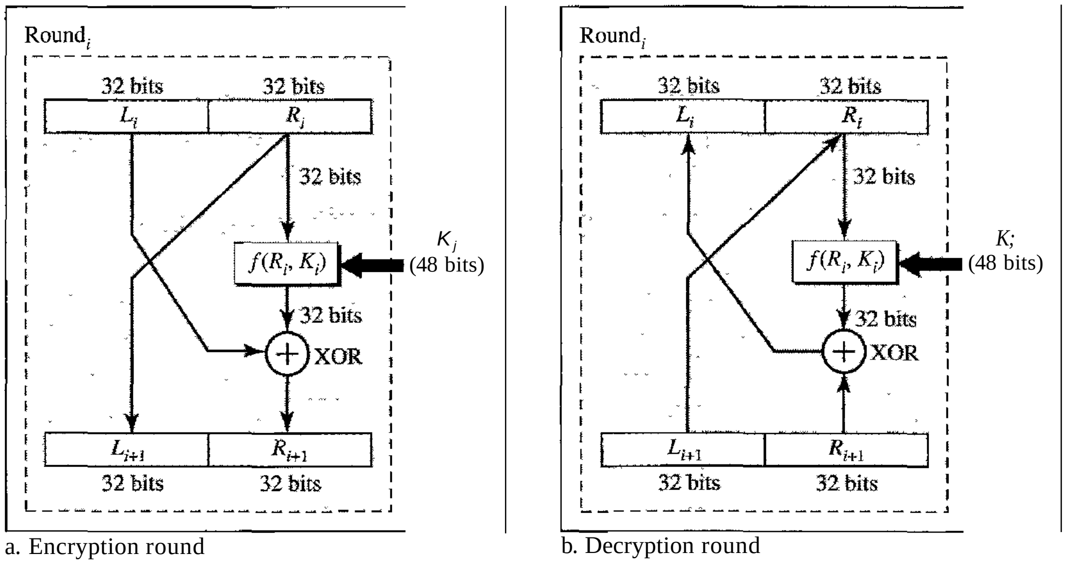


DES has two transposition blocks (P-boxes) and 16 complex round ciphers (they are repeated). Although the 16 iteration round ciphers are conceptually the same, each uses a different key derived from the original key.

The initial and final permutations are keyless straight permutations that are the inverse of each other. The permutation takes a 64-bit input and permutes them according to predefined values.

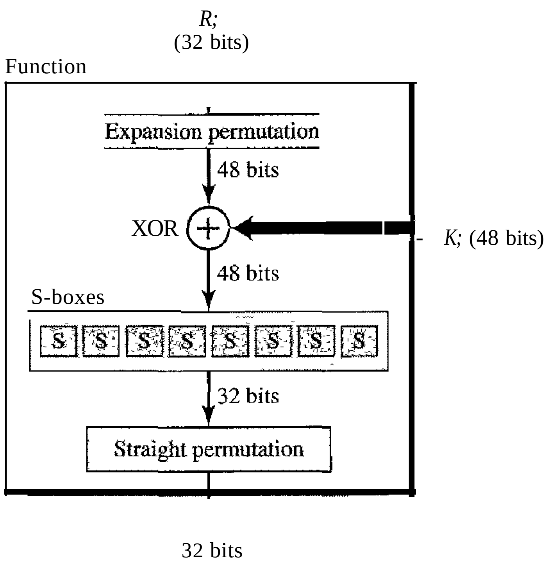
Each round of DES is a complex round cipher, as shown in Figure 30.14. Note that the structure of the encryption round ciphers is different from that of the decryption one.

Figure 30.14 *One round in DES ciphers*



DES Function The heart of DES is the **DES function.** The DES function applies a 48-bit key to the rightmost 32 bits R_i to produce a 32-bit output. This function is made up of four operations: an XOR, an expansion permutation, a group of S-boxes, and a straight permutation, as shown in Figure 30.15.

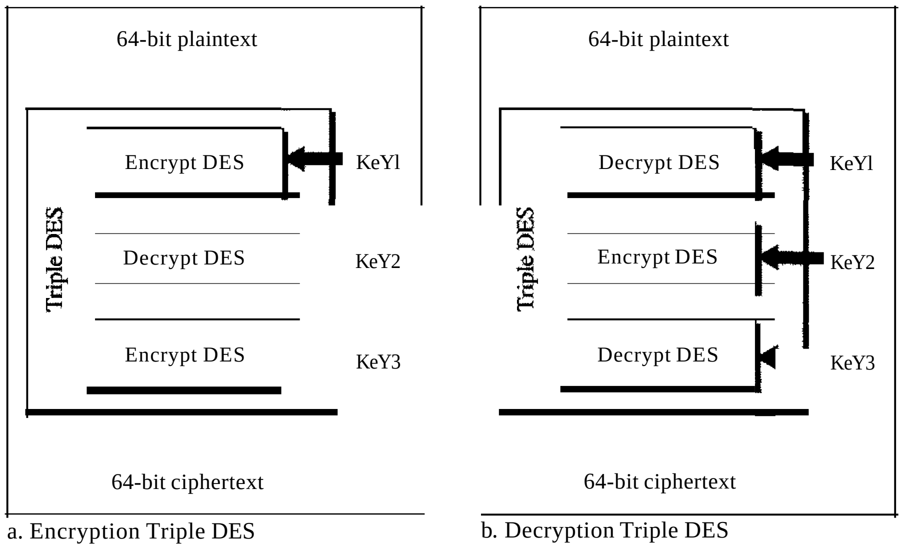
Figure 30.15 *DES function*



Triple DES

Critics of DES contend that the key is too short. To lengthen the key, Triple DES or 3DES has been proposed and implemented. This uses three DES blocks, as shown in Figure 30.16. Note that the encrypting block uses an encryption-decryption-encryption combination of DESs, while the decryption block uses a decryption-encryption-decryption combination. Two different versions of 3DES are in use: 3DES with two keys and 3DES with three keys. To make the key size 112 bits and at the same time protect DES from attacks such as the man-in-the-middle attack, 3DES with two keys was designed. In this version, the first and the third keys are the same (KeYl = KeY3)' This has the advantage in that a text encrypted by a single DES block can be decrypted by the new 3DES. We just set all keys equal to KeYl' Many algorithms use a 3DES cipher with three keys. This increases the size of the key to 168 bits.

Figure 30.16 *Triple DES*



Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES) was designed because DES's key was too small. Although Triple DES ODES) increased the key size, the process was too slow. The National Institute of Standards and Technology (NIST) chose the Rijndael algorithm, named after its two Belgian inventors, Vincent Rijmen and Joan Daemen, as the basis of AES. AES is a very complex round cipher. AES is designed with three key sizes: 128, 192, or 256 bits. Table 30.1 shows the relationship between the data block, number of rounds, and key size.

Table 30.1 *AES configuration*

Size ofData Block	*Number ofRounds*	*Key Size*
128 bits	10	128 bits
	12	192 bits
	14	256 bits

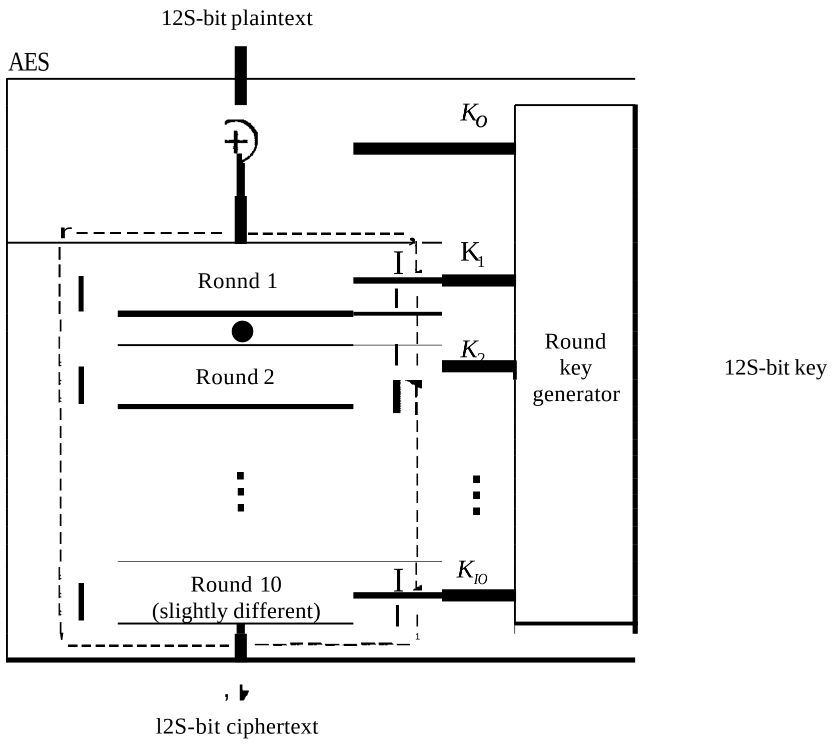
AES has three different configurations with respect
to the number of rounds and key size.

In this text, we discuss just the lO-round, 12S-bit key configuration. The structure and operation of the other configurations are similar. The difference lies in the key generation.

The general structure is shown in Figure 30.17. There is an initial XOR operation followed by 10 round ciphers. The last round is slightly different from the preceding rounds; it is missing one operation.

Although the 10 iteration blocks are almost identical, each uses a different key derived from the original key.

Figure 30.17 *AES*

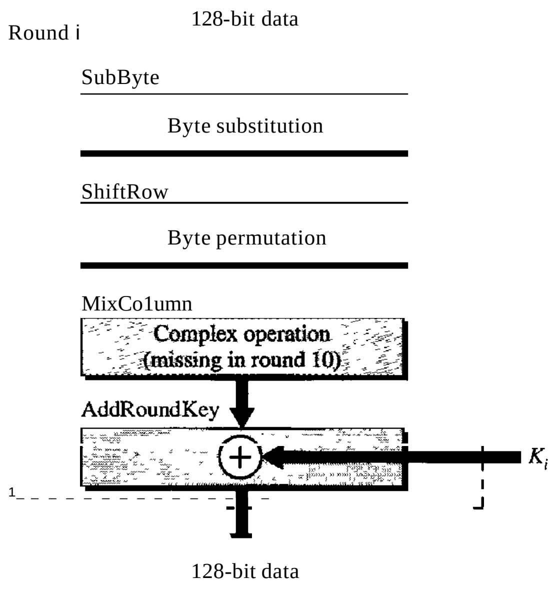


Structure of Each Round Each round of AES, except for the last, is a cipher with four operations that are invertible. The last round has only three operations. Figure 30.18 is a flowchart that shows the operations in each round. Each of the four operations used in each round uses a complex cipher; this topic is beyond the scope of this book.

Other Ciphers

During the last two decades, a few other symmetric block ciphers have been designed and used. Most of these ciphers have similar characteristics to the two ciphers we discussed in this chapter (DES and AES). The difference is usually in the size of the block or key, the number of rounds, and the functions used. The principles are the same. In order not to burden the user with the details of these ciphers, we give a brief description of each.

Figure 30.18 *Structure ofeach round*



IDEA The International Data Encryption Algorithm (IDEA) was developed by Xuejia Lai and James Massey. The block size is 64 and the key size is 128. It can be implemented in both hardware and software.

Blowfish Blowfish was developed by Bruce Schneier. The block size is 64 and the key size between 32 and 448.

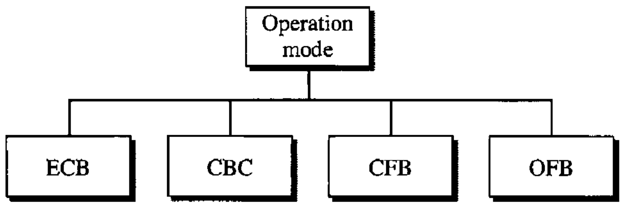
CAST-128 CAST-128 was developed by Carlisle Adams and Stafford Tavares. It is a Feistel cipher with 16 rounds and a block size of 64 bits; the key size is 128 bits.

ReS RCS was designed by Ron Rivest. It is a family of ciphers with different block sizes, key sizes, and numbers of rounds.

Mode of Operation

A mode of operation is a technique that employs the modern block ciphers such as DES andAES that we discussed earlier (see Figure 30.19).

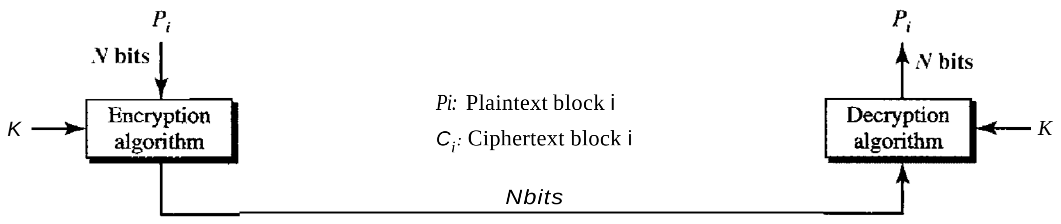
Figure 30.19 *Modes ofoperation for block ciphers*



Electronic Code Book

The electronic code book (ECB) mode is a purely block cipher technique. The plaintext is divided into blocks of *N* bits. The ciphertext is made of blocks of *N* bits. The value of *N* depends on the type of cipher used. Figure 30.20 shows the method.

Figure 30.20 *ECB mode*



We mention four characteristics of this mode:

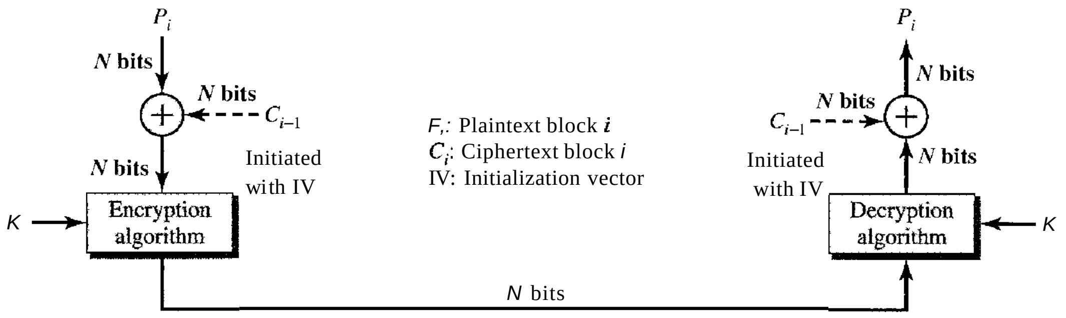
1. Because the key and the encryption/decryption algorithm are the same, equal blocks in the plaintext become equal blocks in the ciphertext. For example, if plaintext blocks 1, 5, and 9 are the same, ciphertext blocks I, 5, and 9 are also the same. This can be a security problem; the adversary can guess that the plaintext blocks are the same if the corresponding ciphertext blocks are the same.
2. If we reorder the plaintext block, the ciphertext is also reordered.
3. Blocks are independent of each other. Each block is encrypted or decrypted independently. A problem in encryption or decryption of a block does not affect other blocks.
4. An error in one block is not propagated to other blocks. If one or more bits are corrupted during transmission, it only affects the bits in the corresponding plaintext after decryption. Other plaintext blocks are not affected. This is a real advantage if the channel is not noise-free.

Cipher Block Chaining

The cipher block chaining (CBC) mode tries to alleviate some of the problems in ECB by including the previous cipher block in the preparation of the current block. If the current block is *i*, the previous ciphertext block C_{i-1} is included in the encryption of block *i*. In other words, when a block is completely enciphered, the block is sent, but a copy of it is kept in a register (a place where data can be held) to be used in the encryption of the next block. The reader may wonder about the initial block. There is no ciphertext block before the first block. In this case, a phony block called the initiation vector (IV) is used. Both the sender and receiver agree upon a specific predetermined IV. In other words, the IV is used instead of the nonexistent CO, Figure 30.21 shows the CBC mode.

The reader may wonder about the decryption. Does the configuration shown in the figure guarantee the correct decryption? It can be proven that it does, but we leave the proof to a textbook in network security.

Figure 30.21 *CBC mode*



The following are some characteristics of CBC.

1. Even though the key and the encryption/decryption algorithm are the same, equal blocks in the plaintext do not become equal blocks in the ciphertext. For example, if plaintext blocks 1, 5, and 9 are the same, ciphertext blocks I, 5, and 9 will not be the same. An adversary will not be able to guess from the ciphertext that two blocks are the same.
2. Blocks are dependent on each other. Each block is encrypted or decrypted based on a previous block. A problem in encryption or decryption of a block affects other blocks.
3. The error in one block is propagated to the other blocks. If one or more bits are corrupted during the transmission, it affects the bits in the next blocks of the plaintext after decryption.

Cipher Feedback

The cipher feedback (CFB) mode was created for those situations in which we need to send or receive *r* bits of data, where *r* is a number different from the underlying block size of the encryption cipher used. The value of *r* can be 1, 4, 8, or any number of bits. Since all block ciphers work on a block of data at a time, the problem is how to encrypt just *r* bits. The solution is to let the cipher encrypt a block of bits and use only the first *r* bits as a new key (stream key) to encrypt the *r* bits of user data. Figure 30.22 shows the configuration.

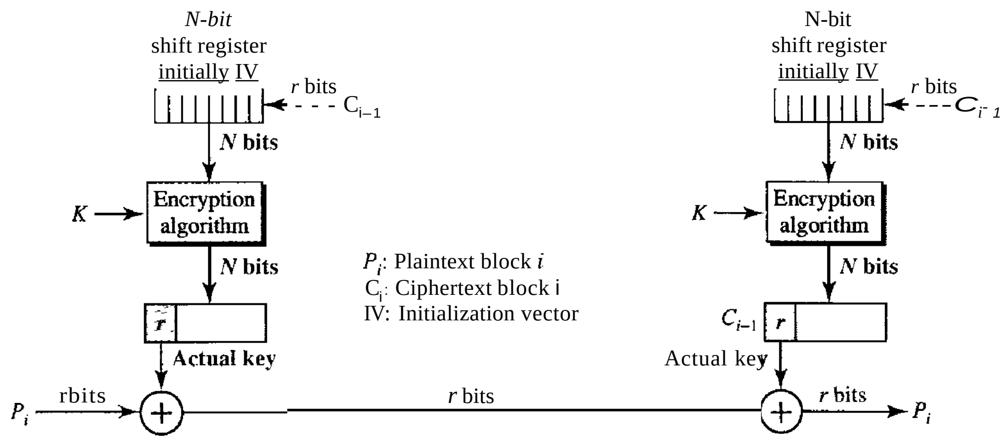
The following are some characteristics of the CFB mode:

1. If we change the IV from one encryption to another using the same plaintext, the ciphertext is different.
2. The ciphertext C_i depends on both *Pi* and the preceding ciphertext block.
3. Errors in one or more bits of the ciphertext block affect the next ciphertext blocks.

Output Feedback

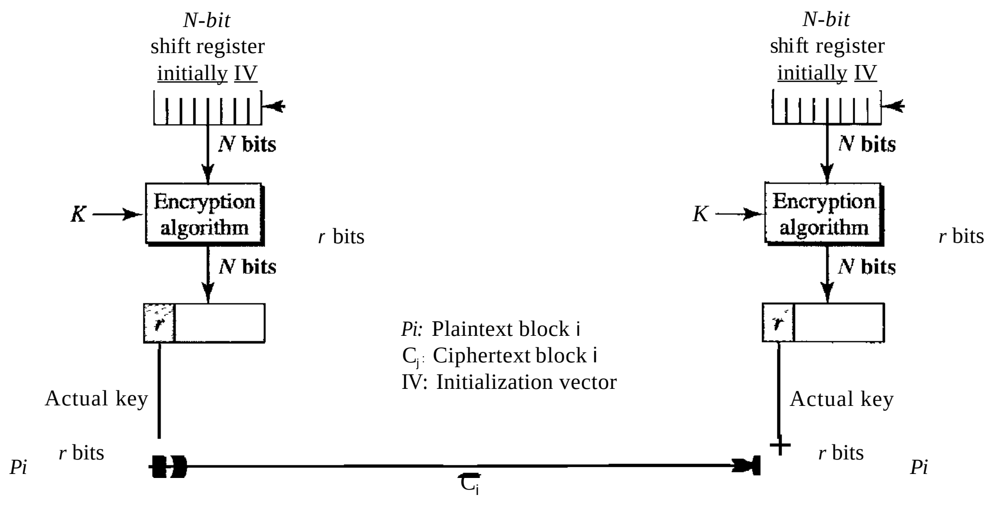
The **output** feedback (OFB) mode is very similar to the CFB mode with one difference. Each bit in the ciphertext is independent of the previous bit or bits. This avoids error

Figure 30.22 *CFB mode*



propagation. If an error occurs in transmission, it does not affect the future bits. Note that, as in CFB, both the sender and the receiver use the encryption algorithm. Note also that in OFB, block ciphers such as DES or AES can only be used to create the key stream. The feedback for creating the next bit stream comes from the previous bits of the key stream instead of the ciphertext. The ciphertext does not take part in creating the key stream. Figure 30.23 shows the OFB mode.

Figure 30.23 *OFB mode*



The following are some of the characteristics of the OFB mode.

1. If we change the IV from one encryption to another using the same plaintext, the ciphertext will be different.
2. The ciphertext C_i depends on the plaintext *Pi'*
3. Errors in one or more bits of the ciphertext do not affect future ciphertext blocks.

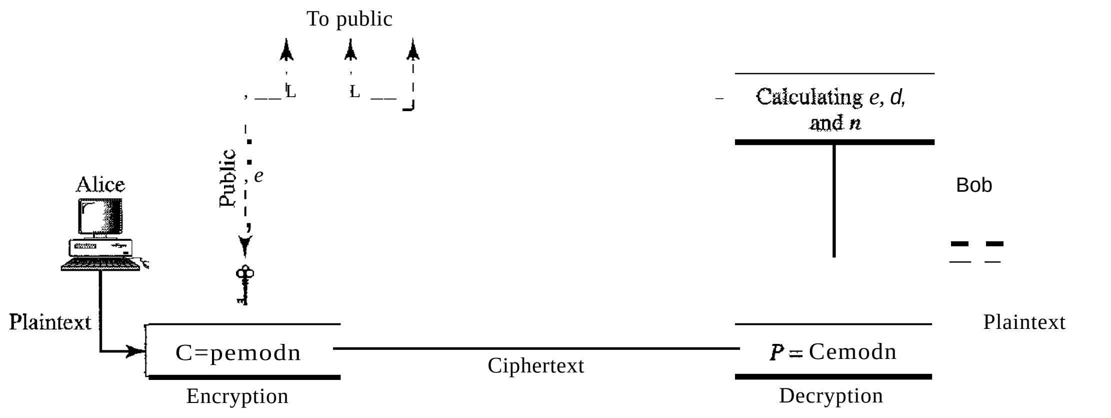
30.3 ASYMMETRIC-KEY CRYPTOGRAPHY

In the previous sections, we discussed symmetric-key cryptography. In this section we introduce asymmetric-key (public key cryptography). As we mentioned before, an asymmetric-key (or public-key) cipher uses two keys: one private and one public. We discuss two algorithms: RSA and Diffie-Hellman.

RSA

The most common public key algorithm is RSA, named for its inventors Rivest, Shamir, and Adleman (RSA). It uses two numbers, e and d, as the public and private keys, as shown in Figure 30.24.

Figure 30.24 *RSA*



The two keys, *e* and *d*, have a special relationship to each other, a discussion of this relationship is beyond the scope of this book. We just show how to calculate the keys without proof.

Selecting Keys

Bob use the following steps to select the private and public keys:

1. Bob chooses two very large prime numbers *p* and *q*. Remember that a prime number is one that can be divided evenly only by 1 and itself.
2. Bob multiplies the above two primes to find *n*, the modulus for encryption and decryption. In other words, $n ::: p \times q$.
3. Bob calculates another number $\phi ::: (p - 1) \times (q - 1)$.
4. Bob chooses a random integer *e*. He then calculates *d* so that $d \times e ::: 1 \bmod \phi$.
5. Bob announces *e* and *n* to the public; he keeps ϕ and *d* secret.

In RSA, *e* and *n* are announced to the public; *d* and ϕ are kept secret.

Encryption

Anyone who needs to send a message to Bob can use n and e. For example, if Alice needs to send a message to Bob, she can change the message, usually a short one, to an integer. This is the plaintext. She then calculates the ciphertext, using e and n.

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

Alice sends C, the ciphertext, to Bob.

Decryption

Bob keeps ϕ and d private. When he receives the ciphertext, he uses his private key d to decrypt the message:

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

Restriction

For RSA to work, the value of P must be less than the value of n. If P is a large number, the plaintext needs to be divided into blocks to make P less than n.

Example 30.7

Bob chooses 7 and 11 as p and q and calculates $n = 7 \cdot 11 = 77$. The value of $\phi = (7 - 1)(11 - 1)$ or 60. Now he chooses two keys, e and d. If he chooses e to be 13, then d is 37. Now imagine Alice sends the plaintext 5 to Bob. She uses the public key 13 to encrypt 5.

Plaintext: 5
$C = 5^{13} = 26 \bmod 77$
Ciphertext: 26

Bob receives the ciphertext 26 and uses the private key 37 to decipher the ciphertext:

Ciphertext: 26
$P = 26^{37} = 5 \bmod 77$
Plaintext: 5 Intended message sent by Alice

The plaintext 5 sent by Alice is received as plaintext 5 by Bob.

Example 30.8

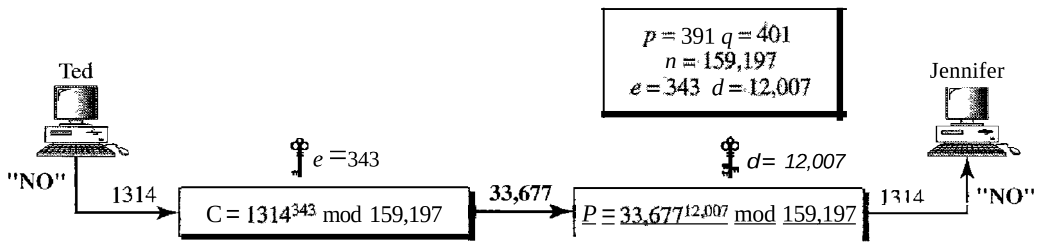
Jennifer creates a pair of keys for herself. She chooses $p = 397$ and $q = 401$. She calculates $n = 159{,}197$ and $\phi = 396 \cdot 400 = 158{,}400$. She then chooses $e = 343$ and $d = 12{,}007$. Show how Ted can send a message to Jennifer if he knows e and n.

Solution

Suppose Ted wants to send the message "NO" to Jennifer. He changes each character to a number (from 00 to 25) with each character coded as two digits. He then concatenates the two coded characters and gets a four-digit number. The plaintext is 1314. Ted then uses e and n to encrypt the message. The ciphertext is $1314^{343} = 33{,}677 \bmod 159{,}197$. Jennifer receives the message

33,677 and uses the decryption key *d* to decipher it as $33{,}677^{12{,}007} = 1314 \bmod 159{,}197$. Jennifer then decodes 1314 as the message "NO". Figure 30.25 shows the process.

Figure 30.25 *Example 30.8*



Example 30.9

Let us give a realistic example. We choose a 512-bit *p* and *q*. We calculate *n* and ϕ. We then choose *e* and test for relative primeness with $\phi(n)$. We calculate *d*. Finally, we show the results of encryption and decryption. We have written a program written in Java to do so; this type of calculation cannot be done by a calculator.

We randomly chose an integer of 512 bits. The integer *p* is a 159-digit number.

p=96130345313583504574191581280615427909309845594996215822583150879647940
45505647063849125716018034750312098666606492420191808780667421096063354
219926661209

The integer *q* is a 160-digitnumber.

q= 12060191957231446918276794204450896001555925054637033936061798321731482
14848376465921538945320917522527322683010712069560460251388714552496900
0359660045617

We calculate *n*. **It** has 309 digits.

n=11593504I73967614968892509864615887523771457375454144775485526137614788
54083263508172768788159683251684688493006254857641112501624145523391829
27162507656772727460097082714127730434960500556347274566628060099924037
10299142447229221577279853172703383938133469268413732762200096667667183
1831088373420823444370953

We calculate ϕ. **It** has 309 digits:

ϕ = 11593504173967614968892509864615887523771457375454144775485526137614788
54083263508172768788159683251684688493006254857641112501624145523391829
27162507656751054233608492916752034482627988117554787657013923444405716
98958172819609822636107546721186461217135910735864061400888517026537727
1264467341066243857664128

We choose e =35,535. We then find d.

e = 35535

d=58oo8302860037763936093661289677917594669062089650962180422866111380593852
82235873170628691003002171085904433840217072986908760061153062025249598844
48047568240966247081485817130463240644077704833134010850947385295645071936
77406119732655742423721761767462077637164207600337085333288532144708859551
36670294831

Alice wants to send the message "THIS IS A TEST" which can be changed to a numeric value by using the 00-26 encoding scheme (26 is the *space* character).

P =1907081826081826002619041819

The ciphertext calculated by Alice is C = p^e, which is

C = 4753091236462268272063655506105451809423717960704917165232392430544529
60613199328566617843418359114151197411252005682979794571736036101278:21'
8847892741566090480023507190715277185914975188465888632101148354103361
6578984679683867637337657774656250792805211481418440481418443081277305",
9004692874248559166462108656

Bob can recover the plaintext from the ciphertext by using P = Cd, which is

P = 1907081826081826002619041819

The recovered plaintext is THIS IS A TEST after decoding.

Applications

Although RSA can be used to encrypt and decrypt actual messages, it is very slow if the message is long. RSA, therefore, is useful for short messages such as a small message digest (see Chapter 31) or a symmetric key to be used for a symmetric-key cryptosystem. In particular, we will see that RSA is used in digital signatures and other cryptosystems that often need to encrypt a small message without having access to a symmetric key. RSA is also used for authentication as we will see later.

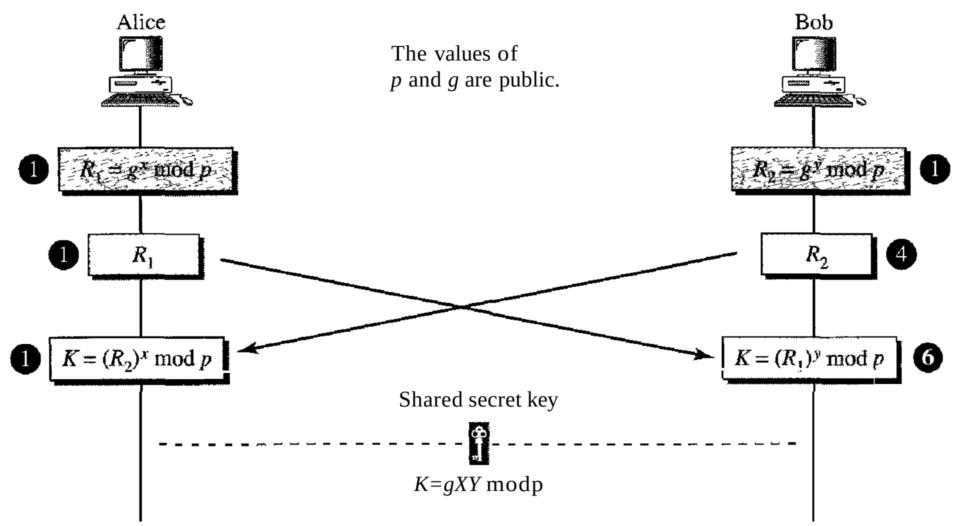
Diffie-Hellman

RSA is a public-key cryptosystem that is often used to encrypt and decrypt symmetric keys. Diffie-Hellman, on the other hand, was originally designed for key exchange. In the **Diffie-Hellman** cryptosystem, two parties create a symmetric session key to exchange data without having to remember or store the key for future use. They do not have to meet to agree on the key; it can be done through the Internet. Let us see how the protocol works when Alice and Bob need a symmetric key to communicate. Before establishing a symmetric key, the two parties need to choose two numbers p and g. The first number, p, is a large prime number on the order of 300 decimal digits (1024 bits). The second number is a random number. These two numbers need not be confidential. They can be sent through the Internet; they can be public.

Procedure

Figure 30.26 shows the procedure. The steps are as follows:

Figure 30.26 *Diffie-Hellman method*



O Step 1: Alice chooses a large random number x and calculates $R1 = g^x$ mod p.

O Step 2: Bob chooses another large random number y and calculates $R_2 = gY$ mod p.

O Step 3: Alice sends $R1$ to Bob. Note that Alice does not send the value of x; she sends only $R1$-

O Step 4: Bob sends R_2 to Alice. Again, note that Bob does not send the value of y, he sends only R_2.

O Step 5: Alice calculates $K = (R_2)^x$ mod p.

O Step 6: Bob also calculates $K = (R_1)^y$ mod p.

The symmetric key for the session is K.

$$(g^x \bmod p)^y \bmod p = (gY \bmod p)X \bmod p = g^{xy} \bmod p$$

Bob has calculated $K = (R_1)^y \bmod p = (g^x \bmod p)^y \bmod p = g^{xy} \bmod p$. Alice has calculated $K = (R_2)X \bmod p = (gY \bmod p)X \bmod = g^{xy} \bmod p$. Both have reached the same value without Bob knowing the value of x and without Alice knowing the value of y.

The symmetric (shared) key in the Diffie-Hellman protocol is
$\boldsymbol{K = g^{xy} \bmod p}$.

Example 30.10

Let us give a trivial example to make the procedure clear. Our example uses small numbers, but note that in a real situation, the numbers are very large. Assume $g = 7$ and $p = 23$. The steps are as follows:

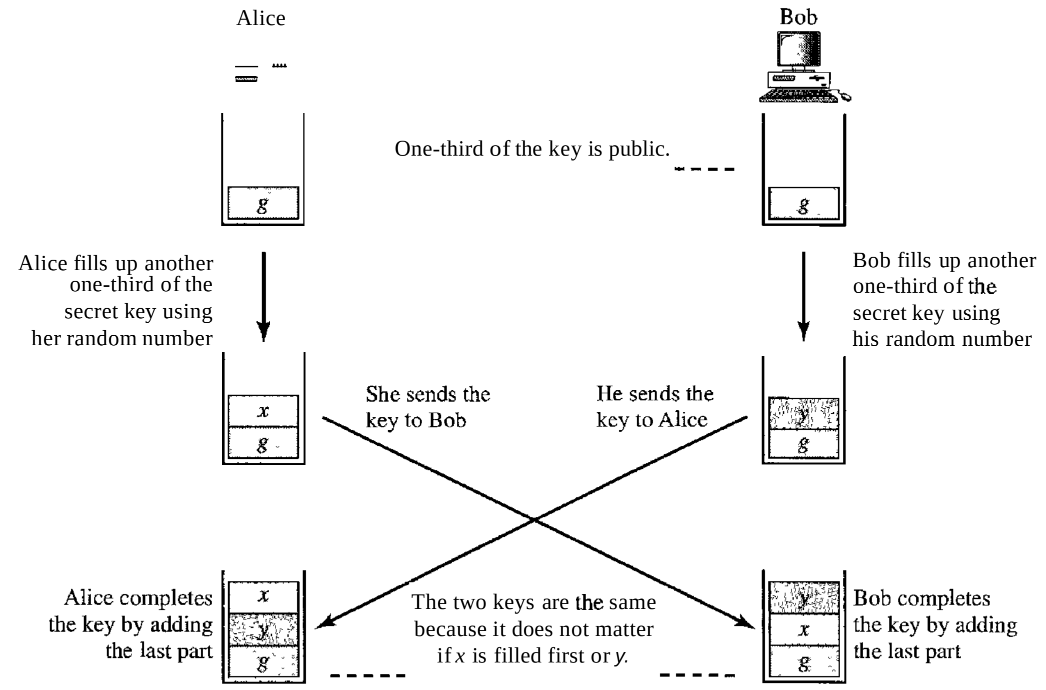
1. Alice chooses $x = 3$ and calculates $R_1 = 7^3 \bmod 23 = 21$.
2. Bob chooses $y = 6$ and calculates $R_2 = 7^6 \bmod 23 = 4$.
3. Alice sends the number 21 to Bob.
4. Bob sends the number 4 to Alice.
5. Alice calculates the symmetric key $K = 4^3 \bmod 23 = 18$.
6. Bob calculates the symmetric key $K = 21^6 \bmod 23 = 18$.

The value of K is the same for both Alice and Bob; $g^{xy} \bmod p = 7^{18} \bmod 23 = 18$.

Idea of Diffie-Hellman

The Diffie-Hellman concept, shown in Figure 30.27, is simple but elegant. We can think of the secret key between Alice and Bob as made of three parts: g, x, and y. The first part is public. Everyone knows one-third of the key; g is a public value. The other two parts must be added by Alice and Bob. Each adds one part. Alice adds x as the second part for Bob; Bob adds y as the second part for Alice. When Alice receives the two-thirds completed key from Bob, she adds the last part, her x, to complete the key. When Bob receives the two-thirds completed key from Alice, he adds the last part, his y, to complete the key. Note that although the key in Alice's hand consists of g-y-x and the key in Bob's hand is g-x-y, these two keys are the same because $g^{xy} = g^{yx}$.

Figure 30.27 *Diffie-Hellman idea*


Alice
Bob
One-third of the key is public.
g
g
Alice fills up another one-third of the secret key using her random number
Bob fills up another one-third of the secret key using his random number
x
g
She sends the key to Bob
He sends the key to Alice
y
g
Alice completes the key by adding the last part
x
y
g
The two keys are the same because it does not matter if x is filled first or y.
y
x
g
Bob completes the key by adding the last part

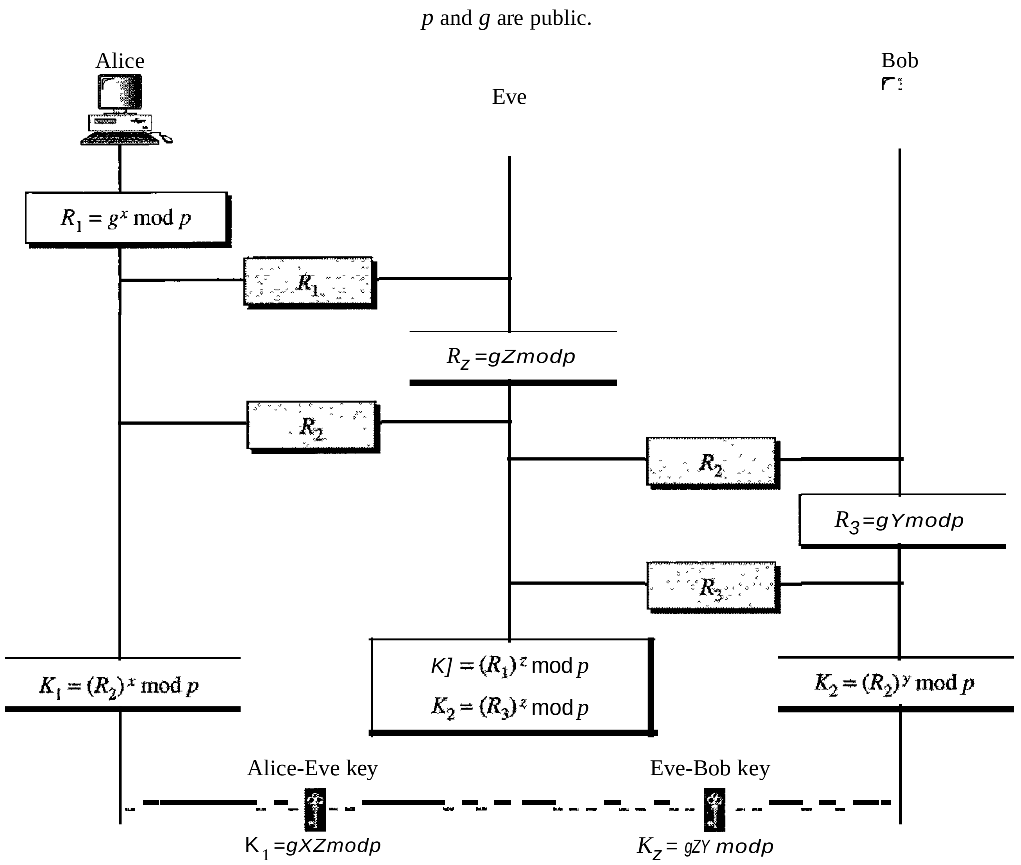
Note also that although the two keys are the same, Alice cannot find the value y used by Bob because the calculation is done in modulo p; Alice receives $g^y \bmod p$ from Bob, not g^y.

Man-in-the-Middle Attack

Diffie-Hellman is a very sophisticated symmetric-key creation algorithm. If x and y are very large numbers, it is extremely difficult for Eve to find the key, knowing only p and g. An intruder needs to determine x and y if R_1 and R_2 are intercepted. But finding x from R_1 and y from R_2 are two difficult tasks. Even a sophisticated computer would need perhaps years to find the key by trying different numbers. In addition, Alice and Bob will change the key the next time they need to communicate.

However, the protocol does have a weakness. Eve does not have to find the value of x and y to attack the protocol. She can fool Alice and Bob by creating two keys: one between herself and Alice and another between herself and Bob. Figure 30.28 shows the situation.

Figure 30.28 *Man-in-the-middle attack*



The following can happen:

1. Alice chooses x, calculates $R_1 = g^x \bmod p$, and sends R_1 to Bob.
2. Eve, the intruder, intercepts R_1. She chooses z, calculates $R_2 = g^z \bmod p$, and sends R_2 to both Alice and Bob.

3. Bob chooses y, calculates $R_3 = g^y \bmod p$, and sends R_3 to Alice; R_3 is intercepted by Eve and never reaches Alice.
4. Alice and Eve calculate $K_1 = g^{xz} \bmod p$, which becomes a shared key between Alice and Eve. Alice, however, thinks that it is a key shared between Bob and herself.
5. Eve and Bob calculate $K_2 = g^{zy} \bmod p$, which becomes a shared key between Eve and Bob. Bob, however, thinks that it is a key shared between Alice and himself.

In other words, two keys, instead of one, are created: one between Alice and Eve and one between Eve and Bob. When Alice sends data to Bob encrypted with K_1 (shared by Alice and Eve), it can be deciphered and read by Eve. Eve can send the message to Bob encrypted by K_2 (shared key between Eve and Bob); or she can even change the message or send a totally new message. Bob is fooled into believing that the message has come from Alice. A similar scenario can happen to Alice in the other direction.

This situation is called a man-in-the-middle attack because Eve comes in between and intercepts R_1, sent by Alice to Bob, and R_3, sent by Bob to Alice. It is also known as a bucket brigade attack because it resembles a short line of volunteers passing a bucket of water from person to person.

Authentication

The man-in-the-middle attack can be avoided if Bob and Alice first authenticate each other. In other words, the exchange key process can be combined with an authentication scheme to prevent a man-in-the-middle attack. We discuss authentication in Chapter 31.

30.4 RECOMMENDED READING

For more details about subjects discussed in this chapter, we recommend the following books. The items in brackets [...] refer to the reference list at the end of the text.

Books

Cryptography can be found in many books dedicated to the subject such as [Bar02], [GartH], [Sti02], [Mao04], [MOV97], and [Sch96].

30.5 KEY TERMS

Advanced Encryption Standard (AES)	cryptography
block cipher	Data Encryption Standard (DES)
bucket brigade attack	decryption
Caesar cipher	decryption algorithm
cipher block chaining (CBC) mode	DES function
cipher feedback (CFB) mode	Diffie-Hellman cryptosystem
ciphertext	electronic code book (ECB) mode
compression permutation	encryption