UNIT-1(C&NS)

Computer Security-generic name for the collection of tools designed to protect data and to thwart hackers

Network Security-measures to protect data during their transmission.This area covers the use of cryptographic algorithms in network protocols and network applications.

Cryptographic algorithms: This is the study of techniques for ensuring the secrecy and/or authenticity of information

SECURITY GOALS:

CONFEDENTIALITY:

- hiding information from an authorized access
- information while exchange should remain secret

DATA INTEGRITY:

- preventing information from un authorized modification
- need techniques to ensure the integrity of the data
    - preventing the modification
    - detect any modification made

AVAILABILITY:

- should be easily available to authorized users
- data must be available to authorized users

cryptographic algorithms are used to achieve the above goals

THE OSI SECURITY ARCHITECTURE

The OSI security architecture focuses on security attacks, mechanisms, and services. These can be defined briefly as

• **Security attack:** Any action that compromises the security of information owned by an organization.

• **Security mechanism:** A process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack.

• **Security service:** A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks, and they make use of one or more security mechanisms to provide the service.

SECURITY ATTACKS

Generic types of attacks

- Passive attacks
- Active attacks

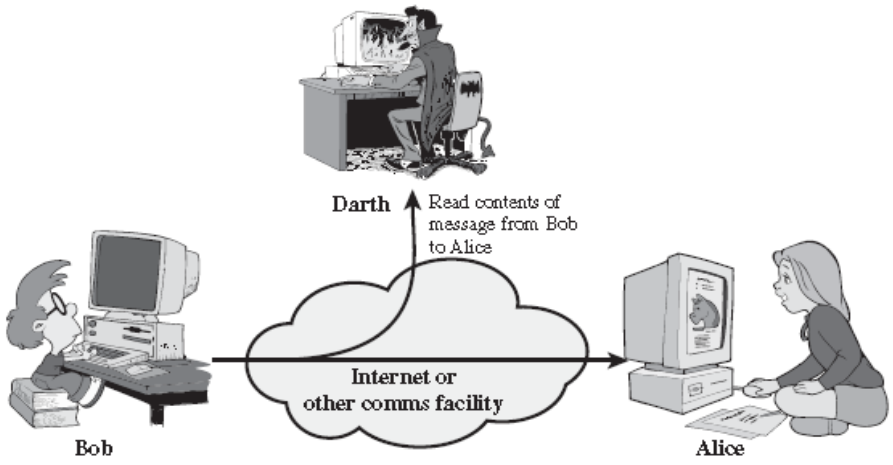
. A passive attack attempts to learn or make use of information from the system but does not affect system resources. An active attack attempts to alter system resources or affect their operation.

Passive Attacks

Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Two types of passive attacks are the release of message contents and traffic analysis.

1) **Release of message contents:**

The **release of message contents** is easily understood .A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information.We would like to prevent an opponent from learning the contents of these transmissions.

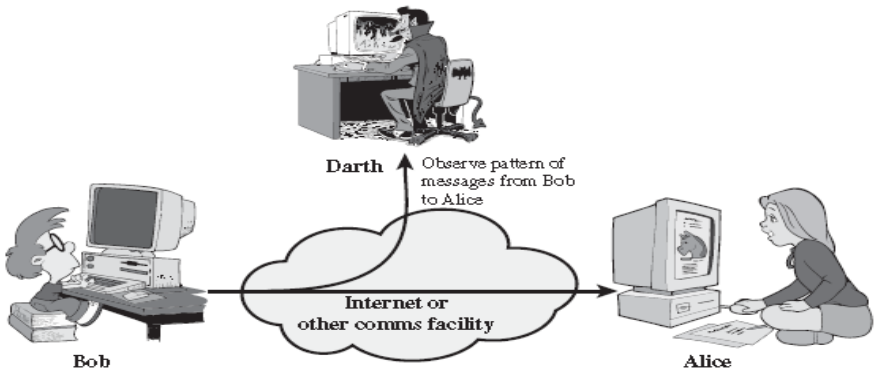


(a) Release of message contents

2) **Traffic analysis:**

A second type of passive attack, **traffic analysis**, is subtler .Suppose that we had a way of masking the contents of messages or otherinformation traffic so that opponents, even if they captured the message, couldnot extract the information from the message. The common technique formasking contents is encryption. If we had encryption protection in place, anopponent might still be able to observe the pattern of these messages. Theopponent could determine the location and identity of communicating hosts andcould observe the frequency and length of messages being exchanged. Thisinformation might be useful in guessing the nature of the communication thatwas taking place.

Passive attacks are very difficult to detect, because they do not involve anyalteration of the data.



(b) Traffic analysis

Active attack: An active attack attempts to alter system resources or affect their operation. Active attacks involve some modification of the data stream or the creation of a false stream.

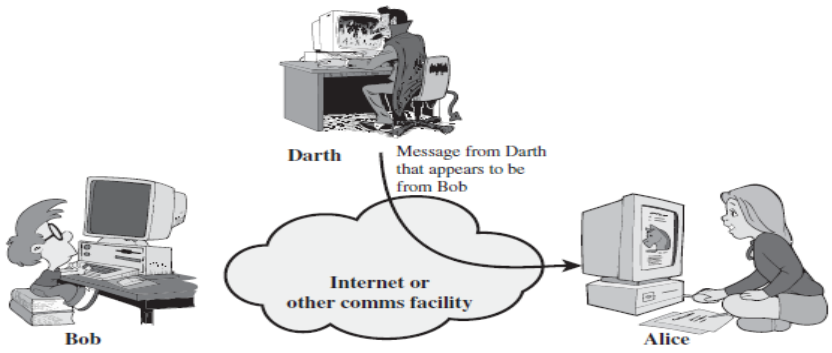
Active attacks can be subdivided into four categories:

- masquerade,
- replay,
- modification of messages, and
- Denial of service.

Masquerade:

A **masquerade** takes place when one entity pretends to be a different entity (Figure:). A masquerade attack usually includes one of the other forms of active attack.

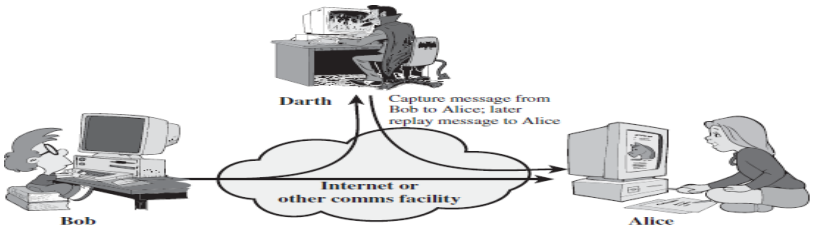
For example, authentication sequences can be captured and replayed after a valid authentication sequence has taken place, thus enabling an authorized entity with few privileges to obtain extra privileges by impersonating an entity that has those privileges.



(a) Masquerade

Replay :

Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.

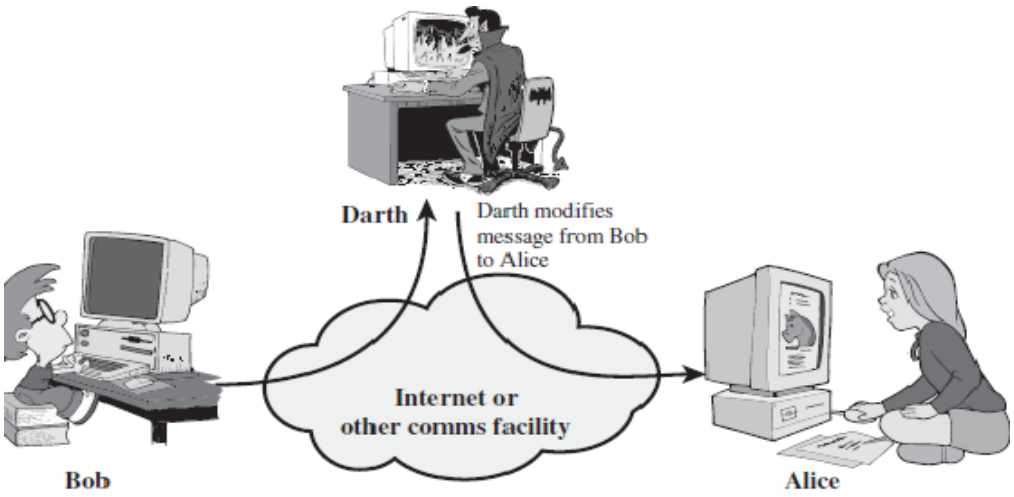


(b) Replay

Modification of messages:

Modification of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect (Figure: c).

For example, a message meaning "Allow John Smith to read confidential file accounts" is modified to mean "Allow Fred Brown to read confidential file accounts



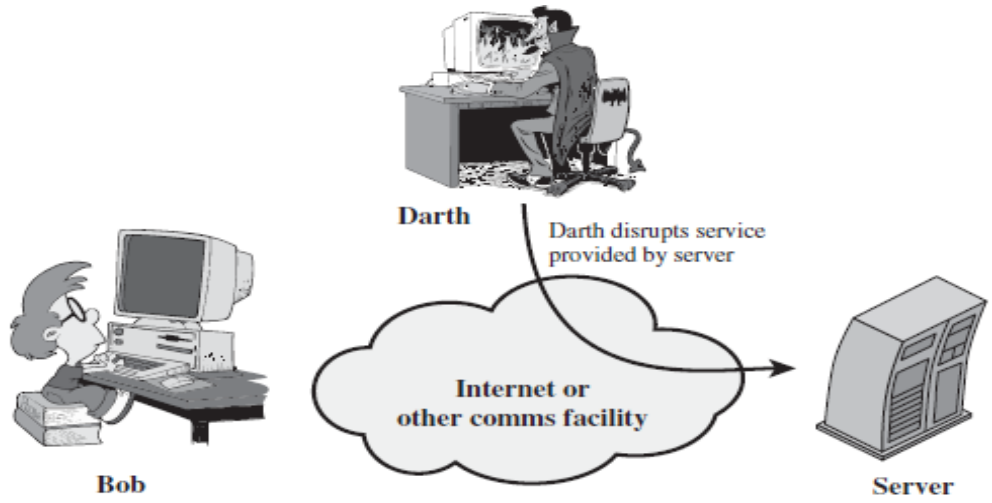
(c) Modification of messages

Denial of service:

The **denial of service** prevents or inhibits the normal use or management of communications facilities (Figure d). This attack may have a specific target;

For example, an entity may suppress all messages directed to a particular destination (e.g., the security audit service).

Another form of service denial is the disruption of an entire network—either by disabling the network or by overloading it with messages so as to degrade performance



(d) Denial of service

1.7 SECURITY SERVICES

The classification of security services are as follows:

i)CONFIDENTIALITY: Ensures that the information in a computer system and transmittedinformation are accessible only for reading by authorized parties. Confidentiality is the protection of transmitted data from passive attacks. For example, when a TCP connection is set up between two systems, this broad protection prevents the release of any user data transmitted over the TCP connection.

Connection Confidentiality

The protection of all user data on a connection.

Connectionless Confidentiality

The protection of all user data in a single data block

Selective-Field Confidentiality

The confidentiality of selected fields within the user data on a connection or in a single data block.

Traffic-Flow Confidentiality

The protection of the information that might be derived from observation of traffic flows.

ii)AUTHENTICATION: The authentication service is concerned with assuring that a communication is Authentic. The assurance that the communicating entity is the one that it claims to be.

Ensures that the origin of a message or electronic document is correctly identified, with an assurance that the identity is not false.

Peer Entity Authentication

Used in association with a logical connection to provide confidence in the identity of the entities connected.

Data-Origin Authentication

In a connectionless transfer, provides assurance that the source of received data is as claimed.

iii)INTEGRITY: Ensures that only authorized parties are able to modify computer system assets and

transmitted information. Modification includes writing, changing status, deleting, creating and delaying or replaying of transmitted messages.

iv)NON REPUDIATION: Requires that neither the sender nor the receiver of a message be able to deny the transmission. when a message is sent, the receiver can prove that the alleged sender in

fact sent the message. Similarly, when a message is received, the sender can prove that the alleged receiver in fact received the message

v)ACCESS CONTROL: Requires that access to information resources may be controlled by the target system . access control is the ability to limit and control the access to host systems and applications via communications links. To achieve this, each entity trying to gain access must first be identified, or authenticated

vi)AVAILABILITY: Requires that computer system assets be available to authorized parties when needed

SECURITY MECHANISMS

One of the most specific security mechanisms in use is cryptographic techniques.

Encryption or encryption-like transformations of information are the most common means of providing security. Some of the mechanisms are

1 ENCIPHERMENT

2 DIGITAL SIGNATURE

3 ACCESS CONTROL

ENCIPHERMENT: It refers to the process of applying mathematical algorithms for converting data into a form that is not intelligible. This depends on algorithm used and encryption keys.

DIGITAL SIGNATURE: The appended data or a cryptographic transformation applied to any data unit allowing to prove the source and integrity of the data unit and protect against forgery.

ACCESS CONTROL: A variety of techniques used for enforcing access permissions to the system resources.

DATA INTEGRITY: A variety of mechanisms used to assure the integrity of a data unit or stream of data units.

AUTHENTICATION EXCHANGE: A mechanism intended to ensure the identity of an entity by means of information exchange.

TRAFFIC PADDING: The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.

ROUTING CONTROL: Enables selection of particular physically secure routes for certain data and allows routing changes once a breach of security is suspected.

NOTARIZATION: The use of a trusted third party to assure certain properties of a data exchange

GENERAL TERMS:

An original message is known as the **plaintext**, while the coded message is called the **ciphertext**.The process of converting from plaintext to ciphertext is known as **enciphering** or **encryption**; restoring the plaintext from the ciphertext is **deciphering** or **decryption**. The many schemes used for encryption constitute the area of study known as **cryptography**. Such a scheme is known as a **cryptographic system** or a **cipher**. Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of **cryptanalysis**. Cryptanalysis is what the layperson calls "breaking the code."The areas of cryptography and cryptanalysis together are called **cryptology**.

SYMMETRIC CIPHER MODEL:

Symmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the same key. It is also known as conventional encryption. Symmetric encryption, also referred to as conventional encryption or single-key encryption.

A symmetric encryption scheme has five ingredients

• **Plaintext:** This is the original intelligible message or data that is fed into the algorithm as input.

• **Encryption algorithm:** The encryption algorithm performs various substitutions and transformations on the plaintext.

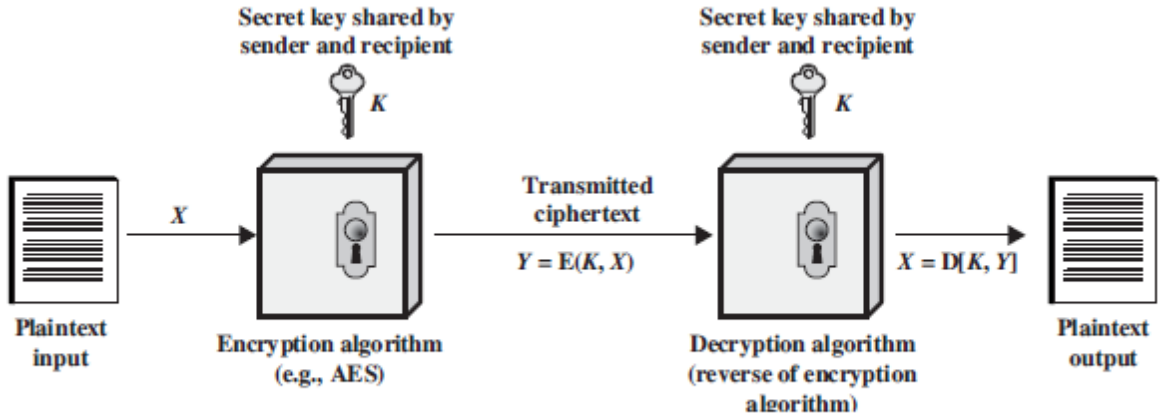
• **Secret key:** The secret key is also input to the encryption algorithm.The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time.The exact substitutions and transformations performed by the algorithm
depend on the key.

• **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts. • **Decryption algorithm:** This is essentially the encryption algorithm run inreverse. It takes the ciphertext and the secret key and produces the originalplaintext.

There are two requirements for secure use of conventional encryption:

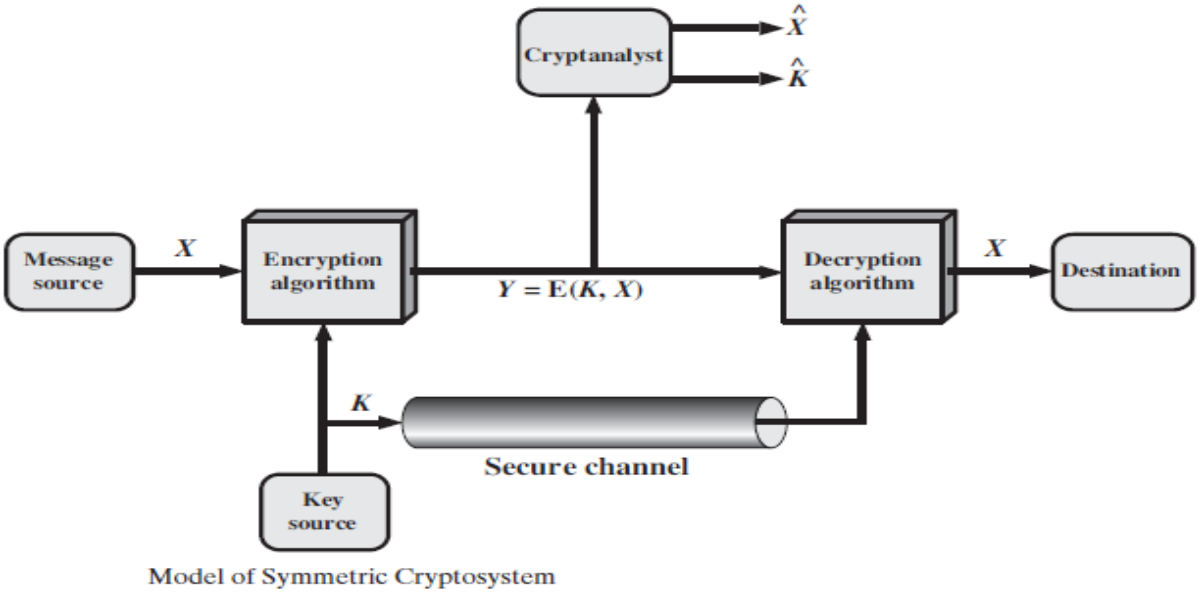
1. We need a strong encryption algorithm. At a minimum, we would like thealgorithm to be such that an opponent who knows the algorithm and hasaccess to one or more ciphertexts would be unable to decipher the ciphertextor figure out the key.

2.Sender and receiver must have obtained copies of the secret key in a securefashion and must keep the key secure.



Simplified Model of Symmetric Encryption

Let us take a closer look at the essential elements of a symmetric encryption scheme, using below Figure. A source produces a message in plaintext, . The elements of are letters in some finite alphabet. Traditionally, the alphabet usually consisted of the 26 capital letters. Nowadays,the binary alphabet {0, 1} is typically used. For encryption, a key of the form is generated. If the key is generated at the message source, then it must also be provided to the destination by means of some secure channel. Alternatively, a third party could generate the key and securely deliver it toboth source and destination



Model of Symmetric Cryptosystem

With the message and the encryption key as input, the encryption algorithm forms the ciphertext .We can write this as This notation indicates that is produced by using encryption algorithm E as a function of the plaintext , with the specific function determined by the value of the key .

The intended receiver, in possession of the key, is able to invert the transformation:

$$X=D(K,Y)$$

Cryptographic systems are characterized along three independent dimensions:

1. The type of operations used for transforming plaintext to ciphertext. All encryption algorithms are based on two general principles: substitution, in which each element in the plaintext (bit, letter, group of bits or letters) is mapped into another element, and transposition, in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost (that is, that all operations are reversible).

2. The number of keys used. If both sender and receiver use the same key, the system is referred to as symmetric, single-key, secret-key, or conventional encryption. If the sender and receiver use different keys, the system is referred to as asymmetric, two-key, or public-key encryption.

3. The way in which the plaintext is processed. A block cipher processes the input one block of elements at a time, producing an output block for each input block. A stream cipher processes the input elements continuously, producing output one element at a time, as it goes along.

UNIT-2

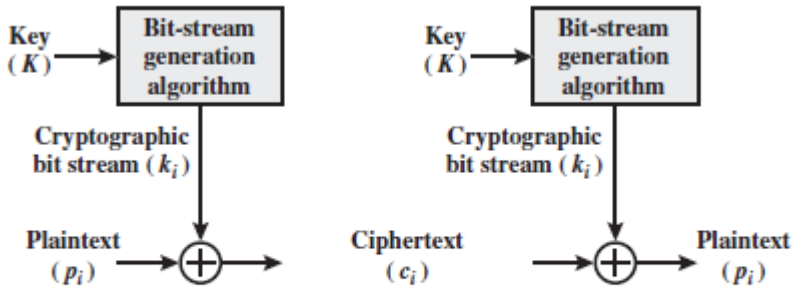
Symmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the same key. It is also known as conventional encryption, Symmetric encryption, secret key or single-key encryption.

TRADITIONAL BLOCK CIPHER STRUCTURE

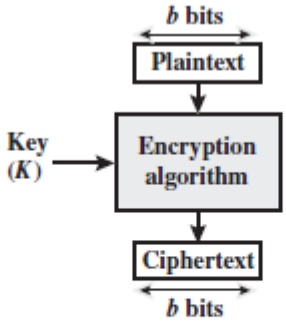
Stream Ciphers and Block Ciphers

A stream cipher is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the autokeyed Vigenère cipher and the Vernam cipher.

A block cipher is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. Typically, a block size of 64 or 128 bits is used.



(a) Stream cipher using algorithmic bit-stream generator



(b) Block cipher

Figure 3.1 Stream Cipher and Block Cipher

A block cipher operates on a plaintext block of n bits to produce a ciphertext block of n bits. There are 2^n possible different plaintext blocks and, for the encryption to be reversible (i.e., for decryption to be possible), each must

produce a unique ciphertext block. Such a transformation is called reversible, or nonsingular. The following examples illustrate nonsingular and singular transformations for $n = 2$.

Reversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	00
11	01

Irreversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	01
11	01

In the latter case, a ciphertext of 01 could have been produced by one of two plaintext blocks. So if we limit ourselves to reversible mappings, the number of different transformations is $2^n!$.[2]

THE FEISTEL CIPHER:

- Feistel cipher is the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers.
- In particular, Feistel proposed the use of a cipher that alternates substitutions and permutations, where these terms are defined as follows:
- Substitution: Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.
- Permutation: A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed.

FEISTEL CIPHER STRUCTURE:

The left-hand side of Figure 3.3 depicts the structure proposed by Feistel.

- The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key . The plaintext block is divided into two halves, L_0 and R_0.
- The two halves of the data pass through n rounds of processing and then combine to produce the ciphertext block.
- Each round i has as inputs *Li*-1 and *Ri*-1 derived from the previous round, as well as a subkey *Ki* derived from the overall K. In general, the subkeys Ki are different from K and from each other.

All rounds have the same structure. A substitution is performed on the left half of the data. This is done by applying a round function F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data.

Following this substitution a Permutation is performed that consists of the interchange of the two halves of the data.

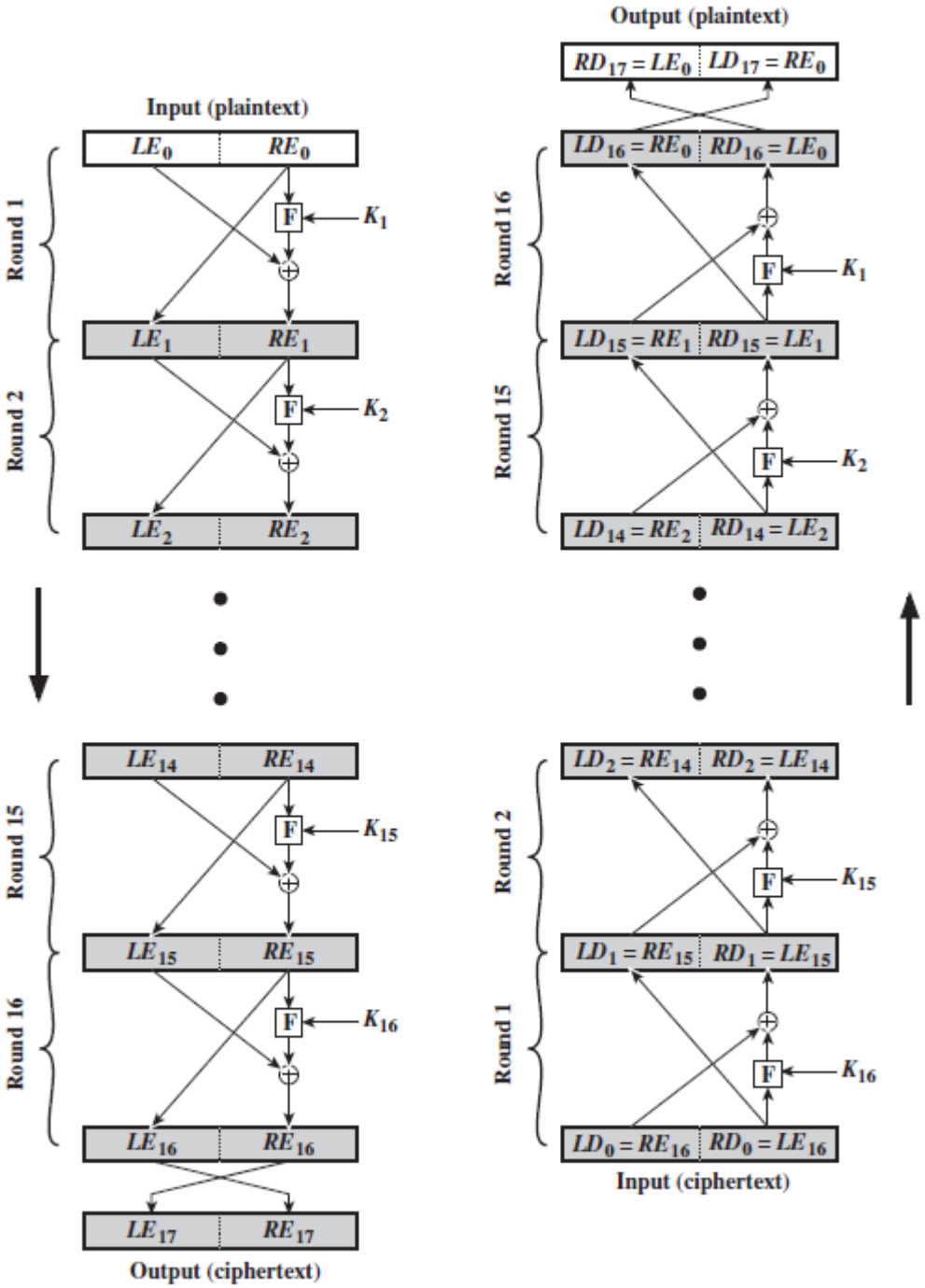


Figure 3.3 Feistel Encryption and Decryption (16 rounds)

encryption process	On the decryption side,
$LE_{16} = RE_{15}$ $RE_{16} = LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})$ **in general** $LE_i = RE_{i-1}$ $RE_i = LE_{i-1} \oplus \mathrm{F}(RE_{i-1}, K_i)$	$LD_1 = RD_0 = LE_{16} = RE_{15}$ $RD_1 = LD_0 \oplus \mathrm{F}(RD_0, K_{16})$ $= RE_{16} \oplus \mathrm{F}(RE_{15}, K_{16})$ $= [LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})] \oplus \mathrm{F}(RE_{15}, K_{16})$

The exact realization of a Feistel network depends on the choice of the following parameters and design features:

Block size: Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed for a given algorithm. Traditionally, a block size of 64 bits has been considered a reasonable tradeoff and was nearly universal in block cipher design. However, the new AES uses a 128-bit block size.

Key size: Larger key size means greater security but may decrease encryption/ decryption speed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion. Key sizes of 64 bits or less are now widely considered to be inadequate, and 128 bits has become a common size.

Number of rounds: The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.

Subkey generation algorithm: Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.

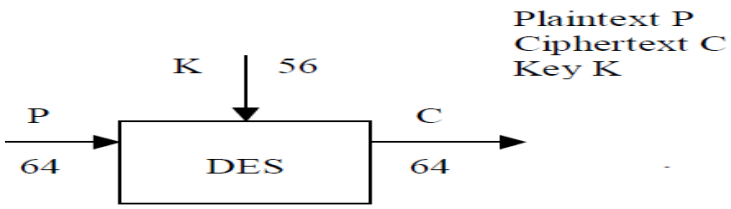
Round function F: Again, greater complexity generally means greater resistance to cryptanalysis.

DATA ENCRYPTION STANDARD (DES):

DES is a Symmetric-key algorithm for the encryption of electronic data.

Data Encryption Standard (DES) is a widely-used method of data encryption using a private (secret) key

DES applies a 56-bit key to each 64-bit block of data. The process can run in several modes and involves 16 rounds or operations.



Overall Structure

DES (and most of the other major symmetric ciphers) is based on a cipher known as the Feistel block cipher.

Looking at the left-hand side of the figure, we can see that the processing of the plaintext proceeds in three phases.

1. First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input.
2. This is followed by a phase consisting of sixteen rounds of the same function, which involves both permutation and substitution functions. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the preoutput.
3. Finally, the preoutput is passed through a permutation that is the inverse of the initial permutation function, to produce the 64-bit cipher text. With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher,

The right-hand portion of below figure shows the way in which the 56-bit key is used. Initially, the key is passed through a permutation function. Then, for each of the sixteen rounds, a *subkey* (*Ki*) is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.

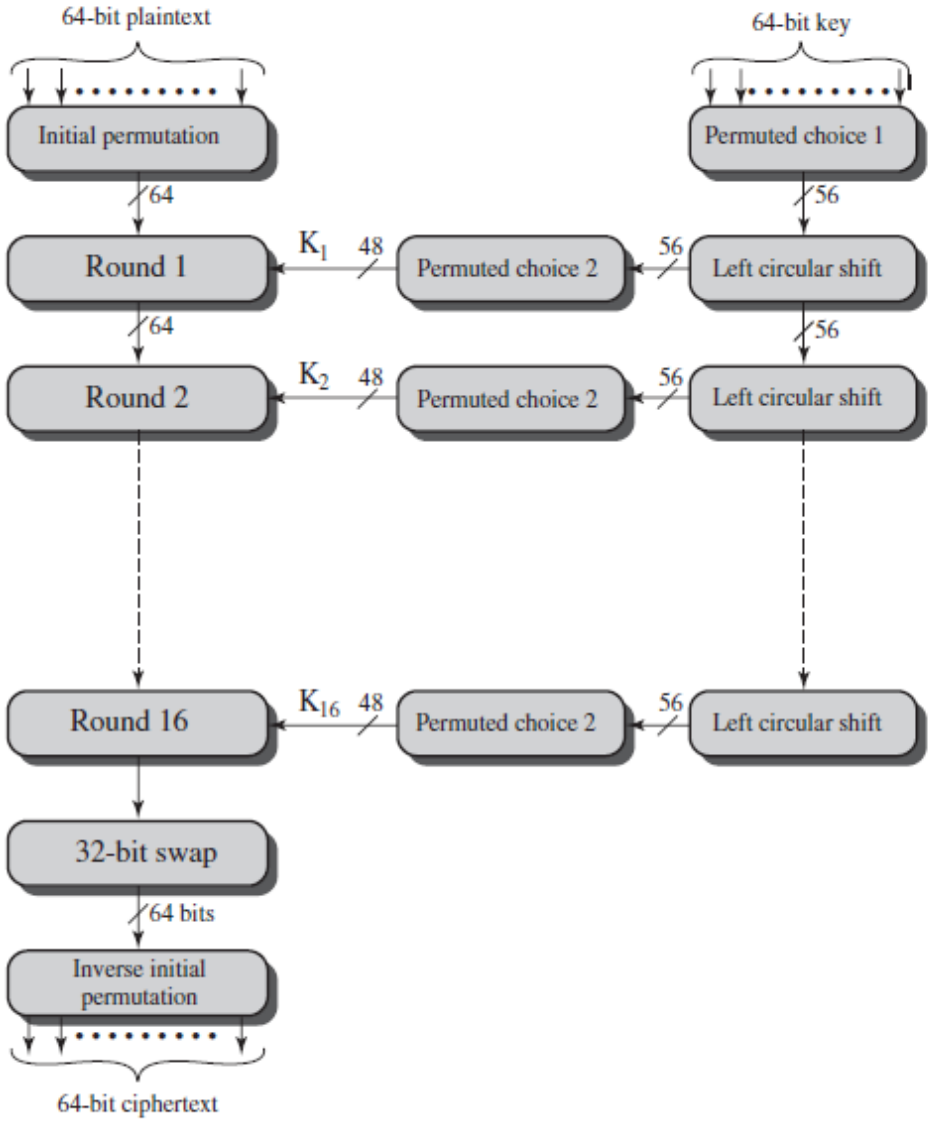


Figure 3.5 General Depiction of DES Encryption Algorithm

Details of Single Round

Below figure shows the internal structure of a single round. Again, begin by focusing on the left-hand side of the diagram.

- The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right).
- As in any classic Feistel cipher, the overall processing at each round can be summarized in the following formulas:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

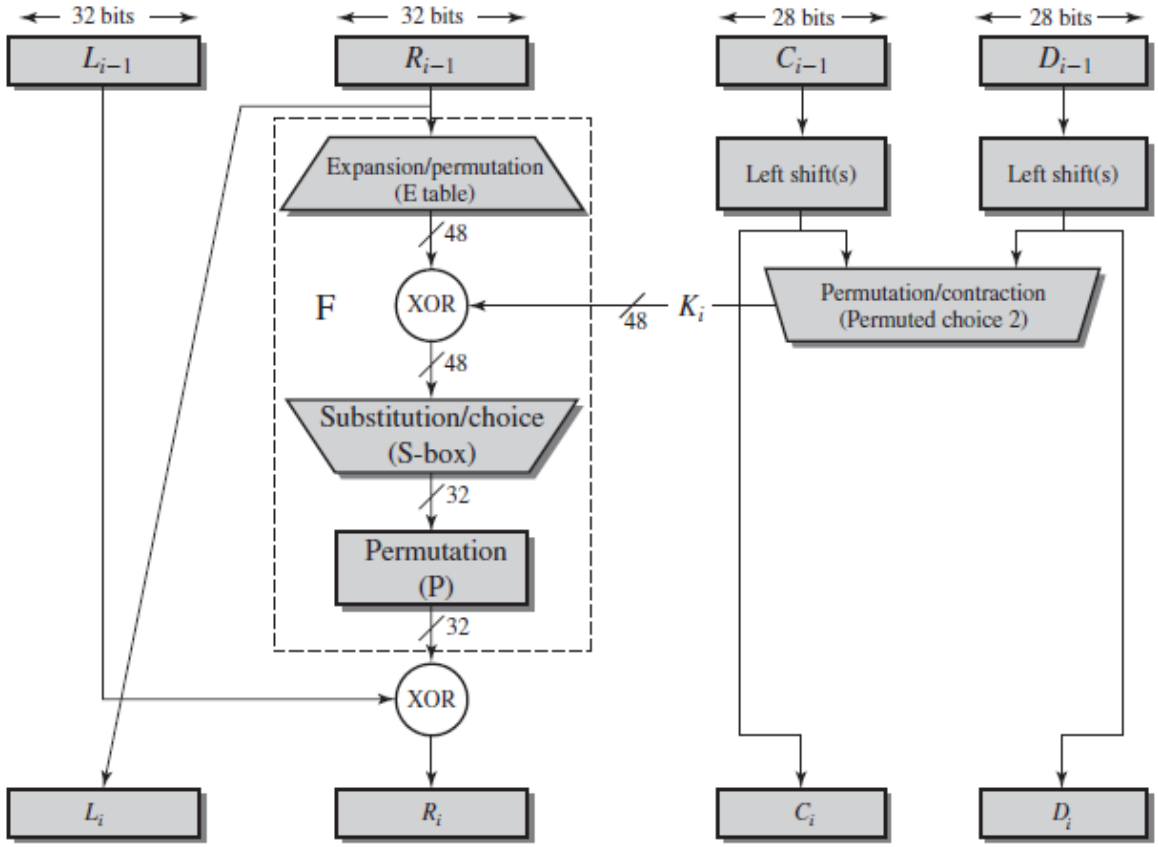


Figure :Single Round of DES Algorithm

- The round key K_i is 48 bits. The *R* input is 32 bits. This *R* input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the *R* bits .
- The resulting 48 bits are XORed with K_i . This 48-bit result passes through a substitution function that produces a 32-bit output, which is permuted .

.

S-Box Design in DES :

The role of the S-boxes in the function F is illustrated in Figure 3.7.The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output

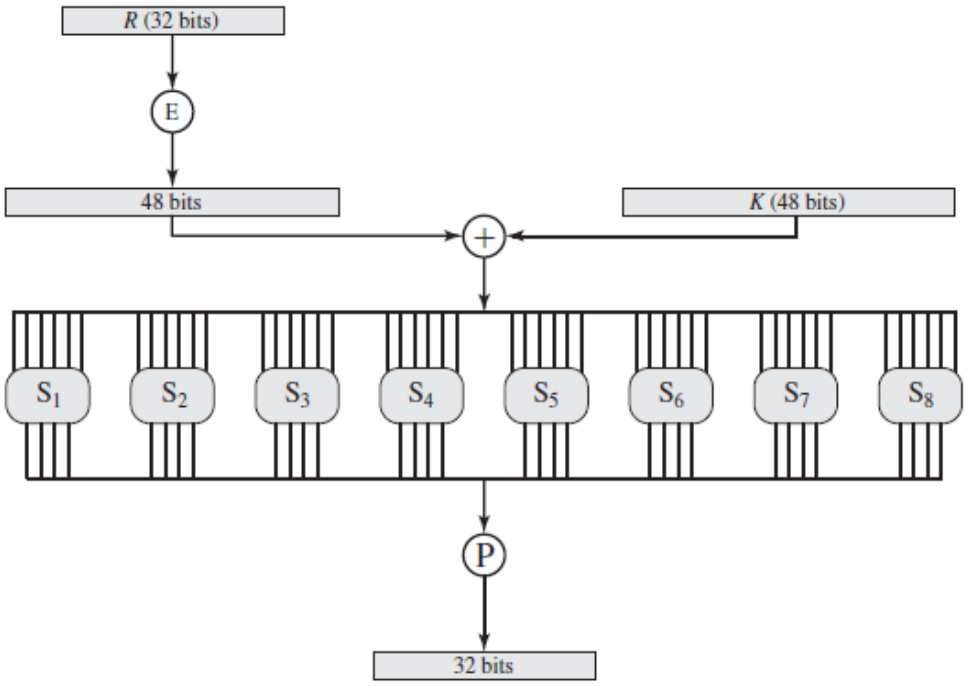


Figure 3.7 Calculation of F(R, K)

Key Generation

- Returning to above figure 3.4, we see that a 64-bit key is used as input to the algorithm.
- The bits of the key are numbered from 1 through 64; every eighth bit is ignored and The key is first subjected to a permutation .
- The resulting 56-bit key is then treated as two 28-bit quantities, labelled C0 and D_0. At each round, C_{i-1} and D_{i-1} are separately subjected to a circular left shift.
- These shifted values serve as input to the next round. They also serve as input to the part labeled Permuted Choice .which produces a 48-bit output that serves as input to the Function $F(R_{i-1}, K_i)$.

Des Decryption:

Whatever process we following in the encryption that process is used for decryption also but the order of key is changed on input message (cipher text).

Reverse order of keys are K_{16}, K_{15} ,……, K_1.

Strengths of DES:

- The DES is a symmetric key block cipher which takes 64bits cipher text and 56 bit key as an input and produce 64 bits cipher text as output.
- The DES function is made up of P & S boxes
- P-boxes transpose bits
- S-boxes Substitution bits to generating the cipher text.

The use of 56bits keys: 56 bit key is used in encryption, there are 2^{56} possible keys,which is approximately $2^{56}=7.2\times10^{16}$ keys, by this a brute force attack on such number of keys is impractical. A machine performing one DES encryption per microsecond would take more than a thousand years to break the cipher.

The nature of algorithm: Cryptanalyst can perform cryptanalysis by exploiting the characteristic of DES algorithm but no one has succeeded in finding out the weakness. This is possible because, in DES, they using 8-substitution tables or S-boxes in each iteration & one P-box transition for the every individual iteration.

Avalanche Effect:

key desirable property of an encryption algorithm

a small change in either the plain text or the key should produce a significant change in the cipher text(this property is called Avalanche Effect)

DES exhibits strong avalanche Effect

Timing Attacks:

Timing attack is one in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various ciphertexts.

The authors conclude that DES appears to be fairly resistant to a successful timing attack

BLOCK CIPHER DESIGN PRINCIPLES

Although much progress has been made in designing block ciphers that are cryptographically strong, the basic principles have not changed

There are three critical aspects of block cipher design:

- The number of rounds,
- Design of the function F,
- Key scheduling

Number of Rounds

The greater the number of rounds, the more difficult it is to perform cryptanalysis, even for a relatively weak F.

In general, the criterion should be that the number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple brute-force key search attack. This criterion was certainly used in the design of DES.

Design of Function F

The heart of a Feistel block cipher is the function F. in DES, this function relies on the use of S-boxes.

Design Criteria For F: *The* function F provides the element of confusion in a Feistel cipher. Thus, it must be difficult to "unscramble" the substitution performed by F. One obvious criterion is that F be nonlinear.

Several other criteria should be considered in designing F. We would like the algorithm to have good avalanche properties. Recall that, in general, this means that a change in one bit of the input should produce a change in many bits of the output.

Key Schedule Algorithm

With any Feistel block cipher, the key is used to generate one subkey for each round. In general, we would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key.

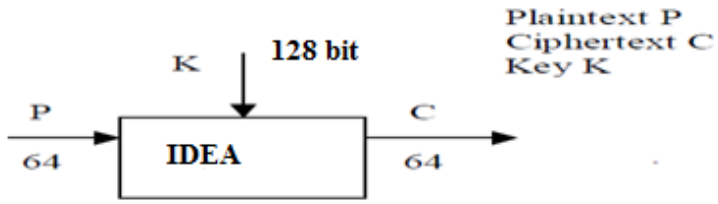
IDEA (International Data Encryption Algorithm):

IDEA originally called "IPES" (Improved proposed Encryption Standard).

IDEA is one of a number of conventional encryption algorithms that have been proposed in recent years to replace DES

IDEA is one of the most successful of these proposals. For example, IDEA is included in PGP.

Details of IDEA algorithm:



IDEA operates with 64 bit plain text and cipher text blocks and is controlled b a 128 bit key.

It avoids substitution boxes & lookup tables used in the block cipher.

The algorithm structure has been chosen such that different key sub-blocks are used; the encryption process is identical to the decryption process.

Encryption process in IDEA:

- The design principle behind IDEA is mixing of arithmetical operations form different algebraic groups.
- The underling operations are
    1. Exclusive-OR.
    2. Addition of integers modulo 2^{16}
    3. Multiplication modulo $2^{10}+1$
- The algorithm structure has been chosen such that when different key sub-blocks are used, the encryption process is identical to the decryption process
- The IDEA algorithm consists of eight rounds followed by a final transformation function. The algorithm divides the input into four 16-bit subblocks. Each of the rounds takes four 16-bit subblocks as input and produces four 16-bit output blocks. The final transformation also produces four %-bit blocks, which are concatenated to form the 64-bit ciphertext.

- Each of the rounds also makes use of six 16-bit subkeys, whereas the final transformation uses four subkeys, for a total of 52 subkeys

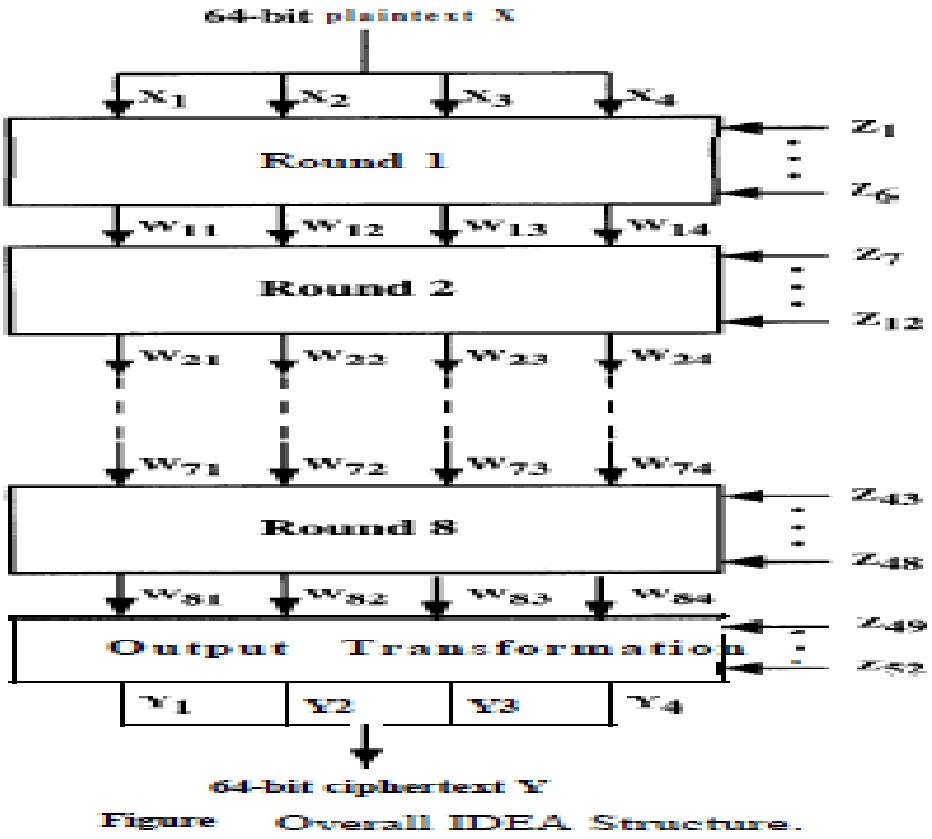


Figure Overall IDEA Structure.

Key Expansion (Encryption):

- The 128-bit key is expanded into 52 16-bit keys: K1, K2, ...K52. (in diagram we represented these keys with Z_1 to z_{52})
  Step 1: Keys K1….K8 are generated by taking 8 chunks of 16-bits each

- Step 2: Keys K9…K16 are generated by starting from the 25th bit, wrapping around the first 25 bits at the end, and taking 16-bit chunks.

- Step 3: Wrap around 25 more bits to the end, and generate keys K17…K24. This process is repeated until all keys K1…K52 are generated

<u>Details of a Single Round:</u>

64 bit data is divided into 4 16bit data blocks. These 4 blocks are processed through 8 rounds and transformed by the above arithmetical operations among each other and with 6 16 bit subkeys.

1. Multiply X_1 and the first sub key Z_1.
2. Add X_2 and the second sub key Z_2.
3. Add X_3 and the third sub key Z_3.
4. Multiply X_4 and the fourth sub key Z_4.
5. Bitwise XOR the results of steps 1 and 3.
6. Bitwise XOR the results of steps 2 and 4.
7. Multiply the result of step 5 and the fifth sub key Z_5.
8. Add the results of steps 6 and 7.
9. Multiply the result of step 8 and the sixth sub key Z_6.
10. Add the results of steps 7 and 9.
11. Bitwise XOR the results of steps 1 and 9.
12. Bitwise XOR the results of steps 3 and 9.
13. Bitwise XOR the results of steps 2 and 10.
14. Bitwise XOR the results of steps 4 and 10.

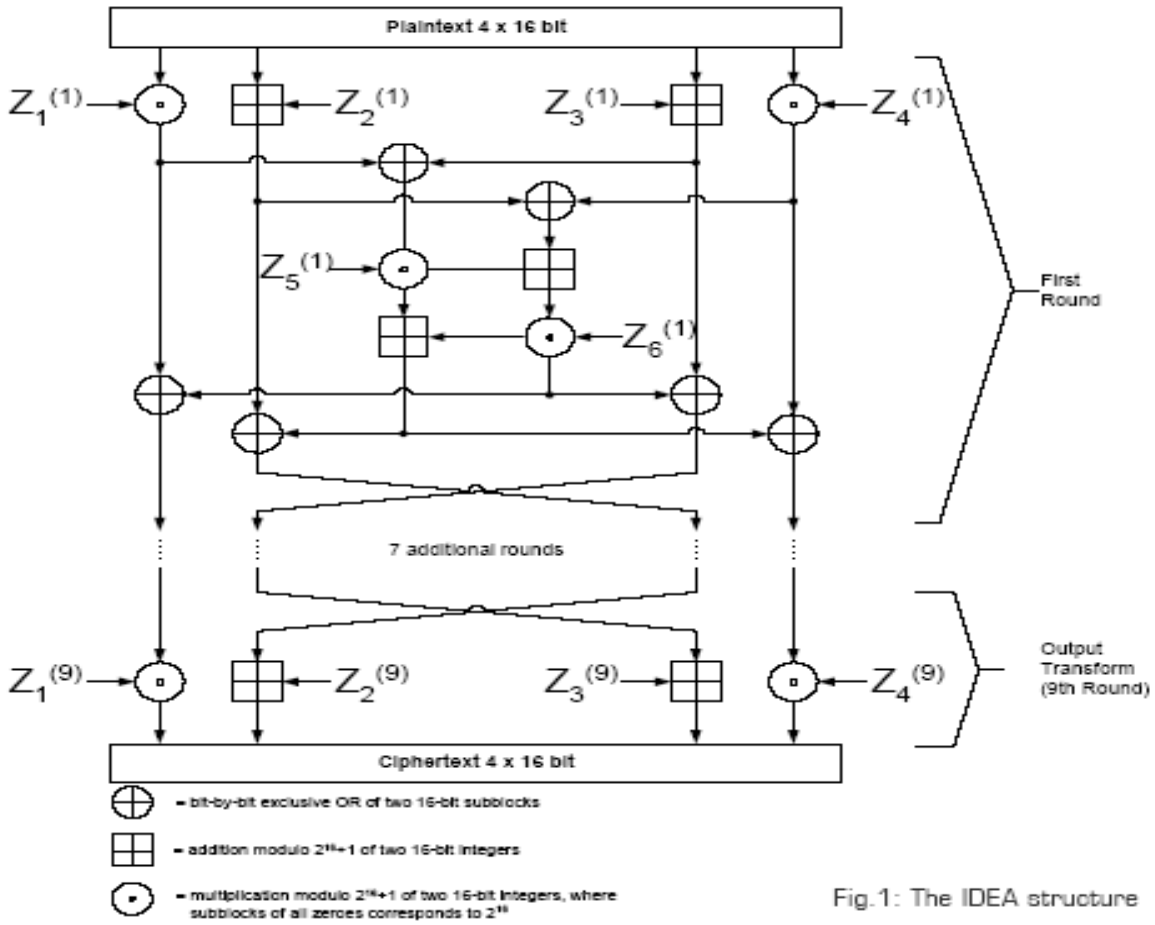


Fig.1: The IDEA structure

BLOW FISH ALGORITHM:

Blow fish is a symmetric block cipher developed by bruce schner in year 1993.

Blow fish is designed to have following characteristics

Speed: Blowfish encrypts data on 32 bit microprocessor at a rate of 18 clock cycles per byte.

Compact: it can run in less than 5k memory.

Simple: very easy to implements.

Variably secure: the key length is variable and can be as long as 448 bits. This allows a trade off between higher speed and higher security.

Blowfish is a feistal type model.

ALGORITHM:

- Blowfish is feistel type model, iterating a simple encryption function 16 times.
- Blowfish block size is 64 & key can be upto 448 bits.
- Blow fish encryption 64bits blocks of plaintext into 64 bit block of cipher.
- Blow fish make use of a key that ranges from 32bits to 448 bits (one to fourteen 32 bit keys).
- The keys are stored in a k-array (one to 14 32 bits)

  K_1, K_2----K_j where $1 \leq j \leq 14$.

- That key is used to generate 18 “32 bit” subkeys & four “8*32”bits S-boxes.
- The subkeys are stored in the p-array

  P_1, P_2,-------P_{18}

- There are four s-boxes(each s-box size is 8*32 bits) each with 256 32bit entries.

$S_{1,0}$, $S_{1,1}$,----------$S_{1,255}$

$S_{2,0}$, $S_{2,1}$,----------$S_{2,255}$

$S_{3,0}$, $S_{3,1}$,----------$S_{3,255}$

$S_{4,0}$, $S_{4,1}$,----------$S_{4,255}$

Encryption and Decryption

Blowfish uses two primitive operations:

Addition: Addition of words, denoted by +, is performed modulo 2^{32}.

Bitwise exclusive-OR: This operation is denoted by $\oplus$

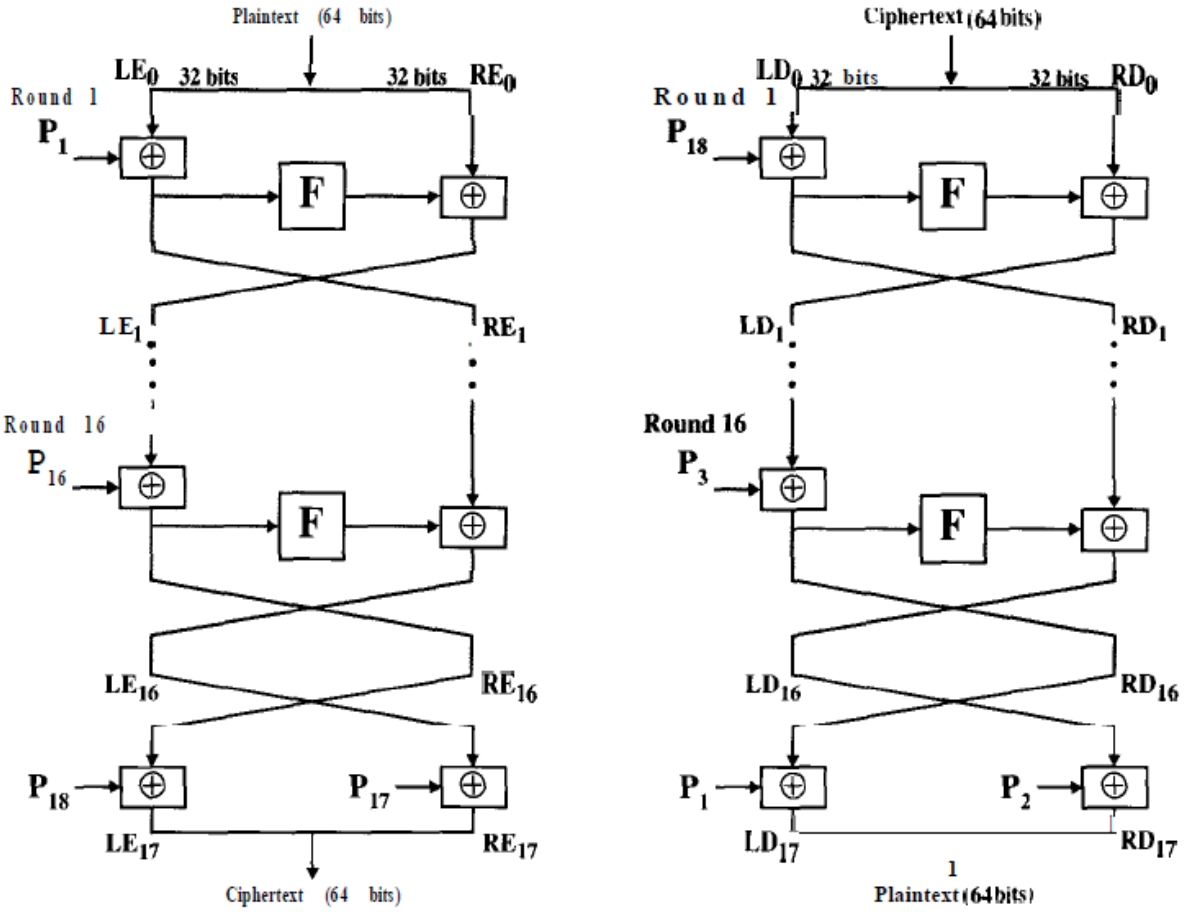


Figure Blowfish Encryption and Decryption.

In the above figure

- The plaintext is divided into two 32-bit halves LE, and RE,. We use the variables LE, and RE, to refer to the left and right half of the data after round i has completed. The algorithm can be defined by the following pseudocode:

for $i = 1$ **to 16 do**
$RE_i = LE_{i-1} \oplus P_i$;
$LE_i = F[RE_i] \oplus RE_{i-1}$;
$LE_{17} = RE_{16} \oplus P_{18}$;
$RE_{17} = LE_{16} \oplus P_{17}$;

Single round of Blowfish

The function F is shown in below Figure. The 32-bit input to F is divided into 4 bytes. If we label those bytes a, b, c, and d, then the function can be defined as follows:

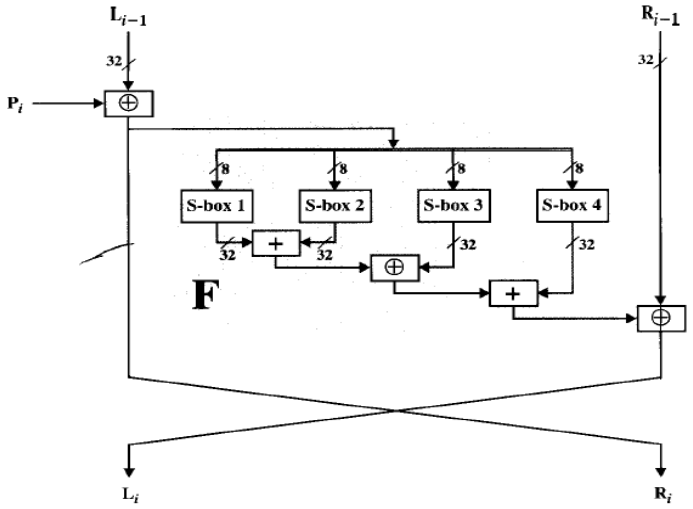


Figure Detail of Single Blowfish Round.

$$F[a, b, c, d,] = ((S_{1,a} + S_{2,b}) \oplus S_{3,c}) + S_{4,d}$$

CAST-64

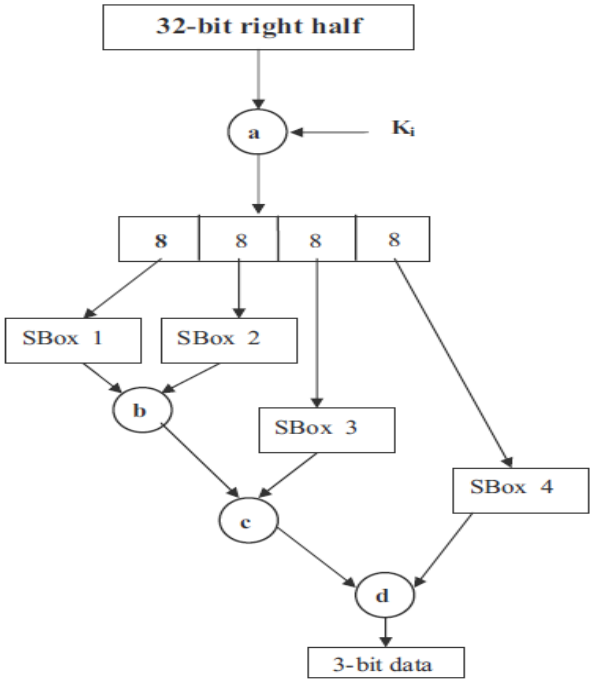
- The CAST Block Cipher is an improvement of the DES block cipher, invented in Canada by Carlisle Adams and Stafford Tavares. The name of the cipher seems to be after the initials of the inventors.
- The CAST algorithm has **64 bit block size** and has **a key of size 64 bits**.
- CAST is based on the Feistel structure to implement the substitution permutation network.

S-Boxes of CAST CAST uses S-Boxes of dimension $m \setminus n$ ($m < n$). The typical dimension of the S-Boxes of CAST is 8 \ 32. The principle behind the construction is as follows: choose n distinct binary bent functions of length $2m$, such that the linear combinations of these functions sum to highly non-linear, Boolen functions. Bent function are Boolean functions with even input variables having the highest possible non-linearity. The resultant functions also satisfy Strict Avalanche Criteria (SAC). SAC states that S-Box output bit j should change with probability . when any single input bit is changed, for all i, j. Note that the probability is computed over the set of all pairs of input vectors which differ only in bit i. Half of the bent functions have a weight of $(2^{m-1}+2^{(m/2)-1})$ and the other have a weight of $(2^{m-1} - 2^{(m/2)-1})$.

Encryption Function The plaintext block is divided into a left half and a right half. The algorithm has 8 rounds. Each round is essentially a Feistel structure. In each round the right half is combined with the round
key using a function f and then XOR-ed with the left half. The new left half after the round is the same as the right half before the round. After 8 iterations of the rounds, the left and the right half are concatenated to form the cipher text.

The Round Function f The round function in CAST can be realized as follows. The 32 bit input can be combined with 32 bits of the round key through a function, denoted by "a". The 32-bit data half is combined using operation "a" and the 32-bit result is split into 8 bit pieces. Each piece is input into a 8 x 32 S-Box. The output of S-Box 1 and 2 are combined using the operation "b"; the 32 bit output is combined with the output of S-Box 3, the output is

combined in turn with the output of S-Box 4. The combining functions are denoted in the fig by "c" and "d". A simple way would be where all the combining functions are XOR functions, however more complex operations may also be used.



Key Scheduling of CAST The key scheduling in CAST has three main components:

1. A key transformation step which converts the primary key (input key) to an intermediate key.
2. A relatively simple bit-selection algorithm mapping the primary key and the intermediate key to a form, referred as partial key bits.
3. A set of key-schedule S-Boxes which are used to create subkeys from the partial key bits.

Let, the input key be denoted by KEY = $k_1k_2k_3k_4k_5k_6k_7k_8$, where k_i is the i^{th} byte of the primary key. The key transformation step generates the intermediate key, KEY' = $k'_1k'_2k'_3k'_4k'_5k'_6k'_7k'_8$ as follows:

$$k_1'k_2'k_3'k_4' = k_1k_2k_3k_4 \oplus S_1[k_5] \oplus S_2[k_7]$$
$$k_5'k_6'k_7'k_8' = k_5k_6k_7k_8 \oplus S_1[k_2'] \oplus S_2[k_4']$$

Here, S_1 and S_2 are key-schedule S-Boxes of dimension 8×32.

Subsequently, there is a bit-selection step which operates as shown below:

$$K_1' = k_1k_2$$
$$K_2' = k_3k_4$$
$$K_3' = k_5k_6$$
$$K_4' = k_7k_8$$
$$K_5' = k_4'k_3'$$
$$K_6' = k_2'k_1'$$
$$K_7' = k_8'k_7'$$
$$K_8' = k_6'k_5'$$

The partial key bits are used to obtain the subkeys, K_i. The subkeys are 32 bits, and are obtained as follows:

$$K_i = S_1(K_{i,1}') \oplus S_2(K_{i,2}')$$

Here, $K'_{i,j}$ is the j^{th} byte of K'_i. Thus the 8 round subkeys are obtained.

The CAST block cipher can also be implemented with 128 bits, and is referred to as CAST-128. The essential structure of the cipher is still the same as discussed above.

ADVANCED ENCRYPTION STANDARD

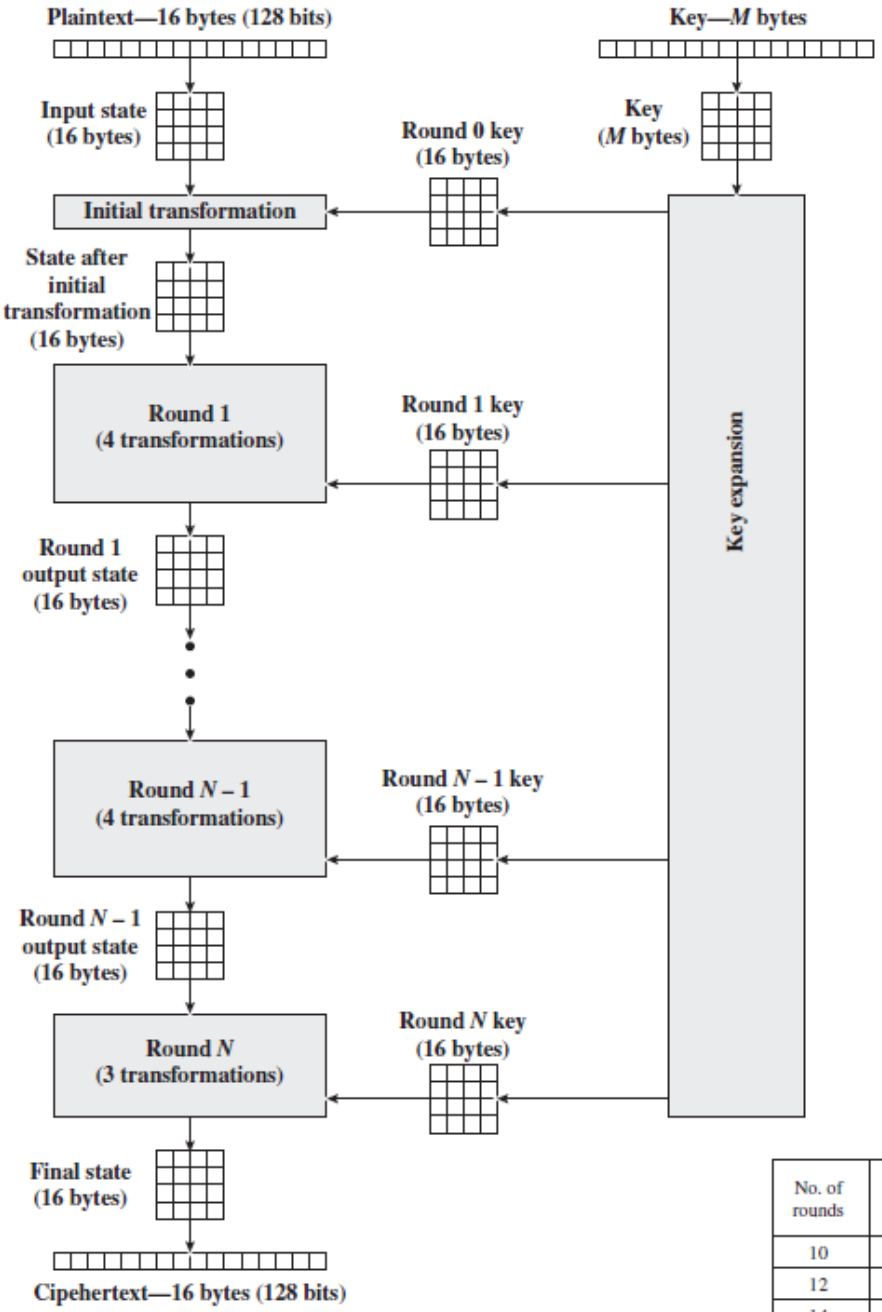
- The Advanced Encryption Standard (AES) was published by the National Institute of Standards and Technology (NIST) in 2001.
- AES is a block cipher intended to replace DES for commercial applications.
- It uses a 128-bit block size and a key size of 128, 192, or 256 bits.the algorithm is referred as AES-128,AES-192 OR AES-256
- AES does not use a Feistel structure. Instead, each full round consists of four separate functions: byte substitution, permutation, arithmetic operations over a finite field, and XOR with a key.
- AES parameters:

Key size(words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext block Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of rounds	10	12	14
Round Key size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded key size (words/bytes)	44/176	52/208	60/240

AES STRUCTURE

General structure

- The input to the encryption and decryption algorithms is a single 128-bit block. , this block is depicted as a 4 * 4 square matrix of bytes.
- This block is copied into the State array, which is modified at each stage of encryption or decryption.
- After the final stage, State is copied to an output matrix. These operations are depicted in Figure 5.2a. Similarly, the key is depicted as a square matrix of bytes.
- This key is then expanded into an array of key schedule words. Figure 5.2b shows the expansion for the 128-bit key. Each word is four bytes, and the total key schedule is 44 words for the 128-bit key



No. of rounds	Key Length (bytes)
10	16
12	24
14	32

Figure 5.1 AES Encryption Process

The cipher consists of N rounds, where the number of rounds depends on the key length: 10 rounds for a 16-byte key, 12 rounds for a 24-byte key, and 14 rounds for a 32-byte key (Table 5.1).

The first N-1 rounds consist of four distinct transformation functions: SubBytes, ShiftRows, MixColumns, and AddRoundKey, which are described subsequently. The final round contains only Three Transformations, and there is a initial single transformation (AddRoundKey) before

the first round,

which can be considered Round 0. Each transformation takes one or more 4×4 matrices as input and produces a 4×4 matrix as output. Figure 5.1 shows that the output of each round is a 4×4 matrix, with the output of the final round being the ciphertext. Also, the key expansion function generates $N + 1$ round keys, each of which is a distinct 4×4 matrix. Each round key serve as one of the inputs to the AddRoundKey transformation in each round.

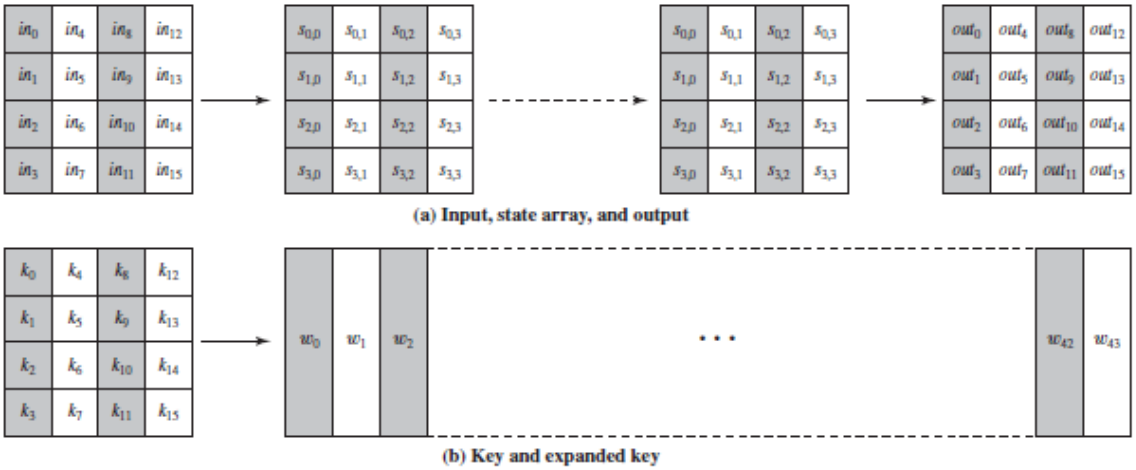


Figure 5.2 AES Data Structures

Detailed Structure

- Figure 5.3 shows the AES cipher in more detail, indicating the sequence of transformations in each round and showing the corresponding decryption function

Four different stages are used, one of permutation and three of substitution:

• Substitute bytes: Uses an S-box to perform a byte-by-byte substitution of the block

• ShiftRows: A simple permutation

• MixColumns: A substitution that makes use of arithmetic

• AddRoundKey: A simple bitwise XOR of the current block with a portion of the expanded key

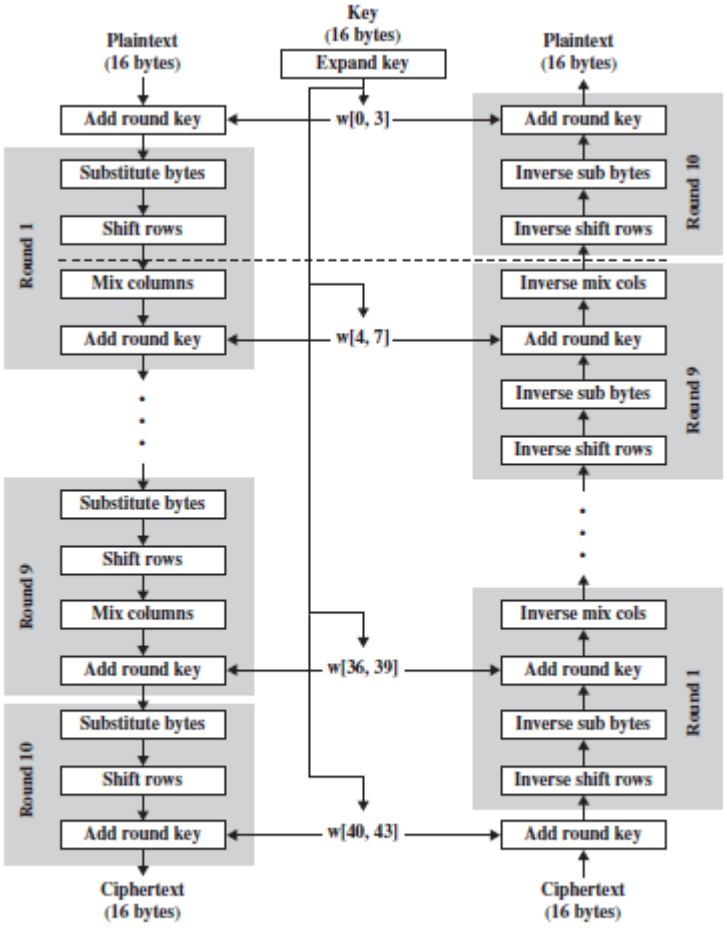


Figure 5.3 AES Encryption and Decryption

AES TRANSORMATION FUNCTIONS

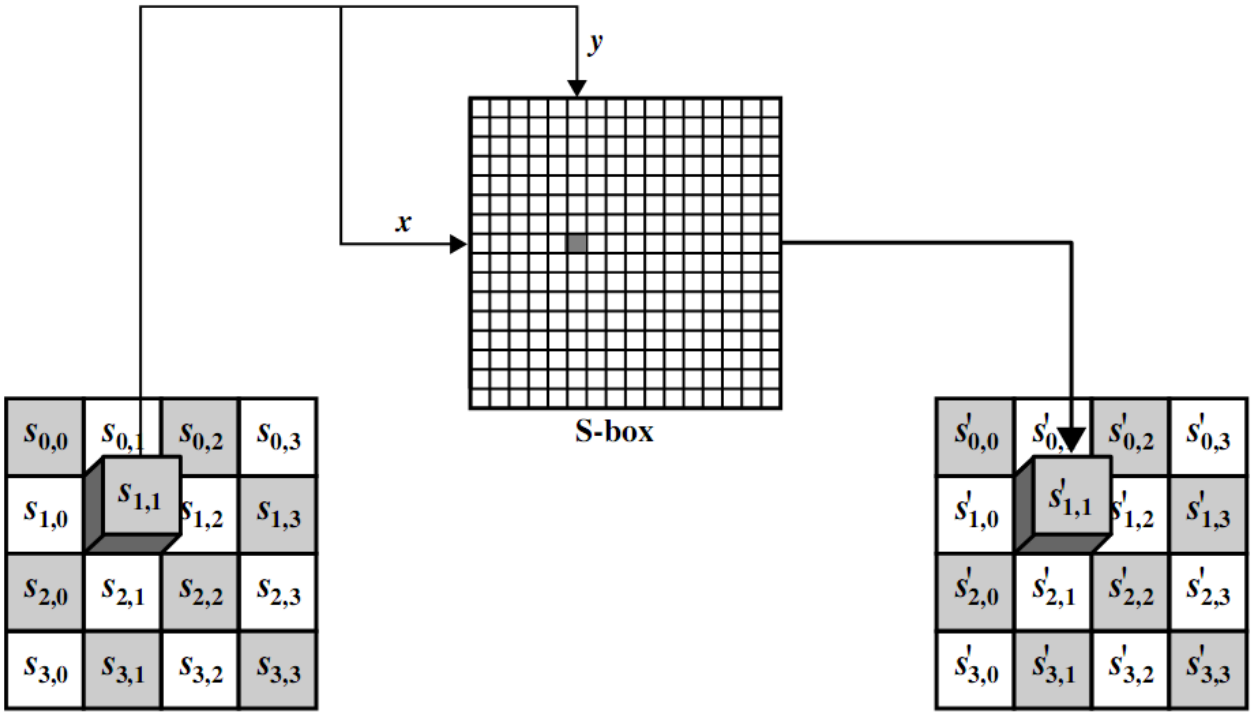
The four transformation functions are

- Substitute bytes
- ShiftRows
- MixColumns
- AddRoundKey

Substitute Bytes Transformation

- The forward substitute byte transformation, called SubBytes, is a simple table lookup (Figure 5.5a).
- AES defines a16 *16 matrix of byte values, called an S-box (Table 5.2a), that contains a permutation of all possible 256 8-bit values.
- Each individual byte of State is mapped into a new byte in the following way: The leftmost 4 bits of the byte are used as a row value and the rightmost 4 bits are used as a

column value. These row and column values serve as indexes into the S-box to select a unique 8-bit output value. For example, the hexadecimal value3 {95} references row 9, column 5 of the S-box,which contain the value {2A}



(a) Substitute bvte transformation

Table 5.2 AES S-Boxes

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

(a) S-box

Here is an example of the SubBytes transformation:

EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

Shift Rows Transformation

- The first row of State is not altered.
- For the second row, a 1-byte circular left shift is performed.
- For the third row, a 2-byte circular left shift is performed.
- For the fourth row, a 3-byte circular left shift is performed

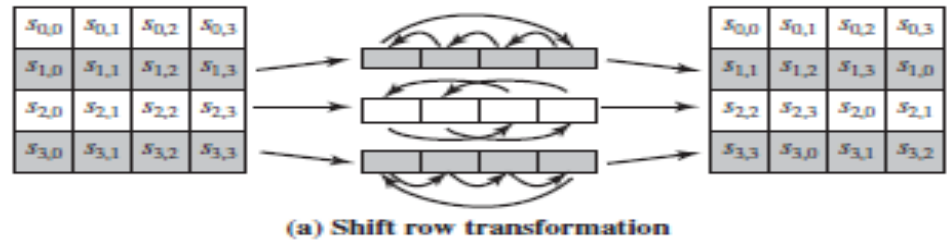
The following is an example of ShiftRows:

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

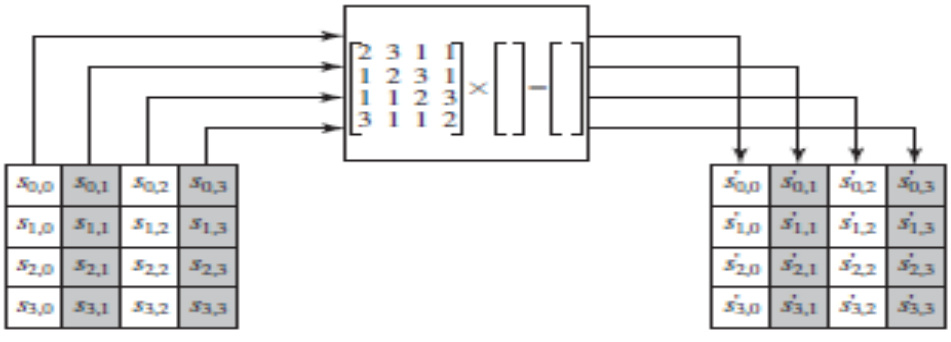
→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

The **inverse shift row transformation**, called InvShiftRows, performs the circular shifts in the opposite direction for each of the last three rows, with a 1-byte circular right shift for the second row, and so on.



(a) Shift row transformation



(b) Mix column transformation

Figure 5.7 AES Row and Column Operations

Mix columns Transformation

- The forward mix column transformation, called MixColumns, operates on each column individually.
- Each byte of a column is mapped into a new value that is a function of all four bytes in that column.

transformation can be defined by the following matrix multiplication on **State** (Figure 5.7b):

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} \quad (5.3)$$

The following is an example of MixColumns:

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$$\begin{aligned}
(\{02\} \cdot \{87\}) \oplus (\{03\} \cdot \{6E\}) \oplus \{46\} \oplus \{A6\} &= \{47\} \\
\{87\} \oplus (\{02\} \cdot \{6E\}) \oplus (\{03\} \cdot \{46\}) \oplus \{A6\} &= \{37\} \\
\{87\} \oplus \{6E\} \oplus (\{02\} \cdot \{46\}) \oplus (\{03\} \cdot \{A6\}) &= \{94\} \\
(\{03\} \cdot \{87\}) \oplus \{6E\} \oplus \{46\} \oplus (\{02\} \cdot \{A6\}) &= \{ED\}
\end{aligned}$$

For the first equation, we have $\{02\} \cdot \{87\} = (0000\ 1110) \oplus (0001\ 1011) = (0001\ 0101)$ and $\{03\} \cdot \{6E\} = \{6E\} \oplus (\{02\} \cdot \{6E\}) = (0110\ 1110) \oplus (1101\ 1100) = (1011\ 0010)$. Then,

$$\begin{aligned}
\{02\} \cdot \{87\} &= 0001\ 0101 \\
\{03\} \cdot \{6E\} &= 1011\ 0010 \\
\{46\} &= 0100\ 0110 \\
\{A6\} &= \underline{1010\ 0110} \\
&\quad 0100\ 0111 = \{47\}
\end{aligned}$$

The other equations can be similarly verified.

AddRoundKey Transformation

- In the AddRoundKey transformation, the 128 bits of State are bitwise XORed with the 128 bits of the round key.
- The operation is viewed as a columnwise operation between the 4 bytes of a State column and one word of the round key; it can also be viewed as a byte-level operation.

The following is an example of AddRoundKey:

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$\oplus$

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

$=$

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D6

The first matrix is **State**, and the second matrix is the round key.
The **inverse add round key transformation** is identical to the forward add round key transformation, because the XOR operation is its own inverse.

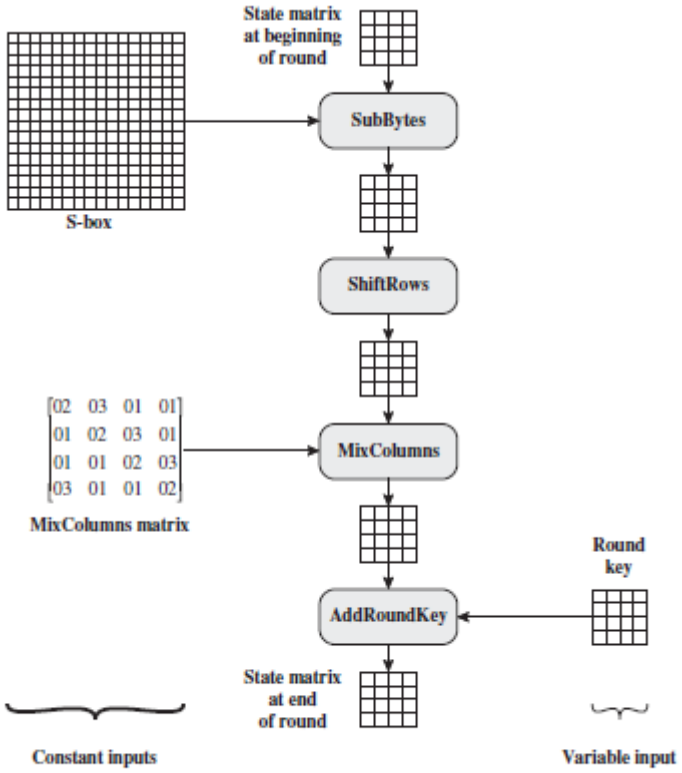
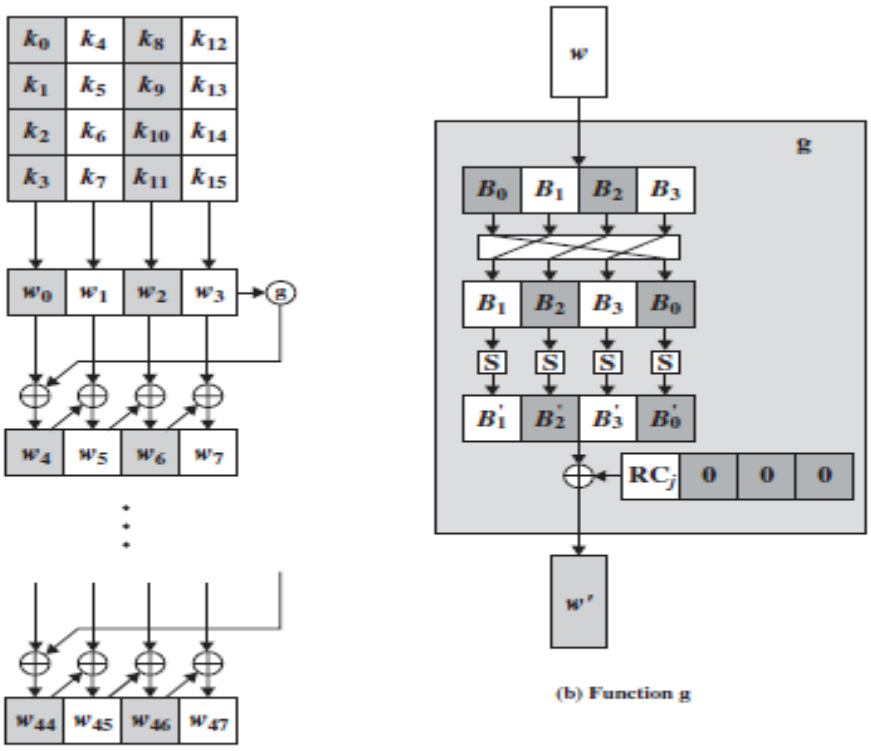


Figure 5.8 Inputs for Single AES Round

AES KEY EXPANSION

- The AES key expansion algorithm takes as input a 4-word (16-byte) key and produces a linear array of 44 words (176 bytes).
- This is sufficient to provide a 4-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher.
- The key is copied into the first four words of the expanded key. The remainder of the expanded key is filled in four words at a time.
- Each added word w[i] depends on the immediately preceding word, w[i 1], and the word four positions back,w[i 4].



(a) Overall algorithm

(b) Function g

Figure 5.9 AES Key Expansion

In three out of four cases, a simple XOR is used. For a word whose position in the w array is a multiple of 4, a more complex function is used. The function 'g' consists of the following subfunctions:

1. RotWord performs a one-byte circular left shift on a word. This means that an input word [b0, b1, b2, b3] is transformed into [b1, b2, b3, b0].
2. SubWord performs a byte substitution on each byte of its input word, using the S-box.
3. The result of steps 1 and 2 is XORed with a round constant, Rcon[j].

UNIT-3(PART-I)

(NUMBER THEORY)

Prime Numbers

An integer $p > 1$ is a prime number if and only if its only divisors[2] are ± 1 and $\pm p$. **Prime numbers** play a critical role in number theory

Any integer $a > 1$ can be factored in a unique way as

$$a = p_1^{a_1} \times p_2^{a_2} \times \cdots \times p_t^{a_t} \quad \textbf{(8.1)}$$

where $p_1 < p_2 < \ldots < p_t$ are prime numbers and where each a_i is a positive integer. This is known as the fundamental theorem of arithmetic; a proof can be found in any text on number theory.

$$91 = 7 \times 13$$
$$3600 = 2^4 \times 3^2 \times 5^2$$
$$11011 = 7 \times 11^2 \times 13$$

It is useful for what follows to express this another way. If P is the set of all prime numbers, then any positive integer a can be written uniquely in the following form:

$$a = \prod_{p \in P} p^{a_p} \qquad \text{where each } a_p \geq 0$$

Relative Prime Numbers

Two numbers are said to be relative prime numbers when they share no factors in common other than one

If two integers a, b are relatively prime if gcd(a,b)=1.

Examples:

1) 15 and 28 are relatively prime numbers
   15=the factors are (1,3,5)
   28=the factors are(1,2,4,7)
2) 7 and 20 are relatively prime numbers
3) 12 and 13 are relatively prime numbers

MODULAR ARITHMETIC

The Modulus

If ‘a’ is an integer and’n’ is a positive integer, we define **amodn** to be the remainder when is ‘a’ is divided by‘n’ . The integer is called the **modulus**. Thus, for any integer , we can write the Equation as follows:

$$\boldsymbol{a = qn + r}$$

example:11 mod 7 = 4; - 11 mod 7 = 3

Congruence

Two integers and are said to be **congruent modulo *n***, if (a mod n)=(*b* mod *n*).This is written as

$$a \equiv b \pmod{n}.$$

We say that a is congruent to b modulo m if m divides b-a or a-b.

Examples:

$73 \equiv 4 \pmod{23}; \qquad 21 \equiv -9 \pmod{10}$

$20 \equiv 0 \pmod{10}$

Properties of Congruences

Congruences have the following properties:

1. $a \equiv b \pmod{n}$ if $n|(a - b)$.
2. $a \equiv b \pmod{n}$ implies $b \equiv a \pmod{n}$.
3. $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$ imply $a \equiv c \pmod{n}$.

Modular Arithmetic Operations

Modular arithmetic exhibits the following properties:

1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
2. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
3. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

Examples:

$11 \bmod 8 = 3; 15 \bmod 8 = 7$
$[(11 \bmod 8) + (15 \bmod 8)] \bmod 8 = 10 \bmod 8 = 2$
$(11 + 15) \bmod 8 = 26 \bmod 8 = 2$

$[(11 \bmod 8) - (15 \bmod 8)] \bmod 8 = -4 \bmod 8 = 4$
$(11 - 15) \bmod 8 = -4 \bmod 8 = 4$

$[(11 \bmod 8) \times (15 \bmod 8)] \bmod 8 = 21 \bmod 8 = 5$
$(11 \times 15) \bmod 8 = 165 \bmod 8 = 5$

Modular Addition And Multiplication

Table 4.2 provides an illustration of modular addition and multiplication modulo 8

Table 4.2 Arithmetic Modulo 8

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Addition modulo 8

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

(b) Multiplication modulo 8

Properties of Modular Arithmetic

Define the set Z_n as the set of nonnegative integers less than n:

$$Z_n = \{0, 1, \ldots, (n-1)\}$$

This is referred to as the **set of residues**, or **residue classes** (mod n). To be more precise, each integer in Z_n represents a residue class. We can label the residue classes (mod n) as $[0], [1], [2], \ldots, [n-1]$, where

$$[r] = \{a: a \text{ is an integer}, a \equiv r \pmod{n}\}$$

The residue classes (mod 4) are

$[0] = \{\ldots, -16, -12, -8, -4, 0, 4, 8, 12, 16, \ldots\}$
$[1] = \{\ldots, -15, -11, -7, -3, 1, 5, 9, 13, 17, \ldots\}$
$[2] = \{\ldots, -14, -10, -6, -2, 2, 6, 10, 14, 18, \ldots\}$
$[3] = \{\ldots, -13, -9, -5, -1, 3, 7, 11, 15, 19, \ldots\}$

Fermat's Theorem:-

This theorem state the following. If 'p' is prime and 'a' is positive integer and not divisible by 'p' then $a^{p-1} \equiv 1 \pmod{p}$ and every integer 'a' $a^p \equiv a \pmod{p}$

This means that if you divide a^p by p then the remainder is 'a'.

It is useful in public key cryptography (RSA) and pramality testing.

Proof:- Consider a set of positive integers lessthan 'p'

$$A = \{1, 2 \ldots (p-1)\}$$

Multiply each element 'a' and "modulo p" then we can get

$$X = \{a \bmod p, 2a \bmod p, \ldots (p-1)a \bmod p\}$$

None of the element of 'x' is equal to 0 because 'p' doesnot divide 'a'

No two of the integers in 'x' are equal.

Assume that $ja \equiv ka \pmod{p}$

where $1 \leq j < k \leq p-1$

Eliminate 'a' from both sides because 'a' is relative prime to 'p'

$$j \equiv k \pmod{p}$$

This is impossible. J and k both are positive integers lessthan 'p'. we know that p-1 elements of x are all positive integers. we can conclude the 'x' consists of a set of integers $\{1, 2 \ldots p-1\}$

Multiply the numbers in both sets and taking the result modp produces

$$a * 2a \ldots (p-1)a = [(1 * 2 \ldots (p-1)] \pmod{p}$$

$$a^{p-1}(p-1)! = (p-1)! \pmod p$$

cancel "(p-1)!" because it is relative prime to p

$$\boxed{a^{p-1} \equiv 1 \pmod p} \rightarrow ①$$

An alternative form of formals theorem is given as

$$\boxed{a^p = a \pmod p} \rightarrow ②$$

Eg:- $(6^{17}) \bmod 13$

$(6^{12} \times 6^5) \bmod 13$

$(6^{12} \bmod 13) * (6^5 \bmod 13)$

$= 1 * 6^5 \bmod 13$

$= 2$

Eg:- $2^{48} \bmod 5$

$= 1$

b) Find the least residue of 9^{794} modulo 73

73 is prime

from fermats theorem $9^{72} \equiv 1 \bmod 73$

$9^{794} = (9^{72})^{11} \cdot 9^2 \equiv 9^2 \pmod{73}$

$\equiv 81 \pmod{73}$

$= 8$

b) Find the solution to the congruence $x^{86} \equiv 6 \pmod{29}$

29 is prime

$x^{28} \equiv 1 \pmod{29} \rightarrow ①$

$x^{86} = (x^{28})^3 \cdot x^2$

$x^2 \equiv 6 \pmod{29}$

x=8

(d) Use fermat theorem to find the number 'x' between 0 and to $28 \cdot x^{85}$ congruence to 6 (mod 35)

$x^{85} \equiv 6 \pmod{35}$

$x^{34} \equiv 1 \pmod{35}$

No solution

35 is not prime
So according to "fermat's" theorem this can ~~not~~ be not solved in

Euler's Theorem:

Euler's totient function:-

Before going to euler's theorem we must know about important quantity in number theorey ie Euler's totient function. It can be represented as ϕ

$\phi(n)$ is the number of positive integers lessthan 'n' and the relative prime to 'n'.

Means how many numbers that are between 1 and (n-1) that are relative prime to n.

$\phi(35)$: 1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 22, 23, 24, 26, 27, 29, 31, 32, 33, 34

$\phi(35) = 24$

$\phi(37) = 36$

$\phi(p) = (p-1)$

It should clear that for a prime number p. $\boxed{\phi(p) = (p-1)}$

By this $\phi(n)$ will be easy to calculate when 'n' has exactly two different prime factors.

That is we have two prime numbers p and q with $p \neq q$ the $n = pq$.

$$\boxed{\phi(n) = \phi(pq) = \phi(p) \times \phi(q) = (p-1) \times (q-1)}$$

Ex: $\phi(21) = \phi(3 \times 7) = \phi(3) \times \phi(7)$

$= 2 \times 6 = 12$

Euler's Theorem:

Euler's theorem states that for every 'a' and 'n' the are relatively prime

$$a^{\phi(n)} = 1 \pmod{n} \longrightarrow (1)$$

1. $a=3$, $n=10$, show that $a^{\phi(n)} \equiv 1 \pmod{n}$

$$3^{\phi(10)} = 1 \pmod{10} \quad [\phi(10) = \phi(5\times 2) = 4\times 1 = 4]$$

$$3^4 = 1 \pmod{10}$$

$$81 = 1 \pmod{10}$$

2. $a=2$, $n=11$ then show that $a^{\phi(n)} \equiv 1 \pmod{n}$

$$2^{\phi(11)} \equiv 1 \pmod{n}$$

$$2^{10} = 1 \pmod{11}$$

Proof:- Above equation (1) is true if 'n' is prime because in that case $\phi(n) = n-1$ and fermat's theorem holds.

From Euler's totient function $\phi(n)$ is the number of positive integers less than 'n' that are relative prime to 'n'

Consider a set of such integers labeled as $R = \{x_1, x_2, \ldots x_{\phi(n)}\}$

Now multiply each element by 'a', modulo 'n'

$$S = \{(ax_1 \bmod n), (ax_2 \bmod n) \cdots (ax_{\phi(n)} \bmod n)\}$$

The set 's' is a permutation of 'r' by the following reasons.

(1) 'a' is a relative prime to 'n' and x_i is a relative prime to 'n', ax_i must also be relative prime to 'n'. Thus all the numbers of s are integers less than 'n' that are relative prime to 'n'.

(2) There are no duplicates in 's'

if $a x_i \bmod n = a x_j \bmod n$

$$\therefore \frac{\phi(n)}{\pi} (a x_i \bmod n) = \prod_{i=1}^{\phi(n)} x_i$$

$$\prod_{i=1}^{\phi(n)} a x_i = \prod_{i=1}^{\phi(n)} x_i \pmod n$$

→ eliminate

$$a^{\phi(n)} \times \left[\prod_{i=1}^{\phi(n)} \right] x_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod n$$

$$a^{\phi(n)} = 1 \bmod n$$

CRT (chinese Remainder Theorem):-

→ One of the most useful elements of number theory is CRT.

→ CRT says it is possible to reconstructs integers in a certain range in their residues module, a set of pairwise relatively prime module

→ The CRT is a method for solving certain system of congruency

→ The CRT is a mechanism for manipulating very large numbers in terms of tuples of small integers.

Theorem:- Suppose that $m_1, m_2 \ldots m_r$ are relative prime numbers and let $a_1, a_2 \ldots a_r$ be integers then the system of congruences

$$X \equiv a_i \pmod{m_i} \text{ for } 1 \le i \le r$$ has a unique solution modulo $M = m_1 \times m_2 \times \ldots \times m_r$ which is given by

$$X \equiv a_1 m_1 y_1 + a_2 m_2 y_2 + \ldots + a_r m_r y_r \pmod m$$ where

Explanation $M_i = M/m_i$ and $Y_i \equiv (M_i)^{-1} \pmod{m_i}$ for $1 \le i \le n$

The CRT can be evaluated in five steps.

$$X \equiv a_1 \pmod{m_1}$$
$$X \equiv a_2 \pmod{m_2}$$
$$X \equiv a_3 \pmod{m_3}$$
$$\overline{X \equiv a_i \pmod{m_i}}$$

Step1:- calculate $M = m_1 \times m_2 \cdots \times m_i$ ie. $\prod_{K=1}^{i} m_k$

Step2:- calculate $M_1 = M/m_1$, $M_2 = M/m_2 \cdots M_i = M/m_i$

Step3:- calculate A's = $\{a_1, a_2, a_3 \cdots a_i\}$ if necessary

Step4:- Find multiplicative inverse of each $M_1, M_2, \ldots m_i$ mod $m_1, m_2 \cdots m_i$ respectively

Step5:- $\sum (a_i M_i M_i^{-1}) \bmod M$ is result

Examples: Refer the examples given in the class room.

DISCRETE LOGARITHMS

Multiplicative order:-

Given an integer 'a' and positive integer 'n' with gcd(a,n)=1 (a, n are relative prime)

The multiplicative order of a 'modulo n' is smallest positive integer 'k' with $a^k \equiv 1 \pmod n$

The order of modulo 'n' is written as $ord_n(a)$ or $O_n(a)$

Ex: Define multiplicative order of 4 mod 7.

$4^2 = 16 \equiv 2 \pmod 7$

$4^3 = 64 \equiv 1 \pmod 7$

$ord_7(4) = 3$

Find the multiplicative order of 2 mod 7

$2 \equiv 2 \pmod 7$

$2^2 \equiv 4 \pmod 7$

$2^3 \equiv 1 \pmod 7$

$ord_7(2) = 3$

Primitive root

If a is a primitive root of n then its powers

$$a, a^2, \ldots, a^{\phi(n)}$$

are distinct (mod n) and are all relatively prime to n. In particular, for a prime number p, if a is a primitive root of p, then

$$a, a^2, \ldots, a^{p-1}$$

are distinct (mod p). For the prime number 19, its primitive roots are 2, 3, 10, 13, 14, and 15.

Exaple::Refer the examples given in the class room

Logarithms in modular arithmetic:

by the definition of modular arithmetic. It follows that for any integer b and a primitive root a of prime number p, we can find a unique exponent i such that

$$b \equiv a^i \pmod{p} \quad \text{where } 0 \leq i \leq (p-1)$$

This exponent i is referred to as the **discrete logarithm** of the number b for the base a (mod p). We denote this value as $\text{dlog}_{a,p}(b)$.[10]

- QUADRATIC CONGRUENCE
- ***In cryptography, we also need to discuss quadratic congruence¾that is, equations of the form***
  $a_2x^2 + a_1x + a_0 \equiv 0 \pmod{n}$. We limit our discussion to quadratic equations in which $a_2 = 1$ and $a_1 = 0$, that is equations of the form
- ***$x^2 \equiv a \pmod{n}$.***

1 . Quadratic Congruence Modulo a Prime

We first consider the case in which the modulus is a prime.

Example: The equation x2 ≡ 3 (mod 11) has two solutions, x ≡ 5 (mod 11) and x ≡ −5 (mod 11). But note that −5 ≡ 6 (mod 11), so the solutions are actually 5 and 6. Also note that these two solutions are incongruent.

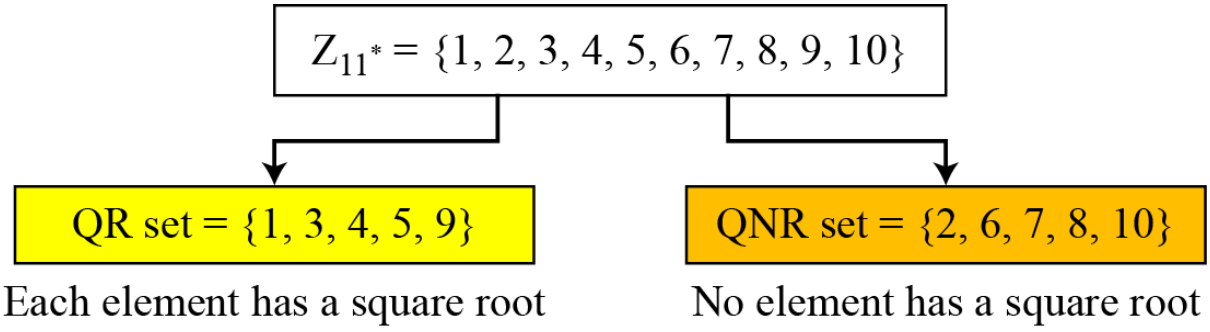
Example: The equation x2 ≡ 2 (mod 11) has no solution. No integer x can be found such that its square is 2 mod 11.

Quadratic Residues and Nonresidue:

In the equation x2 ≡ a (mod p), a is called a quadratic residue (QR) if the equation has two solutions; a is called quadratic nonresidue (QNR) if the equation has no solutions.

There are 10 elements in Z11*. Exactly five of them are quadratic residues and five of them are nonresidues. In other words, Z11* is divided into two separate sets, QR and QNR, as shown in Figure 9.4.

Figure 9.4 Division of Z11* elements into QRs and QNRs



Euler's Criterion

a. If $a^{(p-1)/2} \equiv 1 \pmod{p}$, a is a quadratic residue modulo p.

b. If $a^{(p-1)/2} \equiv -1 \pmod{p}$, a is a quadratic nonresidue modulo p.

Example

To find out if 14 or 16 is a QR in Z23*, we calculate:

$$14^{(23-1)/2} \bmod 23 \rightarrow 22 \bmod 23 \rightarrow -1 \bmod 23 \text{ nonresidue}$$

$$16^{(23-1)/2} \bmod 23 \rightarrow 16^{11} \bmod 23 \rightarrow 1 \bmod 23 \text{ residue}$$

Solving Quadratic Equation Modulo a Prime

Special Case: p = 4k + 3

$$x \equiv a^{(p+1)/4} \pmod{p} \quad \text{and} \quad x \equiv -a^{(p+1)/4} \pmod{p}$$

Example

Solve the following quadratic equations:

a. $x^2 \equiv 3 \pmod{23}$

b. $x^2 \equiv 2 \pmod{11}$

c. $x^2 \equiv 7 \pmod{19}$

Solutions:

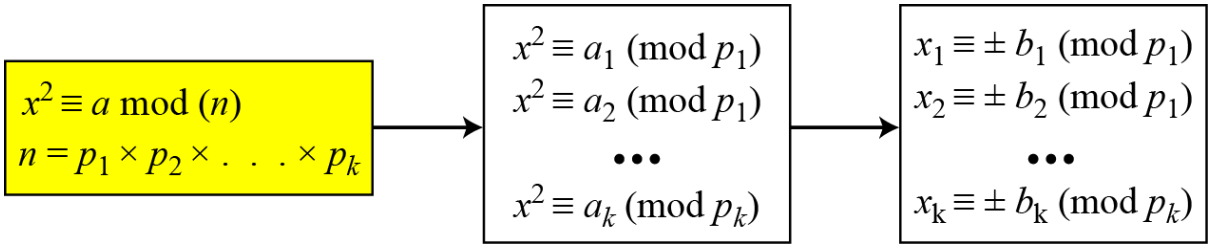
a. $x \equiv \pm 16 \pmod{23}$ $\sqrt{3} \equiv \pm 16 \pmod{23}$.

b. There is no solution for $\sqrt{2}$ in Z11.

c. $x \equiv \pm 11 \pmod{19}$. $\sqrt{7} \equiv \pm 11 \pmod{19}$.

2 Quadratic Congruence Modulo a Composite

Figure 9.5 Decomposition of congruence modulo a composite



Example: Assume that x2 ≡ 36 (mod 77). We know that 77 = 7 × 11. We can write

$$x^2 \equiv 36 \pmod 7 \equiv 1 \pmod 7 \qquad \text{and} \qquad x^2 \equiv 36 \pmod{11} \equiv 3 \pmod{11}$$

The answers are x ≡ +1 (mod 7), x ≡ − 1 (mod 7), x ≡ + 5 (mod 11), and x ≡ − 5 (mod 11). Now we can make four sets of equations out of these:

$$\textbf{Set 1: } x \equiv +1 \pmod 7 \qquad x \equiv +5 \pmod{11}$$
$$\textbf{Set 2: } x \equiv +1 \pmod 7 \qquad x \equiv -5 \pmod{11}$$
$$\textbf{Set 3: } x \equiv -1 \pmod 7 \qquad x \equiv +5 \pmod{11}$$
$$\textbf{Set 4: } x \equiv -1 \pmod 7 \qquad x \equiv -5 \pmod{11}$$

The answers are x = ± 6 and ± 27.

Complexity

How hard is it to solve a quadratic congruence modulo a composite? The main task is the factorization of the modulus. In other words, the complexity of solving a quadratic congruence modulo a composite is the same as factorizing a composite integer. As we have seen, if n is very large, factorization is infeasible.

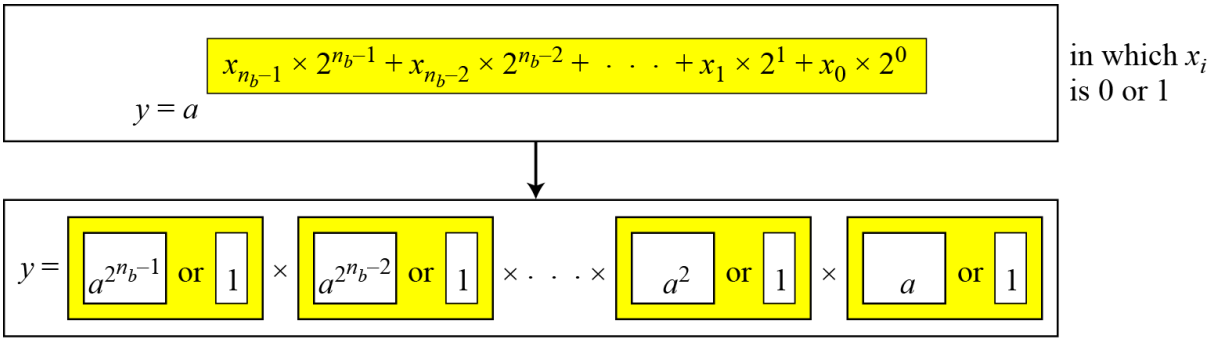
Solving a quadratic congruence modulo a composite is as hard as factorization of the modulus.

6.EXPONENTIATION AND LOGARITHM

$$\textbf{Exponentiation: } y = a^x \quad \rightarrow \quad \textbf{Logarithm: } x = \log_a y$$

6.1 Exponentiation:

Figure 9.6 The idea behind the square-and-multiply method



Example:

$$y = a^9 = a^{1001_2} = a^8 \times 1 \times 1 \times a$$

Algorithm 9.7 *Pseudocode for square-and-multiply algorithm*

```
Square_and_Multiply (a, x, n)
{
    y ← 1
    for (i ← 0 to n_b − 1)                  // n_b is the number of bits in x
    {
        if (x_i = 1)   y ← a × y  mod n     // multiply only if the bit is 1

        a ← a^2 mod n                       // squaring is not needed in the last iteration
    }
    return y
}
```

Figure 9.7 shows the process for calculating y = ax using the Algorithm 9.7 (for simplicity, the modulus is not shown). In this case, x = 22 = (10110)2 in binary. The exponent has five bits.

Figure 9.7 Demonstration of calculation of a22 using square-and-multiply method

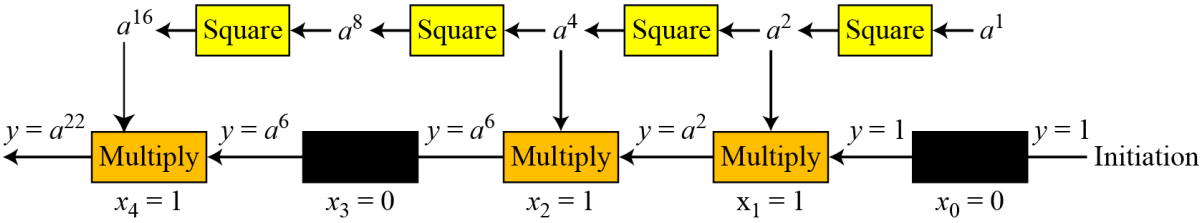


Table 9.3 *Calculation of 17^{22} mod 21*

i	x_i	*Multiplication (Initialization: $y = 1$)*		*Squaring (Initialization: $a = 17$)*
0	0		→	$a = 17^2 \bmod 21 = 16$
1	1	$y = 1 \times 16 \bmod 21 = 16$	→	$a = 16^2 \bmod 21 = 4$
2	1	$y = 16 \times 4 \bmod 21 = 1$	→	$a = 4^2 \bmod 21 = 16$
3	0		→	$a = 16^2 \bmod 21 = 4$
4	1	$y = 1 \times 4 \bmod 21 = 4$	→	

How about 2124 mod 8?

6.2 Logarithm:

In cryptography, we also need to discuss modular logarithm

Exhaustive Search

Algorithm 9.8 *Exhaustive search for modular logarithm*

```
Modular_Logarithm (a, y, n)
{
    for (x = 1 to n −1)                          // k is the number of bits in x
    {
        if (y ≡ a^x mod n) return x
    }
    return failure
}
```

Table 8.3 Powers of Integers, Modulo 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

Order of the Group.

Example

What is the order of group G = $<Z_{21}*, \times>$? $|G| = \phi(21) = \phi(3) \times \phi(7) = 2 \times 6 = 12$. There are 12 elements in this group: 1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, and 20. All are relatively prime with 21.

Order of an Element

Find the order of all elements in G = <Z10∗, ×>.

Solution

This group has only ϕ(10) = 4 elements: 1, 3, 7, 9. We can find the order of each element by trial and error.

a. $1^1 \equiv 1 \bmod (10) \rightarrow \text{ord}(1) = 1.$
b. $3^4 \equiv 1 \bmod (10) \rightarrow \text{ord}(3) = 4.$
c. $7^4 \equiv 1 \bmod (10) \rightarrow \text{ord}(7) = 4.$
d. $9^2 \equiv 1 \bmod (10) \rightarrow \text{ord}(9) = 2.$

Euler's Theorem

Example

Table 9.4 *Finding the orders of elements in Example 9.48*

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$
$a = 1$	**x: 1**	x: 1	x: 1	x: 1	x: 1	x: 1	x: 1
$a = 3$	x: 3	**x: 1**	x: 3	x: 1	x: 3	x: 1	x: 3
$a = 5$	x: 5	**x: 1**	x: 5	x: 1	x: 5	x: 1	x: 5
$a = 7$	x: 7	**x: 1**	x: 7	x: 1	x: 7	x: 1	x: 7

***Primitive Roots* In the group G = <Z_n∗, ×>, when the order of an element is the same as ϕ(n), that element is called the primitive root of the group.**

Example:

Table 9.4 shows that there are no primitive roots in G = <Z_8∗, ×> because no element has the order equal to ϕ(8) = 4. The order of elements are all smaller than 4.

Table 9.5 shows the result of $a^i \equiv x \pmod 7$ for the group G = <Z_7∗, ×>. In this group, ϕ(7) = 6.

Table 9.5 *Example 9.50*

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$
$a = 1$	x: 1	x: 1	x: 1	x: 1	x: 1	x: 1
$a = 2$	x: 2	x: 4	x: 1	x: 2	x: 4	x: 1
Primitive root → $a = 3$	x: 3	x: 2	x: 6	x: 4	x: 5	x: 1
$a = 4$	x: 4	x: 2	x: 1	x: 4	x: 2	x: 1
Primitive root → $a = 5$	x: 5	x: 4	x: 6	x: 2	x: 3	x: 1
$a = 6$	x: 6	x: 1	x: 6	x: 1	x: 6	x: 1

The group $G = <Z_n^*, \times>$ has primitive roots only if n is 2, 4, p^t, or $2p^t$.

Example:

For which value of n, does the group $G = <Z_n^*, \times>$ have primitive roots: 17, 20, 38, and 50?

Solution

a. $G = <Z_{17}^*, \times>$ has primitive roots, 17 is a prime.
b. $G = <Z_{20}^*, \times>$ has no primitive roots.
c. $G = <Z_{38}^*, \times>$ has primitive roots, $38 = 2 \times 19$ prime.
d. $G = <Z_{50}^*, \times>$ has primitive roots, $50 = 2 \times 5^2$ and 5 is a prime.

If the group $G = <Z_n^*, \times>$ has any primitive root, the number of primitive roots is $\phi(\phi(n))$.

Cyclic Group ***If g is a primitive root in the group, we can generate the set Z_n^* as $Z_n^* = \{g^1, g^2, g^3, \ldots, g^{\phi(n)}\}$***

Example

The group $G = <Z_{10}^*, \times>$ has two primitive roots because $\phi(10) = 4$ and $\phi(\phi(10)) = 2$. It can be found that the primitive roots are 3 and 7. The following shows how we can create the whole set Z_{10}^* using each primitive root.

$g = 3 \rightarrow$	$g^1 \bmod 10 = 3$	$g^2 \bmod 10 = 9$	$g^3 \bmod 10 = 7$	$g^4 \bmod 10 = 1$
$g = 7 \rightarrow$	$g^1 \bmod 10 = 7$	$g^2 \bmod 10 = 9$	$g^3 \bmod 10 = 3$	$g^4 \bmod 10 = 1$

The group $G = <Z_n^*, \times>$ is a cyclic group if it has primitive roots.
The group $G = <Z_p^*, \times>$ is always cyclic.

The idea of Discrete Logarithm

Properties of $G = \langle Z_p^*, \times \rangle$:

1. ***Its elements include all integers from 1 to $p - 1$.***

2. ***It always has primitive roots.***

3. ***It is cyclic. The elements can be created using g^x where x is an integer from 1 to $\phi(n) = p - 1$.***
4. ***The primitive roots can be thought as the base of logarithm.***

Solution to Modular Logarithm Using Discrete Logs

Tabulation of Discrete Logarithms

Table 9.6 *Discrete logarithm for* $\mathbf{G} = \langle \mathbf{Z}_7^*, \times \rangle$

y	1	2	3	4	5	6
$x = L_3\, y$	6	2	1	4	5	3
$x = L_5\, y$	6	4	5	2	1	3

Example:

Find x in each of the following cases:

a. $\mathbf{4 \equiv 3^x \pmod 7}$.

b. $\mathbf{6 \equiv 5^x \pmod 7}$.

Solution

We can easily use the tabulation of the discrete logarithm in Table 9.6.

a. $\mathbf{4 \equiv 3^x \bmod 7 \rightarrow x = L_3 4 \bmod 7 = 4 \bmod 7}$

b. $\mathbf{6 \equiv 5^x \bmod 7 \rightarrow x = L_5 6 \bmod 7 = 3 \bmod 7}$

Table 9.7 *Comparison of traditional and discrete logarithms*

Traditional Logarithm	*Discrete Logarithms*
$\log_a 1 = 0$	$L_g 1 \equiv 0 \pmod{\phi(n)}$
$\log_a (x \times y) = \log_a x + \log_a y$	$L_g(x \times y) \equiv (L_g x + L_g y) \pmod{\phi(n)}$
$\log_a x^k = k \times \log_a x$	$L_g x^k \equiv k \times L_g x \pmod{\phi(n)}$

The discrete logarithm problem has the same complexity as the factorization problem

UNIT-3(PART-II)

PUBLIC KEY CRYPTOGRAPHY

PUBLIC KEY CRYPTOSYSTEM PRINCIPLES:

- Public key cryptography also Known as **asymmetric cryptography**.
- Public-key systems depend on the use of mathematical functions.
- The concept of public key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption.
    - The Key Exchange Problem
    - The Trust Problem

    The Key Exchange Problem: The key exchange problem arises from the fact that communicating parties must somehow share a secret key before any secure communication can be initiated, and both parties must then ensure that the key remains secret. Of course, direct key exchange is not always feasible due to risk, inconvenience, and cost factors.

    The Trust Problem: Ensuring the integrity of received data and verifying the identity of the source of that data can be very important. Means in the symmetric key cryptography system, receiver doesn't know whether the message is coming for particular sender.
- This public key cryptosystem uses two keys as pair for encryption of plain text and Decryption of cipher text.
- These two keys are names as "**Public key**" and "**Private key**". The private key is kept secret whereas public key is distributed widely.
- A message or text data which is encrypted with the public key can be decrypted only with the corresponding private-key
- This two key system very useful in the areas of confidentiality (secure) and authentication

A **public-key encryption** scheme has **six ingredients**		
1	**Plaintext**	This is the readable message or data that is fed into the algorithm as input.
2	**Encryption algorithm**	The encryption algorithm performs various transformations on the plaintext.

3	**Public key**	This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input
4	**Private key**	
5	**Ciphertext**	This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
6	**Decryption algorithm**	This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

Public key cryptography for providing confidentiality (secrecy)

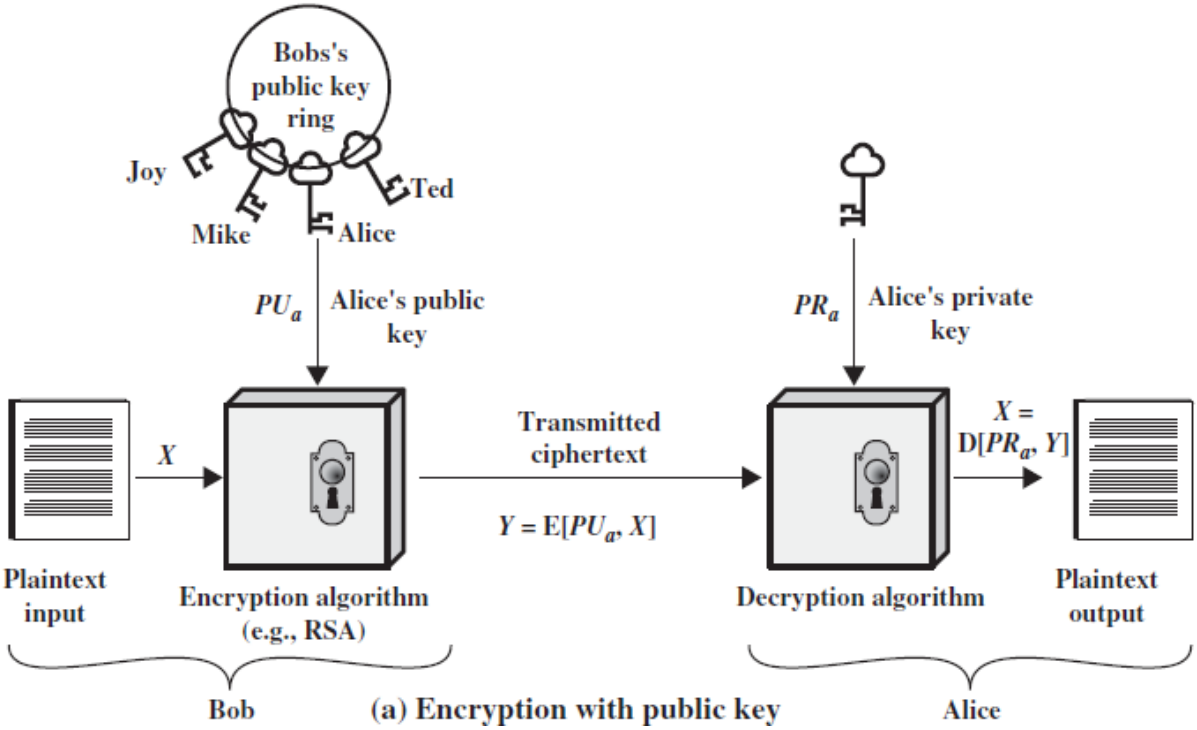


Figure 9.1

The essential steps are the following.

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key.The companion key is kept private. As Figure 9.1a suggests, each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.

4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

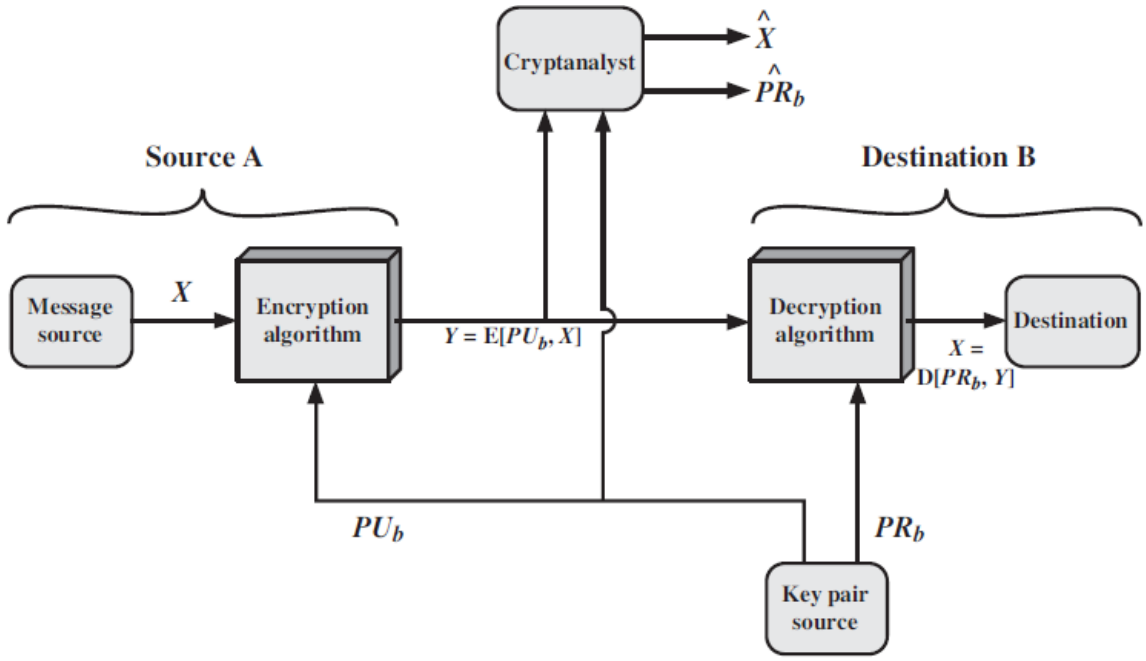


Figure 9.2 Public-Key Cryptosystem: Secrecy

There is some source A that produces a message in plaintext ***X* = [*X*1, *X*2, . . . ,X*M*].**

The ***M***elements of ***X***are letters in some finite alphabet. The message is intended for destination **B**.

B generates a related pair of keys: a public key, PU_b, and a private key, PR_b.

$PR_b$is known only to B, whereas $PU_b$is publicly available and therefore accessible by A.

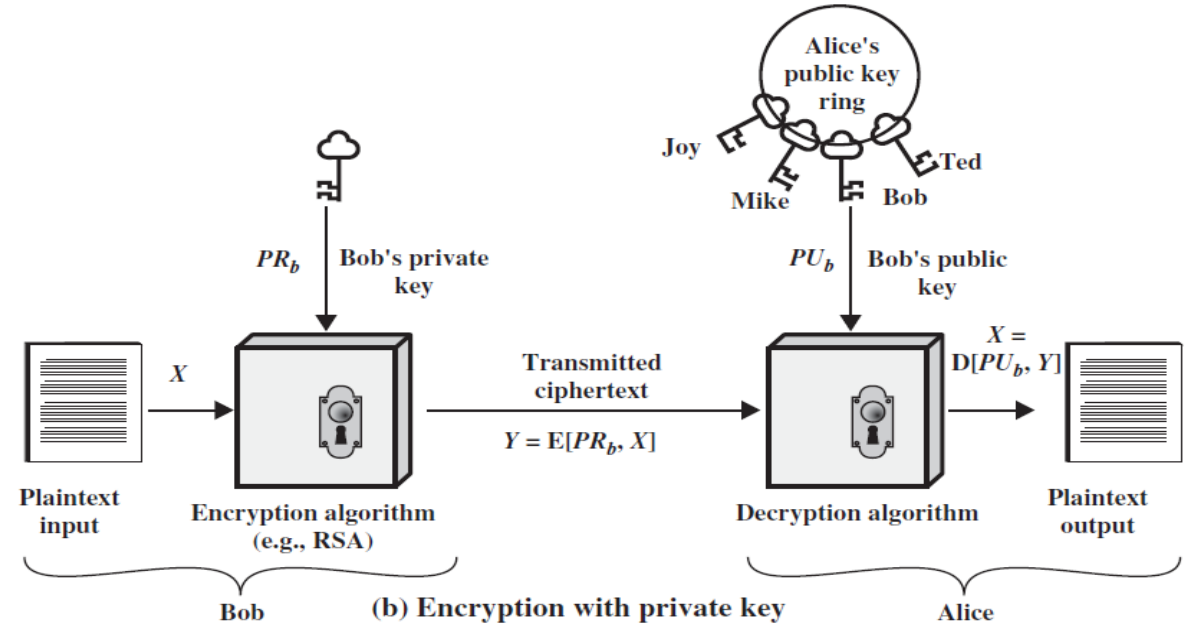
With the message *X* and the encryption key PU_bas input, A forms the ciphertext$Y = [Y1, Y2, . . ,Y_N]$:

$$Y = \mathrm{E}(PU_b, X)$$

The intended receiver, in possession of the matching private key, is able to invert the transformation:

$$X = \mathrm{D}(PR_b, Y)$$

Public key cryptography for proving Authentication:



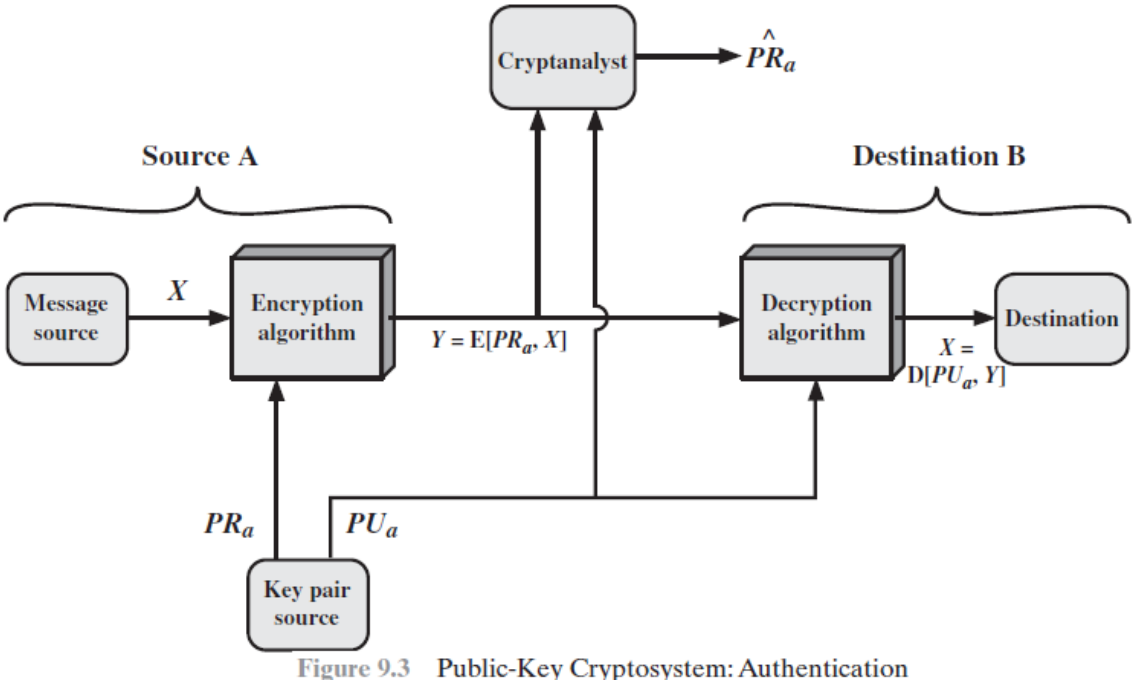


Figure 9.3 Public-Key Cryptosystem: Authentication

The above diagrams show the use of public-key encryption to provide authentication:

$$Y = \mathrm{E}(PR_a, X)$$

$$X = \mathrm{D}(PU_a, Y)$$

- In this case,A prepares a message to B and encrypts it using A's private key before transmitting it. B can decrypt the message using A's public key. Because the message was encrypted using A's private key, only A could have prepared the message. Therefore, the entire encrypted message serves as a **digital signature.**

- It is impossible to alter the message without access to A's private key, so the message is authenticated both in terms of source and in terms of data integrity.

Applications for Public-Key Cryptosystems

Public-key systems are characterized bythe use of a cryptographic algorithm with two keys, one held private and one availablepublicly. Depending on the application, the sender uses either the sender'sprivate key or the receiver's public key, or both, to perform some type of cryptographicfunction.

We can classify the use of public-key cryptosystems into three categories

• Encryption /decryption: The sender encrypts a message with the recipient'spublic key.

• Digital signature: The sender "signs" a message with its private key. Signing isachieved by a cryptographic algorithm applied to the message or to a smallblock of data that is a function of the message.

• Key exchange: Two sides cooperate to exchange a session key. Several differentapproaches are possible, involving the private key(s) of one or both parties.

Applications for Public-Key Cryptosystems

Algorithm	**Encryption/Decryption**	**Digital Signature**	**Key Exchange**
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Public-Key Cryptanalysis

- As with symmetric encryption, a public-key encryption scheme is vulnerable to a brute-force attack. The countermeasure is the same: Use large keys
- Public-key systems depend on the use of some sort of invertible mathematical function.

- Thus, the key size must be large enough to make brute-force attack impractical but small enough for practical encryption and decryption. In practice, the key sizes that have been proposed do make brute-force attack impractical but result in encryption/decryption speeds that are too slow for general-purpose use.
- Instead, as was mentioned earlier, public-key encryption is currently confined to key management and signature applications.

RSA ALGORITHM.

- It is one of the most common public key algorithm.
- It was first published byRivest(R), Shamir (S) and Adleman (A) in the year 1977.
- The RSA scheme is a block cipher in which the plaintext &ciphertext are integers between 0 and n-1 for some 'n'.
- A typical size for 'n' is 1024 bits or 309 decimal digits. That is, n is less than 2^{1024}

Description of the Algorithm:

- RSA algorithm uses an expression with exponentials.
- In RSA plaintext is encrypted in blocks, with each block having a binary value less than some number n. that is, the block size must be less than or equal to **$\log_2(n)$**
- **RSA** uses two exponents 'e' and'd' where e→public and d→private.
- Encryption and decryption are of following form, for some PlainText 'M' and Ciphertext block 'C'

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

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

$$M = C^d \bmod = (M^e \bmod n)^d \text{ mon } n = (M^e)^d \bmod n = M^{ed} \bmod n$$

Both sender and receiver must know the value of n.

The sender knows the value of 'e' & only the reviver knows the value of 'd' thus this is a public key encryption algorithm with a

Public key PU={e, n}

Private key PR={d, n}

Requirements:

The RSA algorithm to be satisfactory for public key encryption, the following requirements must be met:

1. It is possible to find values of e, d n such that " $\mathbf{M^{ed} \bmod n = M}$ " for all M<n
2. It is relatively easy to calculate " $\mathbf{M^{e} \bmod n}$ **"** and " $\mathbf{C^{d} \bmod n}$ "for M<n
3. It is infeasible to determine "d" given 'e' & 'n'. The " $\mathbf{M^{ed} \bmod n = M}$ "

Then the relation between 'e' & 'd' can be expressed as

$$\mathbf{ed \bmod \emptyset(n)=1}$$

this is equivalent to saying

$$\mathbf{ed \equiv 1 \bmod \emptyset(n)}$$

$$\mathbf{d \equiv e^{-1} \bmod \emptyset(n)}$$

That is 'e' and 'd' are multiplicative inverses mod Ø(n).

Steps of RSA algorithm:

Step 1→Select 2 prime numbers p & q

Step 2→Calculate n=pq

Step 3→Calculate Ø(n)=(p-1)(q-1)

Step 4→ Select or find integer e (public key) which is relatively prime to Ø(n).

ie., gcd (Ø(n), e)=1 where 1<e< Ø(n).

Step 5→ Calculate "d" (private key) by using following condition. $\mathbf{ed \equiv 1 \bmod \emptyset(n)}$

d<Ø(n).

Step 6→ Perform encryption by using $C = M^e \bmod n$

Step 7→ perform Decryption by using $M = C^d \bmod n$

Figure 9.5 summarizes the RSA algorithm. If Bob wants to send a message to Alice,Alice generates a public/private key pair; Bob encrypts using Alice's public key; and Alice decrypts using her private key.

Key Generation Alice	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p-1)(q-1)$	
Select integer e	gcd $(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Public Key	
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Figure 9.5 The RSA Algorithm

Example:

1. Select two prime numbers, p = 17 and q = 11.

2. Calculate $n = pq$= 17 × 11 = 187.

3. Calculate Ø(n) = (p - 1)(q - 1) = 16 × 10 = 160.

4. Select e such that e is relatively prime to Ø(n) = 160 and less than Ø (n); we choose e = 7.

5. Determine d such that ***de* ≡1 (mod 160)** and d < 160.The correct value is d = 23, because 23 * 7 = 161 = (1 × 160) + 1; d can be calculated using the extended Euclid's algorithm

The resulting keys are public key PU = {7, 187} and private key PR = {23, 187}.

The example shows the use of these keys for a plaintext input of M= 88. For encryption, we need to calculate $C = 88^7 \bmod 187$. Exploiting the properties of modular arithmetic, we can do this as follows.

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$

$$88^1 \bmod 187 = 88$$

$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$

$$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$$

$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$$

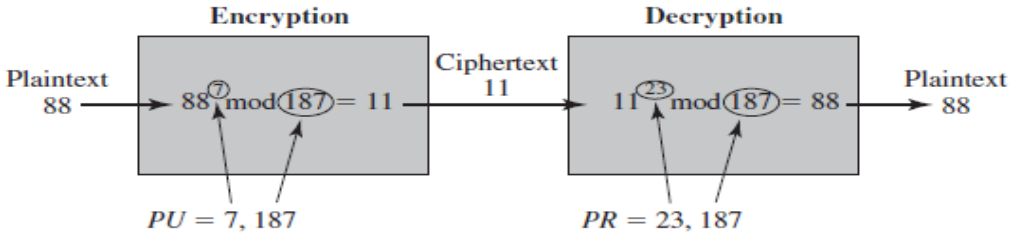


Figure 9.6 Example of RSA Algorithm

For decryption, we calculate $M = 11^{23} \bmod 187$:

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$

$$11^1 \bmod 187 = 11$$

$$11^2 \bmod 187 = 121$$

$$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$$

$$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$$

$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$$

Example 2:

- Choose p = 3 and q = 11
- Compute n = p * q = 3 * 11 = 33
- Compute φ(n) = (p - 1) * (q - 1) = 2 * 10 = 20
- Choose e such that 1 < e <φ(n) and e and n are coprime. Let e = 7
- Compute a value for d. One solution is d = 3 [(3 * 7) % 20 = 1]
- Public key is (e, n) => (7, 33)
- Private key is (d, n) => (3, 33)
- Consider the plain text **m=2**
- The encryption of m = 2 is c = 2^7 mod33= 29
- The decryption of c = 29 is m = 29^3mod33 **= 2**

The Security of RSA

There are three main approaches of attacking RSA algorithm.

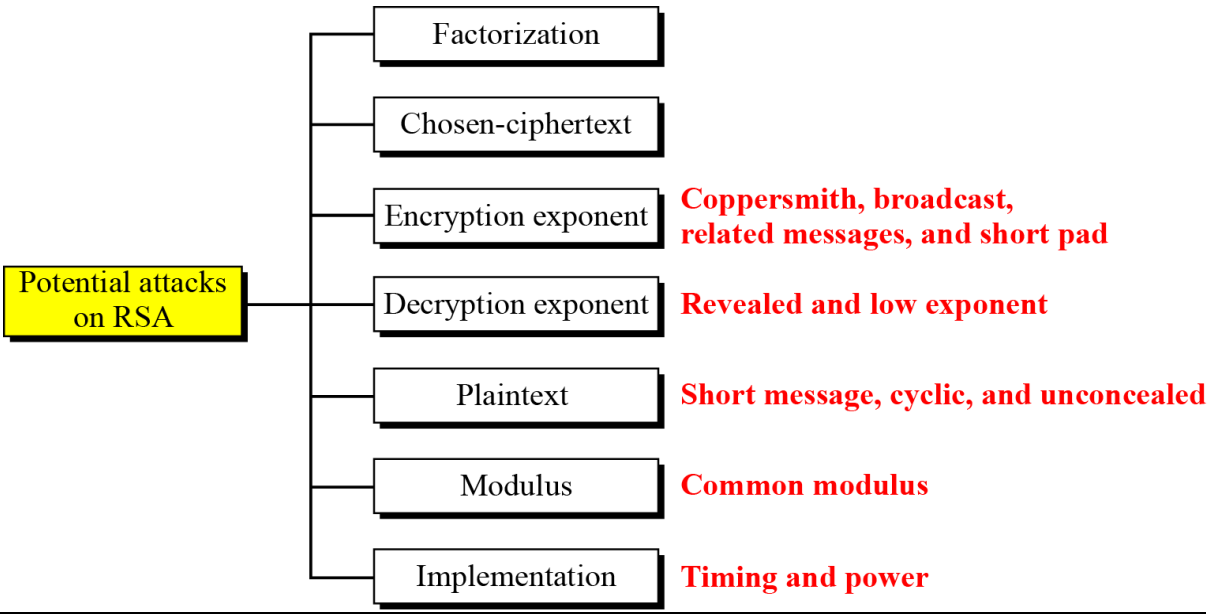
• **Brute force:** This involves trying all possible private keys.

• **Mathematical attacks:** There are several approaches, all equivalent in effort to factoring the product of two primes.

• **Timing attacks:** These depend on the running time of the decryption algorithm.

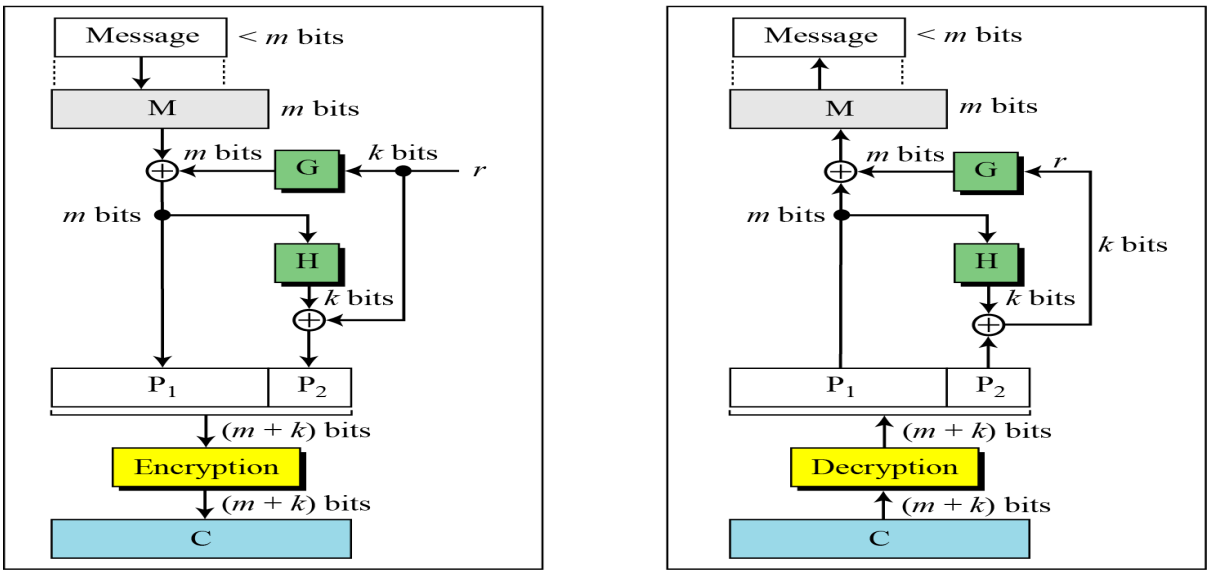
• **Chosen ciphertext attacks:** This type of attack exploits properties of the RSA algorithm.

Taxonomy of potential attacks on RSA:



OAEP(Optimal asymmetric encryption padding (OAEP):

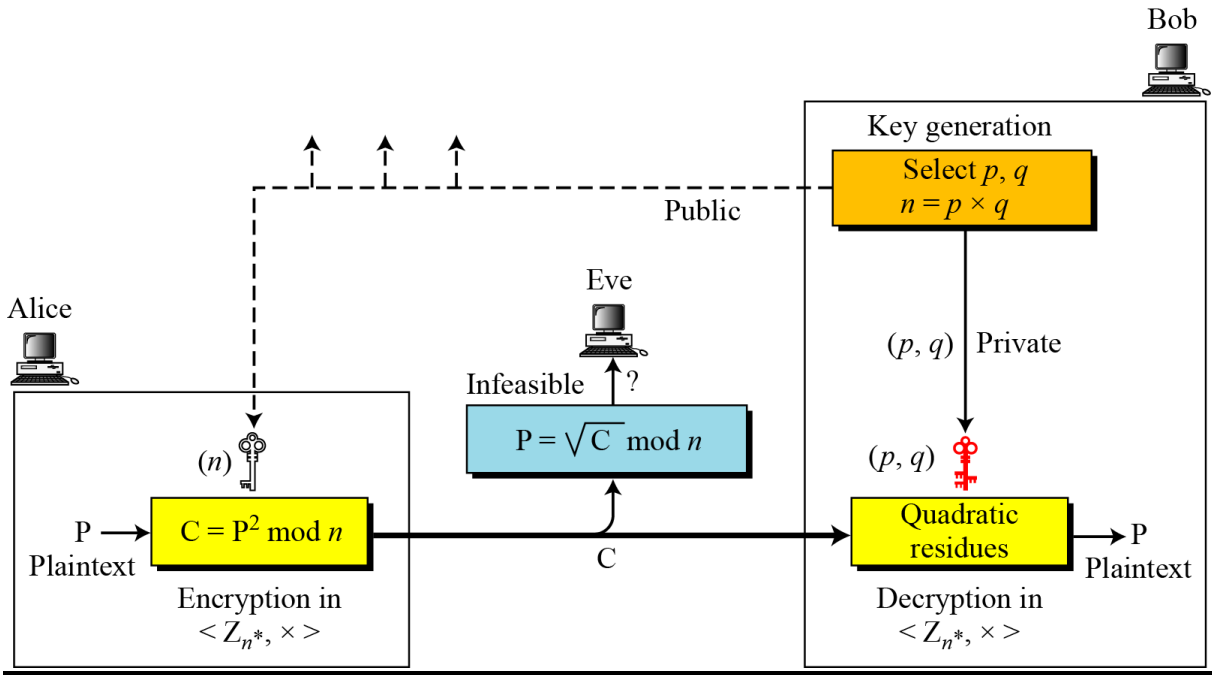
M: Padded message
r: One-time random number
P: Plaintext ($P_1 \| P_2$)
C: Ciphertext
G: Public function (k-bit to m-bit)
H: Public function (m-bit to k-bit)



2. RABIN CRYPTOSYSTEM:

The Rabin cryptosystem can be thought of as an RSA cryptosystem in which the value of e and d are fixed. The encryption is $C \equiv P^2 \pmod{n}$ and the decryption is $P \equiv C^{1/2} \pmod{n}$.

Figure 10.10 Rabin cryptosystem



Key Generation

Algorithm 10.6 *Key generation for Rabin cryptosystem*

Rabin_Key_Generation

{

Choose two large primes p and q in the form $4k + 3$ and $p \neq q$.

$n \leftarrow p \times q$

Public_key $\leftarrow$ n // To be announced publicly

Private_key $\leftarrow$ (q, n) // To be kept secret

return Public_key and Private_key

}

Encryption

Algorithm 10.7 *Encryption in Rabin cryptosystem*

```
Rabin_Encryption (n, P)          // n is the public key; P is the ciphertext from Z_n*
{
    C ← P^2 mod n                // C is the ciphertext
    return C
}
```

Decryption

Algorithm 10.8 *Decryption in Rabin cryptosystem*

```
Rabin_Decryption (p, q, C)       // C is the ciphertext; p and q are private keys
{
    a1 ← +(C^((p+1)/4)) mod p
    a2 ← −(C^((p+1)/4)) mod p
    b1 ← +(C^((q+1)/4)) mod q
    b2 ← −(C^((q+1)/4)) mod q
    // The algorithm for the Chinese remainder algorithm is called four times.
    P1 ← Chinese_Remainder (a1, b1, p, q)
    P2 ← Chinese_Remainder (a1, b2, p, q)
    P3 ← Chinese_Remainder (a2, b1, p, q)
    P4 ← Chinese_Remainder (a2, b2, p, q)
    return P1, P2, P3, and P4
}
```

The Rabin cryptosystem is not deterministic: Decryption creates four plaintexts.

Example

Here is a very trivial example to show the idea.

1. **Bob selects $p = 23$ and $q = 7$. Note that both are congruent to 3 mod 4.**
2. **Bob calculates $n = p \times q = 161$.**
3. **Bob announces n publicly; he keeps p and q private.**
4. **Alice wants to send the plaintext $P = 24$. Note that 161 and 24 are relatively prime; 24 is in Z_{161}^*. She calculates $C = 24^2 = 93 \mod 161$, and sends the ciphertext 93 to Bob.**

---

5. **Bob receives 93 and calculates four values:**
   $a_1 = +(93^{(23+1)/4}) \bmod 23 = 1 \bmod 23$
   $a_2 = -(93^{(23+1)/4}) \bmod 23 = 22 \bmod 23$
   $b_1 = +(93^{(7+1)/4}) \bmod 7 = 4 \bmod 7$
   $b_2 = -(93^{(7+1)/4}) \bmod 7 = 3 \bmod 7$
6. **Bob takes four possible answers, (a_1, b_1), (a_1, b_2), (a_2, b_1), and (a_2, b_2), and uses the Chinese remainder theorem to find four possible plaintexts: 116, 24, 137, and 45. Note that only the second answer is Alice's plaintext.**

DIFFIE-HELLMAN KEY EXCHANGE:

- Diffie-Hellman key exchange is the first published public key algorithm
- This Diffie-Hellman key exchange protocol is also known as exponential key agreement. And it is based on mathematical principles.
- The purpose of the algorithm is to enable two users to exchange a key securely that can then be used for subsequent encryption of messages.
- This algorithm itself is limited to exchange of the keys.
- This algorithm depends for its effectiveness on the difficulty of computing discrete logarithms.
- The discrete logarithms are defined in this algorithm in the way of define a primitive root of a prime number.
    - Primitive root: we define a primitive root of a prime number P as one whose power generate all the integers form 1 to P-1 that is if 'a' is a primitive root of the prime number P, then the numbers

      $a \bmod P, a^2 \bmod P, a^3 \bmod P, \ldots\ldots a^{P-1} \bmod P$ are distinct and consist of the integers form 1 through P-1 in some permutation.

      For any integer 'b' and 'a', here 'a' is a primitive root of prime number P, then

      $b \equiv a^i \bmod P \qquad 0 \leq i \leq (P-1)$

The exponent i→is refer as discrete logarithm or index of b for the base a, mod P.

The value denoted as **$\mathbf{ind}_{a,p}(b)$**

<u>Algorithm for Diffie-Hellman Key Exchange</u>:

Step 1→consider two publicly known numbers q, α

q→ Prime number

α→ primitive root of q and α<q.

Step 2 → if A & B users wish to exchange a key

a) User A select a random integer X_A<q and computes

$$Y_A = \alpha^{X_A} \bmod q$$

b) User B independently select a random integer X_B<q and computes

$$Y_B = \alpha^{X_B} \bmod q$$

c) Each side keeps the X value private and Makes the Y value available publicly to the outer side.

Step 3→ UserA Computes the key as

$$K = (Y_B)^{X_A} \bmod q$$

User B Computes the key as

$$K = (Y_A)^{X_B} \bmod q$$

Step 4→ two calculation produce identical results

$$K = (Y_B)^{X_A} \bmod q$$

$$K = (\alpha^{X_B} \bmod q)^{X_A} \bmod q \quad \text{(We know that } Y_B = \alpha^{X_B} \bmod q \text{)}$$

$$= (\alpha^{X_B})^{X_A} \bmod q$$

$$= (\alpha^{X_A})^{X_B} \bmod q$$

$$= (\alpha^{X_A} \bmod q)^{X_B} \bmod q$$

$$= (Y_A)^{X_B} \bmod q \quad \text{(We know that } Y_A = \alpha^{X_A} \bmod q \text{)}$$

The result is that the two sides have exchanged a secret key.

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

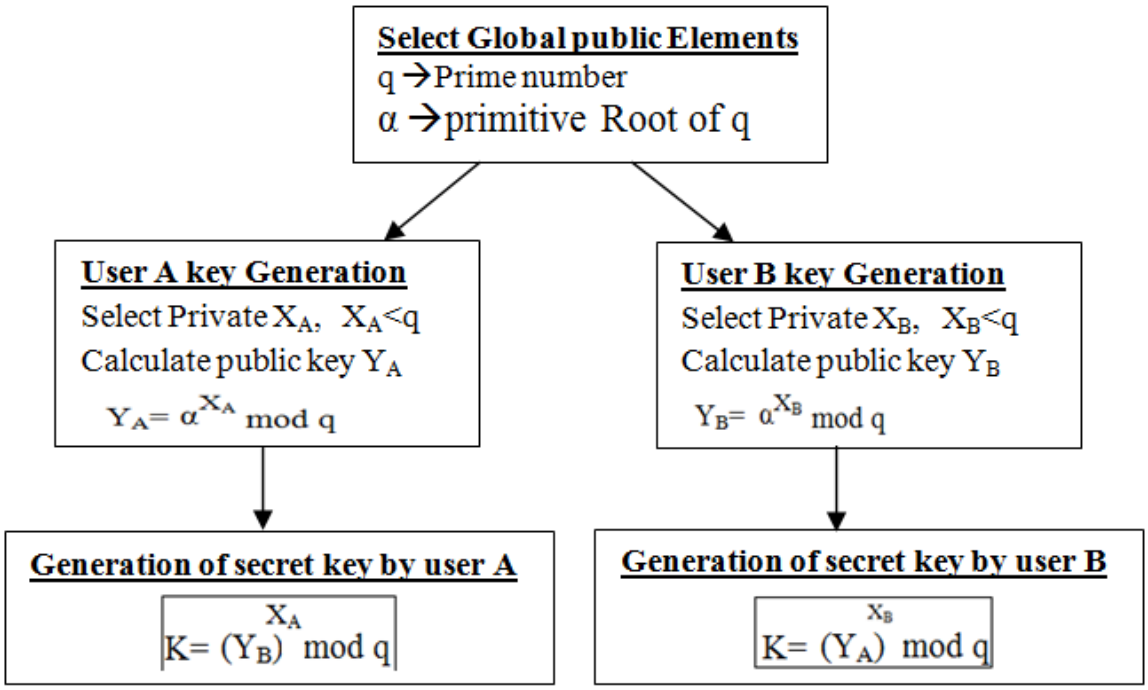
User A Key Generation	
Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{XA} \bmod q$

User B Key Generation	
Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{XB} \bmod q$

Calculation of Secret Key by User A
$K = (Y_B)^{XA} \bmod q$

Calculation of Secret Key by User B
$K = (Y_A)^{XB} \bmod q$

Figure 10.1 The Diffie-Hellman Key Exchange Algorithm



Example:

Here is an example. Key exchange is based on the use of the prime number $q = 353$ and a primitive root of 353, in this case $\alpha = 3$. A and B select secret keys $X_A = 97$ and $X_B = 233$, respectively. Each computes its public key:

A computes $Y_A = 3^{97} \bmod 353 = 40$.

B computes $Y_B = 3^{233} \bmod 353 = 248$.

After they exchange public keys, each can compute the common secret key:

A computes $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$.

B computes $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$.

ELGAMAL CRYPTOGRAPHIC SYSTEM

- Elgamalis a a public-key scheme based on discrete logarithms,
- It is closely related to the Diffie-Hellman technique
- It is used in digital signature standard (DSS), and the S/MIME e-mail standard

As with Diffie-Hellman, the global elements of ElGamal are a prime number q and α, which is a primitive root of q. User A generates a private/public key pair as follows:

1. Generate a random integer X_A, such that $1 < X_A < q - 1$.
2. Compute $Y^A = \alpha^{X_A} \bmod q$.
3. A's private key is X_A; A's pubic key is $\{q, \alpha, Y_A\}$.

Any user B that has access to A's public key can encrypt a message as follows:

1. Represent the message as an integer M in the range $0 \leq M \leq q - 1$. Longer messages are sent as a sequence of blocks, with each block being an integer less than q.
2. Choose a random integer k such that $1 \leq k \leq q - 1$.
3. Compute a one-time key $K = (Y_A)^k \bmod q$.
4. Encrypt M as the pair of integers (C_1, C_2) where

$$C_1 = \alpha^k \bmod q; \quad C_2 = KM \bmod q$$

User A recovers the plaintext as follows:

1. Recover the key by computing $K = (C_1)^{X_A} \bmod q$.
2. Compute $M = (C_2 K^{-1}) \bmod q$.

These steps are summarized in Figure 10.3. It corresponds to Figure 9.1a: Alice generates a public/private key pair; Bob encrypts using Alice's public key; and Alice decrypts using her private key.

Let us demonstrate why the ElGamal scheme works. First, we show how K is recovered by the decryption process:

$K = (Y_A)^k \bmod q$	K is defined during the encryption process
$K = (\alpha^{X_A} \bmod q)^k \bmod q$	substitute using $Y_A = \alpha^{X_A} \bmod q$
$K = \alpha^{kX_A} \bmod q$	by the rules of modular arithmetic
$K = (C_1)^{X_A} \bmod q$	substitute using $C_1 = \alpha^k \bmod q$

Next, using K, we recover the plaintext as

$$C_2 = KM \bmod q$$

$$(C_2 K^{-1}) \bmod q = KMK^{-1} \bmod q = M \bmod q = M$$

We can restate the ElGamal process as follows, using Figure 10.3.

1. Bob generates a random integer k.
2. Bob generates a one-time key K using Alice's public-key components Y_A, q, and k.
3. Bob encrypts k using the public-key component α, yielding C_1. C_1 provides sufficient information for Alice to recover K.
4. Bob encrypts the plaintext message M using K.
5. Alice recovers K from C_1 using her private key.
6. Alice uses K^{-1} to recover the plaintext message from C_2.

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

Key Generation by Alice	
Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{XA} \bmod q$
Public key	$PU = \{q, \alpha, Y_A\}$
Private key	X_A

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

Decryption by Alice with Alice's Private Key	
Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{XA} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

Figure 10.3 The ElGamal Cryptosystem

For example, let us start with the prime field GF(19); that is, $q = 19$. It has primitive roots $\{2, 3, 10, 13, 14, 15\}$, as shown in Table 8.3. We choose $\alpha = 10$.

Alice generates a key pair as follows:

1. Alice chooses $X_A = 5$.
2. Then $Y_A = \alpha^{X_A} \bmod q = \alpha^5 \bmod 19 = 3$ (see Table 8.3).
3. Alice's private key is 5; Alice's pubic key is $\{q, \alpha, Y_A\} = \{19, 10, 3\}$.

Suppose Bob wants to send the message with the value $M = 17$. Then,

1. Bob chooses $k = 6$.
2. Then $K = (Y_A)^k \bmod q = 3^6 \bmod 19 = 729 \bmod 19 = 7$.
3. So
   $C_1 = \alpha^k \bmod q = \alpha^6 \bmod 19 = 11$
   $C_2 = KM \bmod q = 7 \times 17 \bmod 19 = 119 \bmod 19 = 5$
4. Bob sends the ciphertext $(11, 5)$.

For decryption:

1. Alice calculates $K = (C_1)^{X_A} \bmod q = 11^5 \bmod 19 = 161051 \bmod 19 = 7$.
2. Then K^{-1} in GF(19) is $7^{-1} \bmod 19 = 11$.
3. Finally, $M = (C_2 K^{-1}) \bmod q = 5 \times 11 \bmod 19 = 55 \bmod 19 = 17$.

ELLIPTIC CURVE CRYPTOGRAPHY

- **Definition: Elliptic curve cryptography (ECC)** is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. These are analogy of existing public key cryptosystem in which modular arithmetic is replaced by operations defined over elliptic curve.

ECC DIFFIE-HELLMAN KEY EXCHANGE:

ECC can do key exchange, that is analogous to Diffie Hellman.

Key exchange using elliptic curves can be done in the following manner.

First pick alarge integer q , which is either a prime number P or an integer of the form 2^m and elliptic curve parameters a & b for equation $y^2 \bmod p = (x^3 + ax + b) \bmod p$ or $y^2 + xy = x^3 + ax^2 + b$.

This define elliptic group of point $E_q(a,b)$.

Pick a base point G=(x1,y1) in $E_p(a,b)$ whose order is a very large value n.

The order n of a point G on an elliptic curve is the smallest +ve integer n such that $nG=0.E_q(a,b)$

Global Public Elements

$E_q(a, b)$	elliptic curve with parameters a, b, and q, where q is a prime or an integer of the form 2^m
G	point on elliptic curve whose order is large value n

User A Key Generation

Select private n_A	$n_A < n$
Calculate public P_A	$P_A = n_A \times G$

User B Key Generation

Select private n_B	$n_B < n$
Calculate public P_B	$P_B = n_B \times G$

Calculation of Secret Key by User A

$K = n_A \times P_B$

Calculation of Secret Key by User B

$K = n_B \times P_A$

Figure 10.7 ECC Diffie-Hellman Key Exchange

$$n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A$$

As an example,[6] take $p = 211$; $E_p(0, -4)$, which is equivalent to the curve $y^2 = x^3 - 4$; and $G = (2, 2)$. One can calculate that $240G = O$. A's private key is $n_A = 121$, so A's public key is $P_A = 121(2, 2) = (115, 48)$. B's private key is $nB = 203$, so B's public key is $203(2, 3) = (130, 203)$. The shared secret key is $121(130, 203) = 203(115, 48) = (161, 69)$.

Elliptic Curve Encryption/Decryption:

Several approaches to encryption/decryption using elliptic curves have been analyzed in the literature. In this subsection, we look at perhaps the simplest. The first task in this system is to encode the plaintext message m to be sent as an x–y point P_m. It is the point P_m that will be encrypted as a ciphertext and subsequently decrypted. Note that we cannot simply encode the message as the x or y coordinate of a point, because not all such coordinates are in $E_q(a, b)$; for example, see Table 10.1. Again, there are several approaches to this encoding, which we will not address here, but suffice it to say that there are relatively straightforward techniques that can be used.

As with the key exchange system, an encryption/decryption system requires a point G and an elliptic group $E_q(a, b)$ as parameters. Each user A selects a private key n_A and generates a public key $P_A = n_A \times G$.

To encrypt and send a message P_m to B, A chooses a random positive integer k and produces the ciphertext C_m consisting of the pair of points:

$$C_m = \{kG, P_m + kP_B\}$$

Note that A has used B's public key P_B. To decrypt the ciphertext, B multiplies the first point in the pair by B's secret key and subtracts the result from the second point:

$$P_m + kP_B - n_B(kG) = P_m + k(n_BG) - n_B(kG) = P_m$$

A has masked the message P_m by adding kP_B to it. Nobody but A knows the value of k, so even though P_b is a public key, nobody can remove the mask kP_B. However, A also includes a "clue," which is enough to remove the mask if one knows the private key n_B. For an attacker to recover the message, the attacker would have to compute k given G and kG, which is assumed to be hard.

As an example of the encryption process (taken from [KOBL94]), take $p = 751$; $E_p(-1, 188)$, which is equivalent to the curve $y^2 = x^3 - x + 188$; and $G = (0, 376)$. Suppose that A wishes to send a message to B that is encoded in the elliptic point $P_m = (562, 201)$ and that A selects the random number $k = 386$. B's public key is $P_{B = (201, 5)}$. We have $386(0, 376) = (676, 558)$, and $(562, 201) + 386(201, 5) = (385, 328)$. Thus, A sends the cipher text $\{(676, 558), (385, 328)\}$.

UNIT-IV

HASH FUNCTION:

It is a one of the authentication function; it accepts a variable size message M as input and produces a fixed size output.

A hash value 'h' is generated by a function H of the form

h=H (M)

M→ variable length message

H(M)→ fixed length hash value.

The hash code is also referred as Message Digest (MD) or hash value.

The main difference between Hash Function and MAC is a hash code does not use a key but is a function only of the input message.

The hash value is appended to the message at the source at a time when the message is assumed or known to be correct.

The receiver authenticates that message by re-computing the hash value.

Hash functions are often used to determine whether or not data has changed.

Figure 11.1 depicts the general operation of a cryptographic hash function

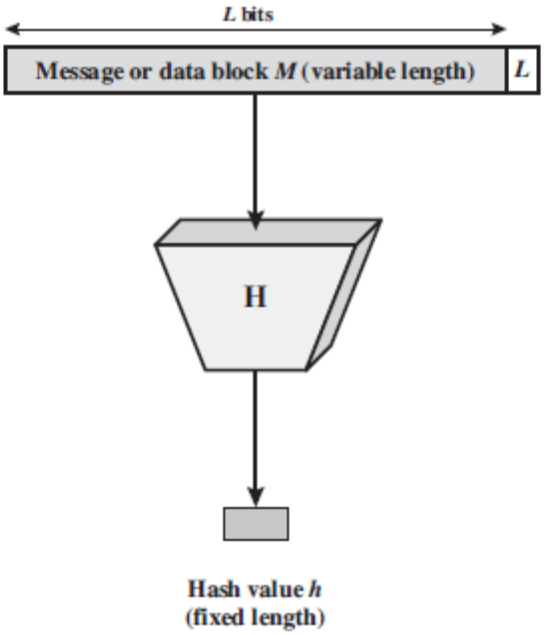


Figure 11.1 Black Diagram of Cryptographic Hash Function; $h = H(M)$

APPLICATIONS OF CRYPTOGRAPHIC HASH FUNCTIONS

It is used in a wide variety of security applications and Internet protocols

Message Authentication

Message authentication is a mechanism or service used to verify the integrity of a message. Message authentication assures that data received are exactly as sent (i.e., contain no modification, insertion, deletion, or replay)

When a hash function is used to provide message authentication, the hash function value is often referred to as a message digest.

Figure 11.2 illustrates a variety of ways in which a hash code can be used to provide message authentication, as follows.

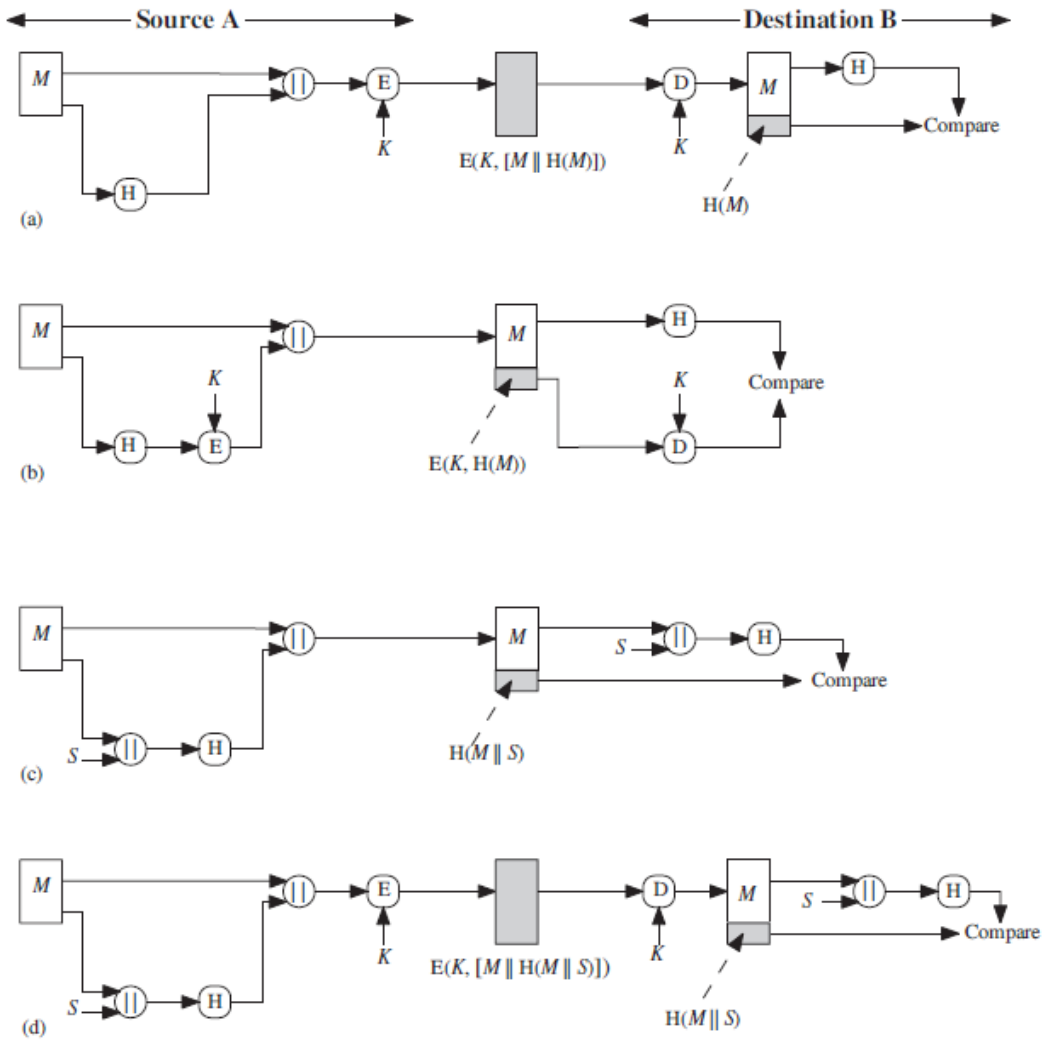


Figure 11.2 Simplified Examples of the Use of a Hash Function for Message Authentication

(a)The message plus concatenated hash code is encrypted using symmetric encryption. Because only A and B share the secret key, the message must have come from A and has not been altered.

The hash code provides the structure or redundancy required to achieve authentication. Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

(b)Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality

(c)It is possible to use a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value S.A computes the hash value over the concatenation of M and S and appends the resulting hash value to M. Because B possesses, it can recomputed the hash value to verify. Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.

(d)Confidentiality can be added to the approach of method (c) by encrypting the entire message plus the hash code.

Digital Signatures

Another important application, which is similar to the message authentication application, is the digital signature.

The operation of the digital signature is similar to that of the MAC. In the case of the digital signature, the hash value of a message is encrypted with a user's private key. Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signature.

Figure 11.3 illustrates, in a simplified fashion, how a hash code is used to provide a digital signature.

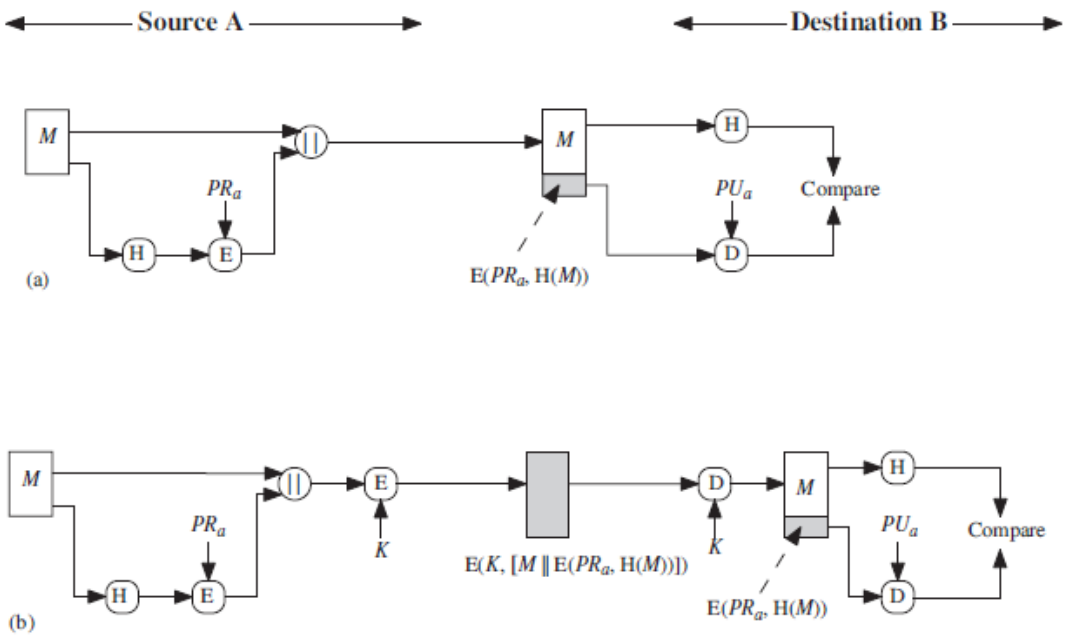


Figure 11.3 Simplified Examples of Digital Signatures

REQUIREMENTS& SECURITY FOR A HASH FUNCTION:

The purpose of a hash function is to produce a "fingerprint" of a file, message or other block of data. To be useful for message authentication, a hash function H must have the following properties:

H can be applied to a block of data of any size

H produces a fixed length output.

H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.

One-way property: - for any given value h, it is computationally infeasible to find x such that H(x)=h. this sometimes referred to in the literature as the one way property.

Weak collision resistance:- for any given block x. it is computationally infeasible to find y≠x with H(y)=H(x). this is referred as weak collision resistance.

Strong collision resistance:- it is computationally infeasible to find any pair (X,Y) such that H(x)=H(y). this is referred as strong collision resistance.

Requirements for a Cryptographic Hash Function H

Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	$H(x)$ is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find y such that $H(y) = h$.
Second preimage resistant (weak collision resistant)	For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.
Pseudorandomness	Output of H meets standard tests for pseudorandomness.

A hash function that satisfies the first five properties in Table 11.1 is referred to as a weak hash function. If the sixth property, collision resistant, is also satisfied, then it is referred to as a strong hash function.

As with encryption algorithms, there are two categories of attacks on hash functions: brute-force attacks and cryptanalysis

Brute-Force Attacks

A brute-force attack does not depend on the specific algorithm but depends only on bit length. In the case of a hash function, a brute-force attack depends only on the bit length of the hash value. A cryptanalysis, in contrast, is an attack based on weaknesses in a particular cryptographic algorithm.

Cryptanalysis

As with encryption algorithms, cryptanalytic attacks on hash functions seek to exploit some property of the algorithm to perform some attack other than an exhaustive search. The way to measure the resistance of a hash algorithm to cryptanalysis is to compare its strength to the effort required for a brute-force attack.

That is, an ideal hash algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort.

SHA(Secure Hash Algorithm):

In recent years, the most widely used hash function has been the Secure Hash Algorithm (SHA).

Introduction:

The Secure Hash Algorithm is a family of cryptographic hash functions developed by the NIST (National Institute of Standards & Technology).

SHA is based on the MD4 algorithm and its design closely models MD5.

SHA-1 is specified in RFC 3174.

Purpose: Authentication, not encryption.

SHA-1 produces a hash value of 160 bits. In 2002, NIST produced a revised version of the standard, FIPS 180-2, that defined three new versions of SHA, with hash value lengths of 256, 384, and 512 bits, known as SHA-256, SHA-384, and SHA-512, respectively.

SHA-1 logic:

The algorithm takes a message with maximum of length of less than 264 bits.

Produce output is 160 bits message digest.

The input is processed 512 bits block.

Processed Steps:

Algorithm processing Steps:

Step1: Append Padding Bits

Step 2: Append Length

Step 3: Initialize MD Buffer

Step 4: Process Message in 512 bit (16-Word) Blocks

Step 5: Output

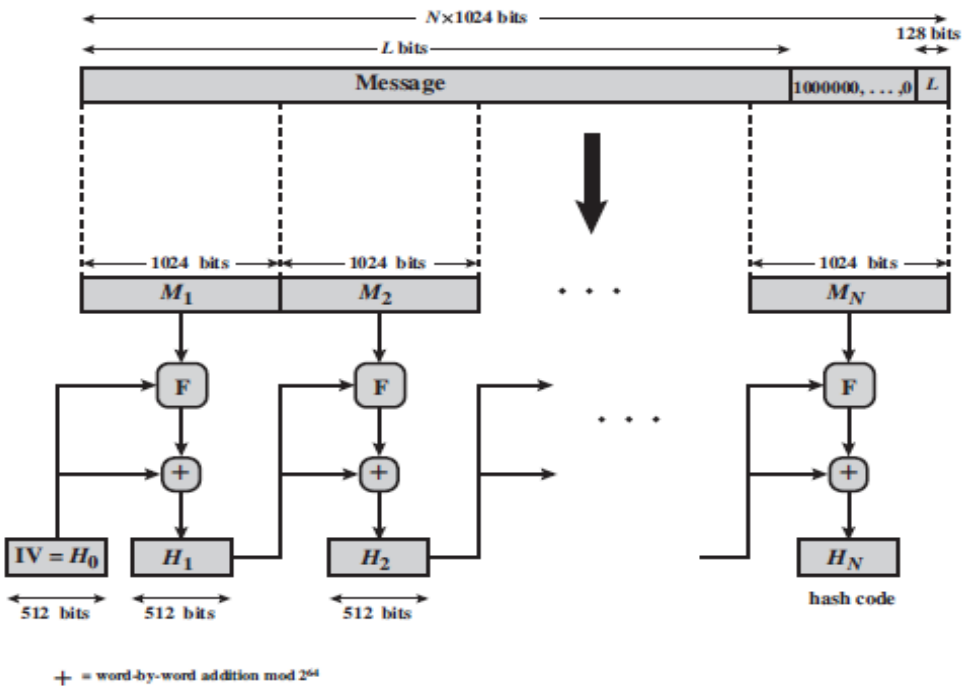


Figure 11.8 Message Digest Generation Using SHA-512

Step-1: **Appending Padding Bits**: The original message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. The padding rules are:

The original message is always padded with one bit "1" first.

Then zero or more bits "0" are padded to bring the length of the message up to 64 bits fewer than a multiple of 512.

Step-2: **append length**: a block of 64 bits is appended to the message. This block is treated as unsigned 64 bit integers (most significant byte first) and contains the length of the original message.

Step-3: **Initialize MD buffer**: 160 bit buffer is used to hold intermediate and final results of the hash function. This buffer can be represented as five 32 bit registers (A, B,C,D,E).

Step 4: Process message in 1024-bit (128-word) blocks. The heart of the algorithm is a module that consists of 80 rounds; this module is labeled F in Figure 11.8. The logic is illustrated in Figure 11.9.

Each round takes as input the 512-bit buffer value, abcdefgh, and updates the contents of the buffer. At input to the first round, the buffer has the value of the intermediate hash value, H_{i-1}. Each round t makes use of a 64-bit value W_t, derived from the current 1024-bit block being processed (M_i). These values are derived using a message schedule described subsequently. Each round also makes use of an additive constant K_t, where $0 \leq t \leq 79$ indicates one of the 80 rounds. These words represent the first 64 bits of the fractional parts of the cube roots of the first 80 prime numbers. The constants provide a "randomized" set of 64-bit patterns, which should eliminate any regularities in the input data. Table 11.4 shows these constants in hexadecimal format (from left to right).

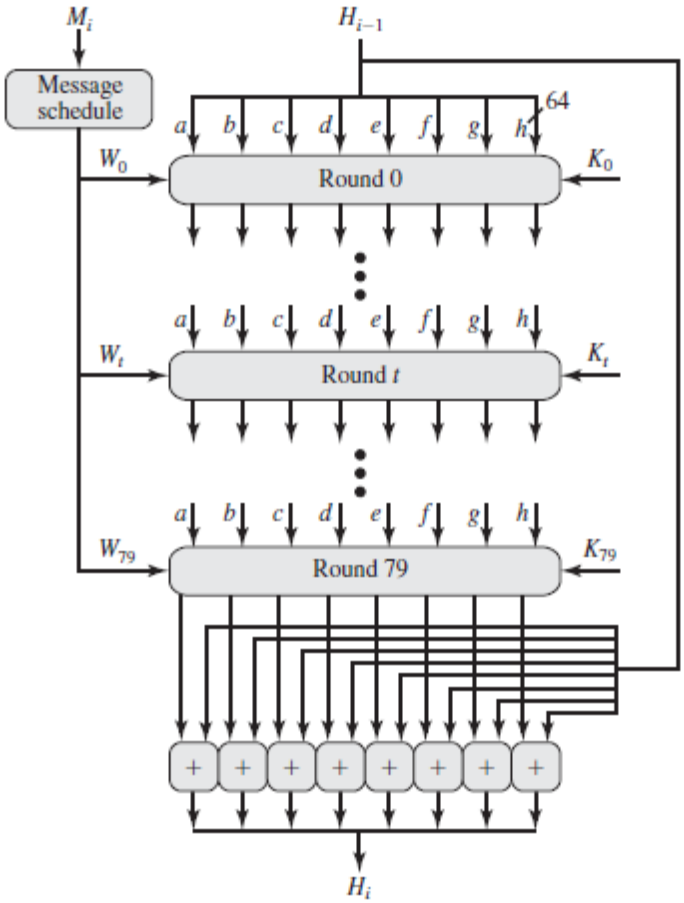


Figure 11.9 SHA-512 Processing of a Single 1024-Bit Block

Step 5 Output. After all N 1024-bit blocks have been processed, the output from the Nth stage is the 512-bit message digest.

We can summarize the behavior of SHA-512 as follows:

$$H_0 = \text{IV}$$
$$H_i = \text{SUM}_{64}(H_{i-1}, \text{abcdefgh}_i)$$
$$MD = H_N$$

where

IV = initial value of the abcdefgh buffer, defined in step 3

abcdefgh_i = the output of the last round of processing of the ith message block

N = the number of blocks in the message (including padding and length fields)

SUM_{64} = addition modulo 2^{64} performed separately on each word of the pair of inputs

MD = final message digest value

MESSAGE AUTHENTICATION

Message authentication is a mechanism or service used to verify theintegrity of a message. Message authentication assures that data receivedare exactly as sent by (i.e., contain no modification, insertion, deletion, orreplay) and that the purported identity of the sender is valid.

MESSAGE AUTHENTICATION REQUIREMENTS

In the context of communications across a network, the following attacks can be identified

1. Disclosure: Release of message contents to any person or process not possessing theappropriate cryptographic key.

2. Traffic analysis: Discovery of the pattern of traffic between parties. In a connectionorientedapplication, the frequency and duration of connections could be determined.

3. Masquerade: Insertion of messages into the network from a fraudulent source.

4. Content modification: Changes to the contents of a message, including insertion,deletion, transposition, and modification.

5. Sequence modification: Any modification to a sequence of messages betweenparties, including insertion, deletion, and reordering.

6. Timing modification: Delay or replay of messages. In a connection-orientedapplication, an entire session or sequence of messages could be a replay of someprevious valid session, or individual messages in the sequence could be delayed orreplayed.

7. Source repudiation: Denial of transmission of message by source.

8. Destination repudiation: Denial of receipt of message by destination.

MESSAGE AUTHENTICATION FUNCTIONS

Any message authentication or digital signature mechanism has two levels of functionality.At the lower level, there must be some sort of function that produces anauthenticator: a value to be used to authenticate a message. This lower-level functionis then used as a primitive in a higher-level authentication protocol that enablesa receiver to verify the authenticity of a message.there are 3 types of functions that may be used to producean authenticator.

• **Hash function:** A function that maps a message of any length into a fixedlengthhash value, which serves as the authenticator
• **Message encryption:** The cipher text of the entire message serves as its authenticator
• **Message authentication code (MAC):** A function of the message and a secretkey that produces a fixed-length value that serves as the authenticator

Message Encryption
Message encryption by itself can provide a measure of authentication. The analysisdiffers for symmetric and public-key encryption schemes.

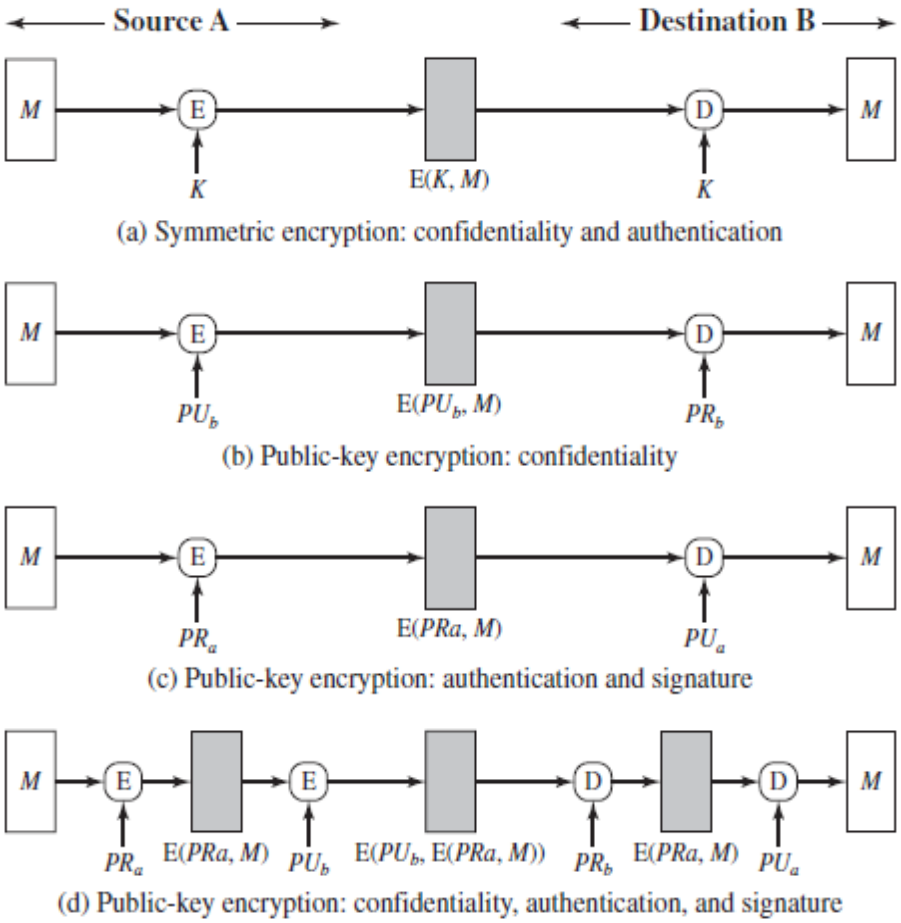


Figure 12.1 Basic Uses of Message Encryption

MESSAGE AUTHENTICATION CODE (MAC)

This authentication technique involves the use of a secret key to generatea small fixed-size block of data, known as a **cryptographic checksum** or MAC, that isappended to the message. This technique assumes that two communicating parties,say A and B, share a common secret key

When A has a message to send to B, itcalculates the MAC as a function of the message and the key

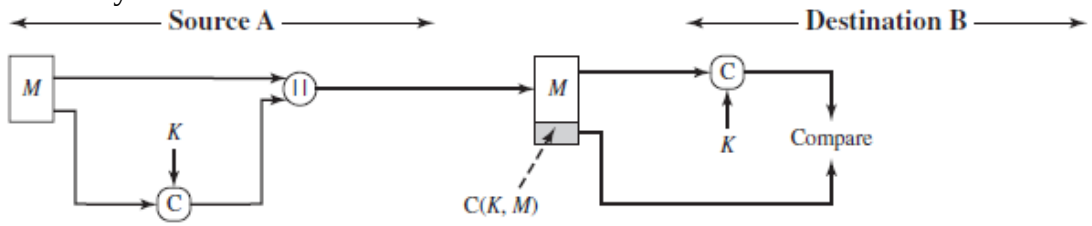
$$MAC = MAC(K, M)$$

where

M = input message
C = MAC function
K = shared secret key
MAC = message authentication code

The message plus MAC are transmitted to the intended recipient.The recipient performsthe same calculation on the received message, using the same secret key, togenerate a new MAC. The received MAC is compared to the calculated MAC(Figure 12.4a). If we assume that only the receiver and the sender know the identityof the secret key, and if the received MAC matches the calculated MAC, then

1. The receiver is assured that the message has not been altered. If an attackeralters the message but does not alter the MAC, then the receiver's calculationof the MAC will differ from the received MAC.
2. The receiver is assured that the message is from the alleged sender. Because noone else knows the secret key.



(a) Message authentication

SECURITY OF MACS:

Just as with symmetric and public-key encryption, we can group attacks on hashfunctions and MACs into two categories: **brute-force attacks and cryptanalysis.**

brute-force attacks

A brute-force attack on a MAC is a more difficult undertaking than a brute-forceattack on a hash function because it requires known message-tag pairs. The strength of ahash function against brute-force attacks depends solely on the length of the hash codeproduced by the algorithm, with cost (2m/2). A brute-force attack on a MAC has costrelated to min(2k, 2n), similar to symmetric encryption algorithms. It would appearreasonable to require that the key length and MAC length satisfy a relationship such asmin(k, n) >= N, where N is perhaps in the range of 128 bits.

cryptanalysis.

As with encryption algorithms, cryptanalytic attacks on hash functions and MACalgorithms seek to exploit some property of the algorithm to perform some attack otherthan an exhaustive search. The way to measure the resistance of a hash or MACalgorithm to cryptanalysis is to compare its strength to the effort required for a bruteforce attack. That is, an ideal hash or MAC algorithm will require a cryptanalytic effortgreater than or equal to the brute-force effort.

HMAC:

In recent years, there has been increased interest in developing a MAC derivedfrom a cryptographic hash function, because they generally execute faster than symmetricblock ciphers, and because code for cryptographic hash functions is widely available.

A hash function such as SHA was not designed for use as a MAC and cannot beused directly for that purpose because it does not rely on a secret key. There have been anumber of proposals for the incorporation of a secret key into an existing hash algorithm,originally by just pre-pending a key to the message. Problems were found with theseearlier, simpler proposals, but they resulted in the development of HMAC.

HMAC Design Objectives:

• To use, without modifications, available hash functions. In particular, to use hashfunctions that perform well in software and for which code is freely and widelyavailable.

• To allow for easy replaceability of the embedded hash function in case faster or moresecure hash functions are found or required.

• To preserve the original performance of the hash function without incurring asignificant degradation.

• To use and handle keys in a simple way.

• To have a well understood cryptographic analysis of the strength of the authenticationmechanism based on reasonable assumptions about the embedded hash function.

HMAC Algorithm:

HMAC Algorithm

Figure 12.5 illustrates the overall operation of HMAC. Define the following terms.

H = embedded hash function (e.g., MD5, SHA-1, RIPEMD-160)
IV = initial value input to hash function
M = message input to HMAC (including the padding specified in the embedded hash function)
Y_i = i th block of $M, 0 \leq i \leq (L-1)$
L = number of blocks in M
b = number of bits in a block
n = length of hash code produced by embedded hash function
K = secret key; recommended length is $\geq n$; if key length is greater than b, the key is input to the hash function to produce an n-bit key

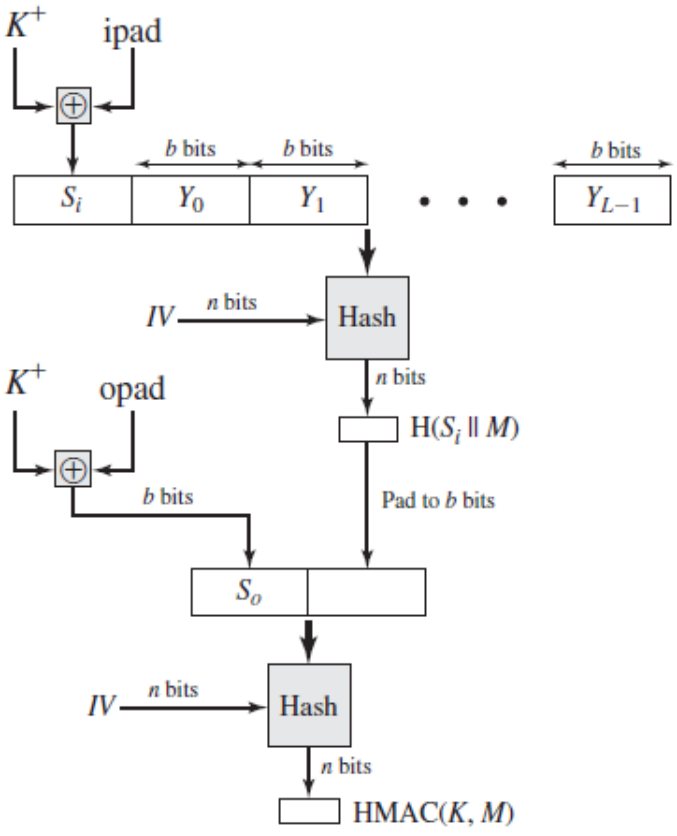
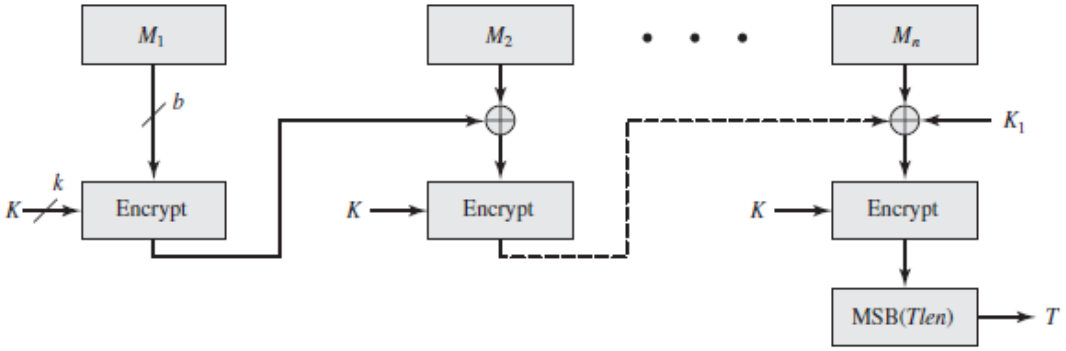


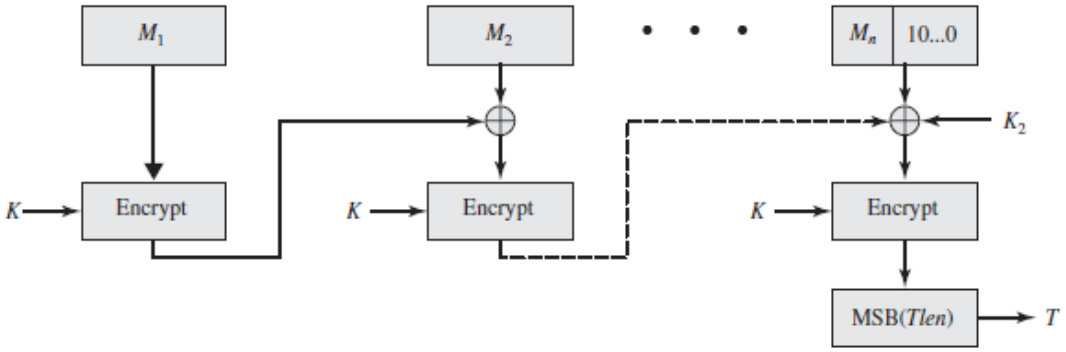
Figure 12.5 HMAC Structure

Cipher-Based Message Authentication Code (CMAC)

First, let us define the operation of CMAC when the message is an integer multiple n of the cipher block length b. For AES, $b = 128$, and for triple DES, $b = 64$. The message is divided into n blocks $(M_1, M_2, \ldots, M_n)$. The algorithm makes use of a k-bit encryption key K and an n-bit constant, K_1. For AES, the key size k is 128, 192, or 256 bits; for triple DES, the key size is 112 or 168 bits. CMAC is calculated as follows (Figure 12.8).



(a) Message length is integer multiple of block size



(b) Message length is not integer multiple of block size

Figure 12.8 Cipher-Based Message Authentication Code (CMAC)

$$
\begin{aligned}
C_1 &= \mathrm{E}(K, M_1) \\
C_2 &= \mathrm{E}(K, [M_2 \oplus C_1]) \\
C_3 &= \mathrm{E}(K, [M_3 \oplus C_2]) \\
&\bullet \\
&\bullet \\
&\bullet \\
C_n &= \mathrm{E}(K, [M_n \oplus C_{n-1} \oplus K_1]) \\
T &= \mathrm{MSB}_{Tlen}(C_n)
\end{aligned}
$$

where

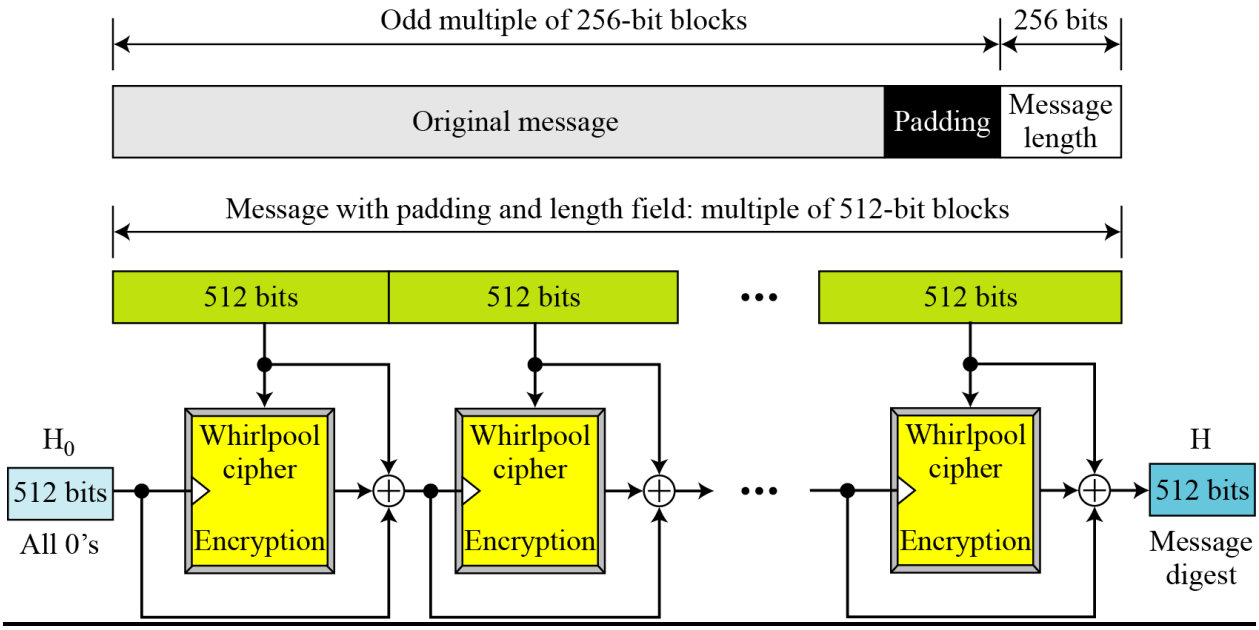
T = message authetication code, also referred to as the tag
Tlen = bit length of T
$\mathrm{MSB}_s(X)$ = the s leftmost bits of the bit string X

If the message is not an integer multiple of the cipher block length, then the final block is padded to the right (least significant bits) with a 1 and as many 0s as necessary so that the final block is also of length b. The CMAC operation then proceeds as before, except that a different n-bit key K_2 is used instead of K_1.

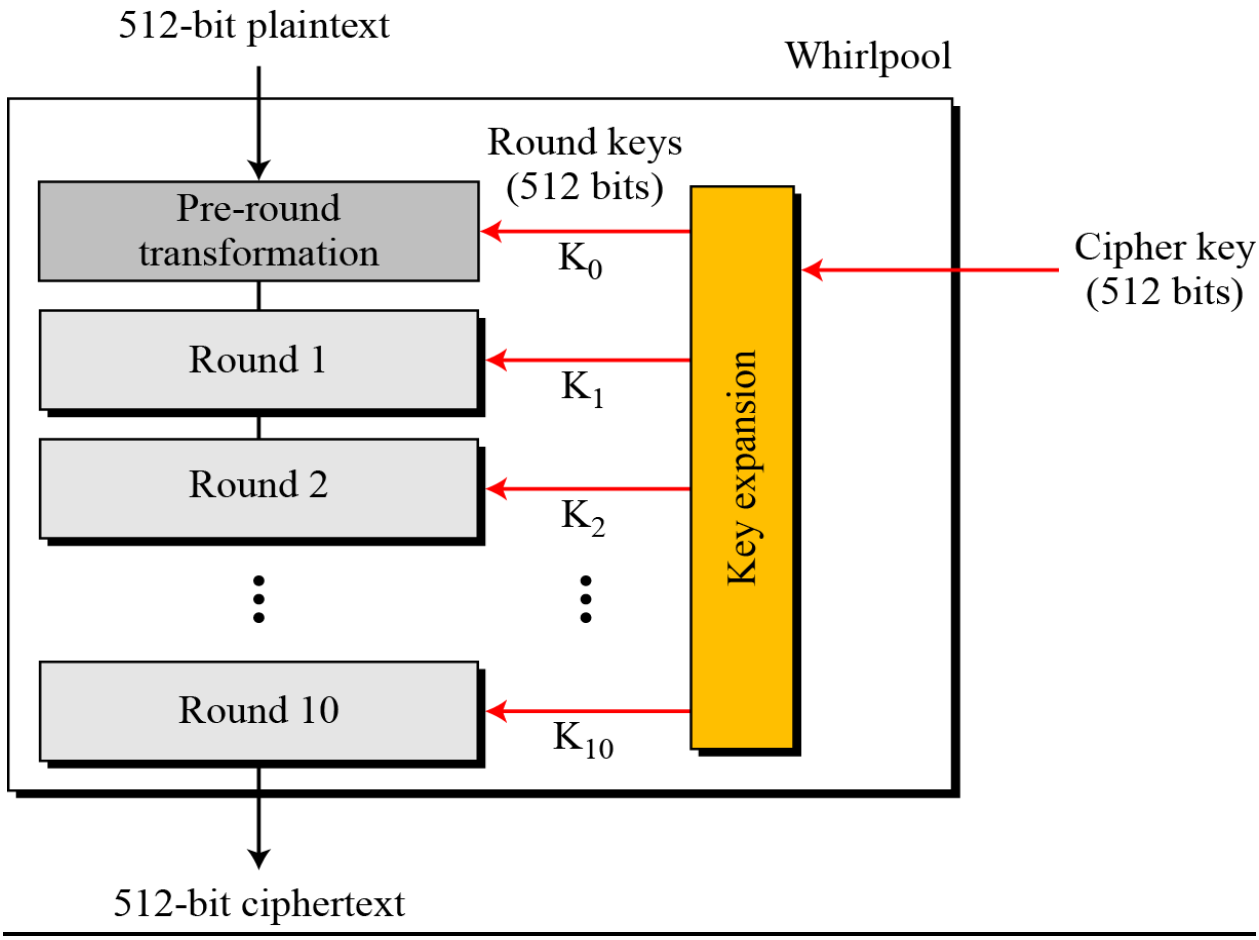
WHIRLPOOL:

Whirlpool is an iterated cryptographic hash function, based on the Miyaguchi-Preneel scheme, that uses a symmetric-key block cipher in place of the compression function. The block cipher is a modified AES cipher that has been tailored for this purpose.

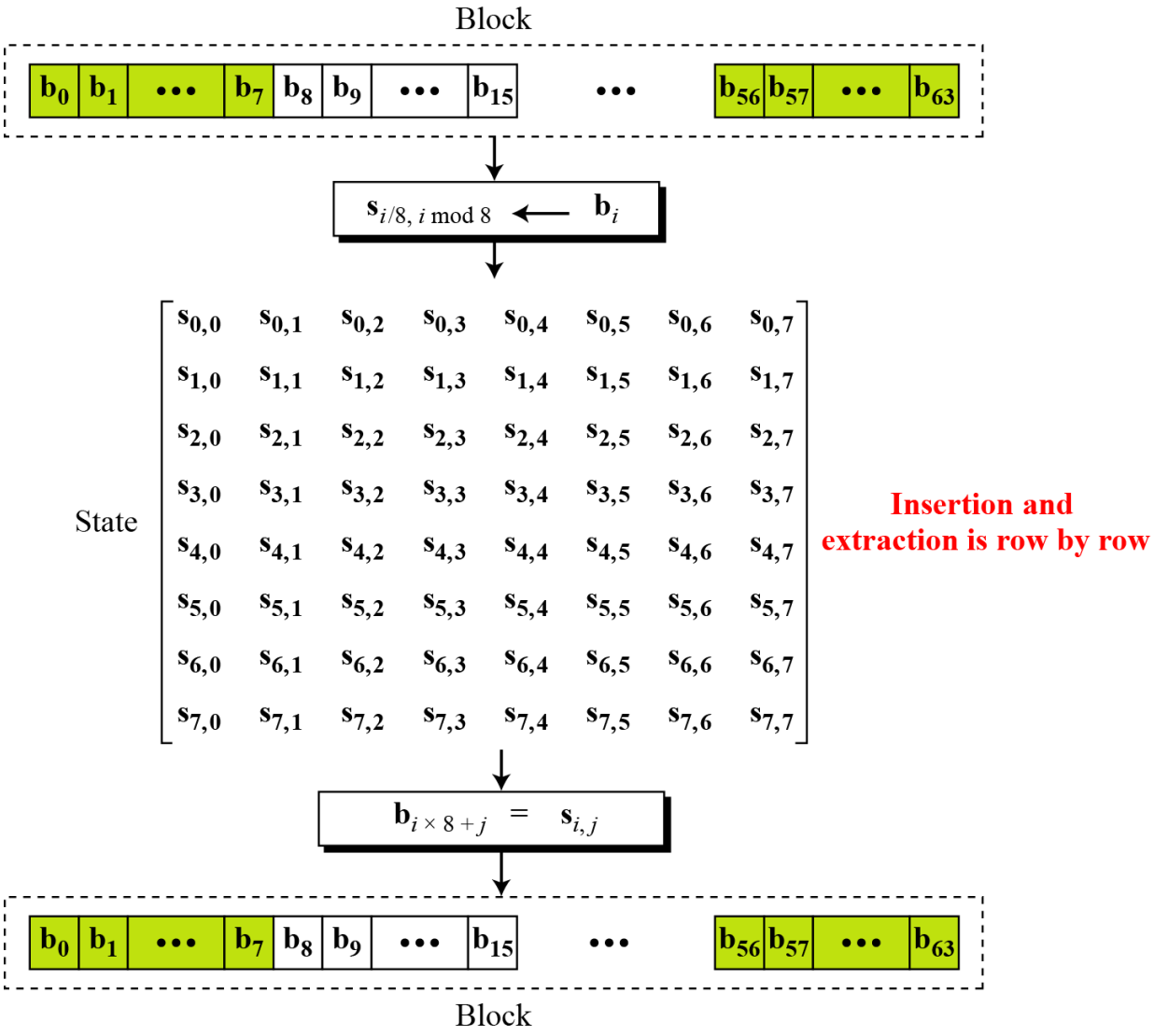
DIAG : Whirlpool hash function



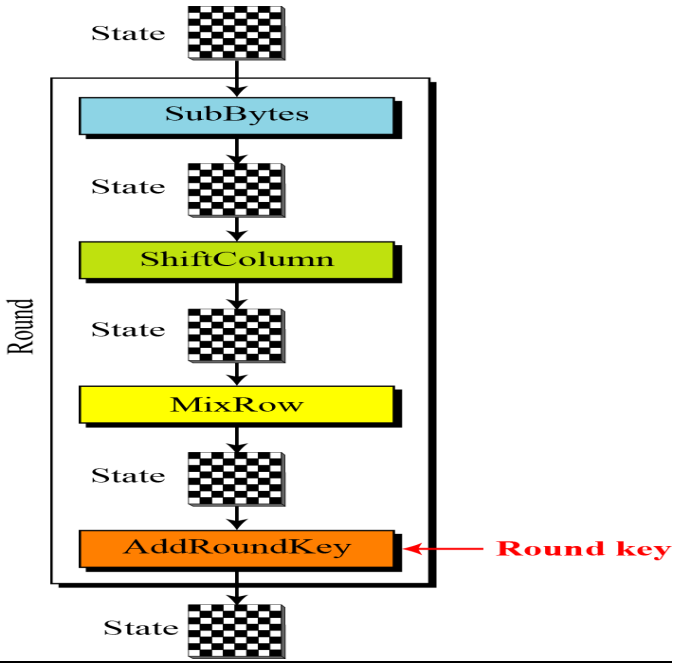
General idea of the Whirlpool cipher



Block and state in the Whirlpool cipher



Structure of Each Round Each round uses four transformations



SubBytes Like in AES, SubBytes provide a nonlinear transformation

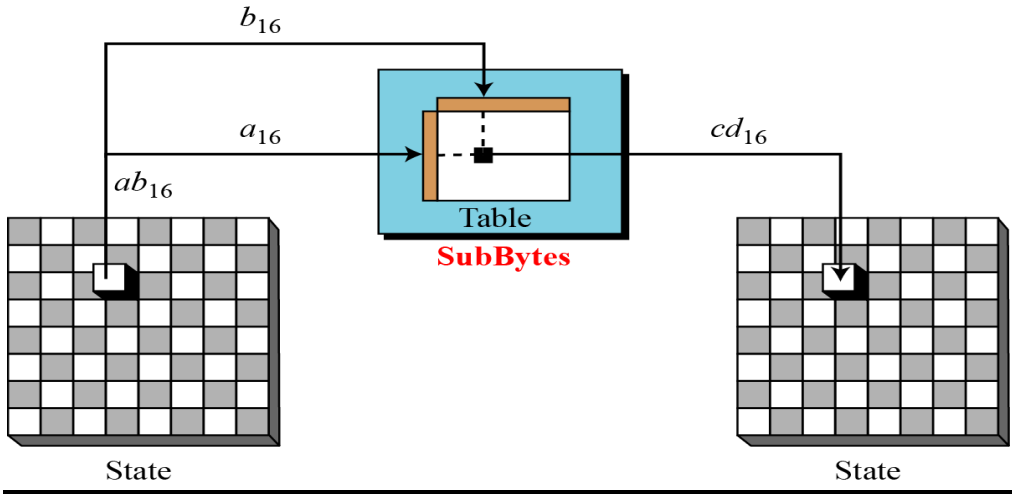
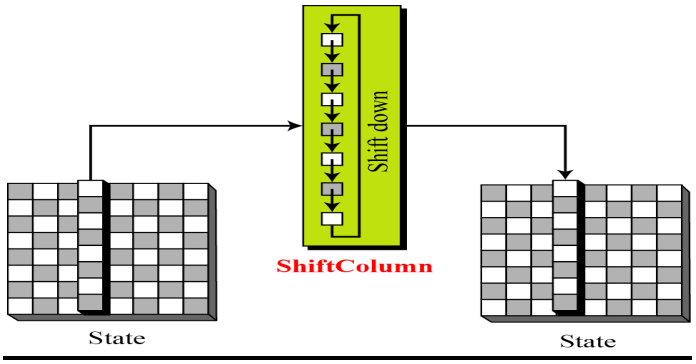


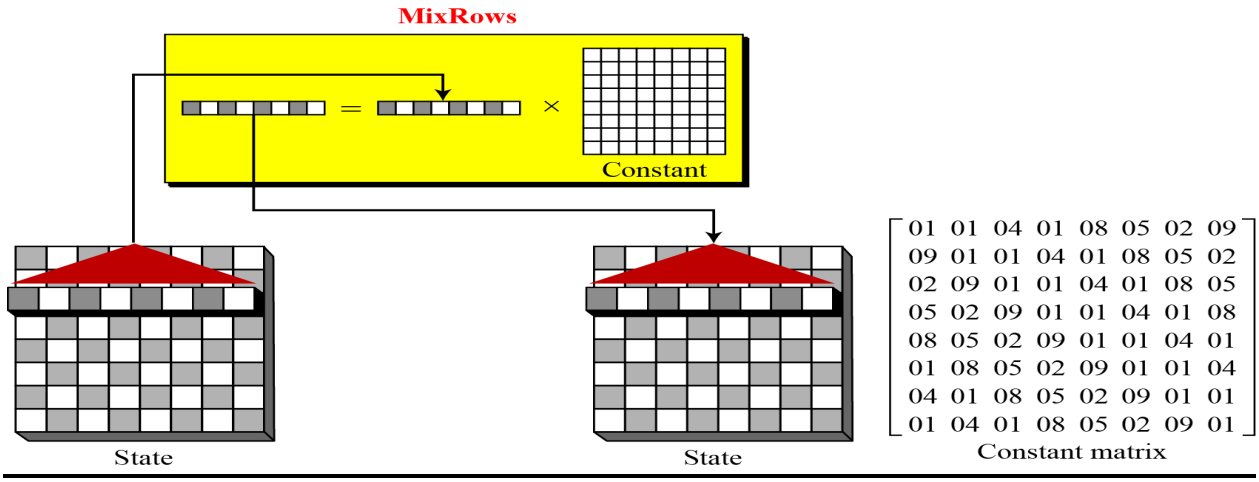
Table 12.4 *SubBytes transformation table (S-Box)*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	18	23	C6	E8	87	B8	01	4F	36	A6	D2	F5	79	6F	91	52
1	16	BC	9B	8E	A3	0C	7B	35	1D	E0	D7	C2	2E	4B	FE	57
2	15	77	37	E5	9F	F0	4A	CA	58	C9	29	0A	B1	A0	6B	85
3	BD	5D	10	F4	CB	3E	05	67	E4	27	41	8B	A7	7D	95	C8
4	FB	EF	7C	66	DD	17	47	9E	CA	2D	BF	07	AD	5A	83	33
5	63	02	AA	71	C8	19	49	C9	F2	E3	5B	88	9A	26	32	B0
6	E9	0F	D5	80	BE	CD	34	48	FF	7A	90	5F	20	68	1A	AE
7	B4	54	93	22	64	F1	73	12	40	08	C3	EC	DB	A1	8D	3D
8	97	00	CF	2B	76	82	D6	1B	B5	AF	6A	50	45	F3	30	EF
9	3F	55	A2	EA	65	BA	2F	C0	DE	1C	FD	4D	92	75	06	8A
A	B2	E6	0E	1F	62	D4	A8	96	F9	C5	25	59	84	72	39	4C
B	5E	78	38	8C	C1	A5	E2	61	B3	21	9C	1E	43	C7	FC	04
C	51	99	6D	0D	FA	DF	7E	24	3B	AB	CE	11	8F	4E	B7	EB
D	3C	81	94	F7	9B	13	2C	D3	E7	6E	C4	03	56	44	7E	A9
E	2A	BB	C1	53	DC	0B	9D	6C	31	74	F6	46	AC	89	14	E1
F	16	3A	69	09	70	B6	C0	ED	CC	42	98	A4	28	5C	F8	86

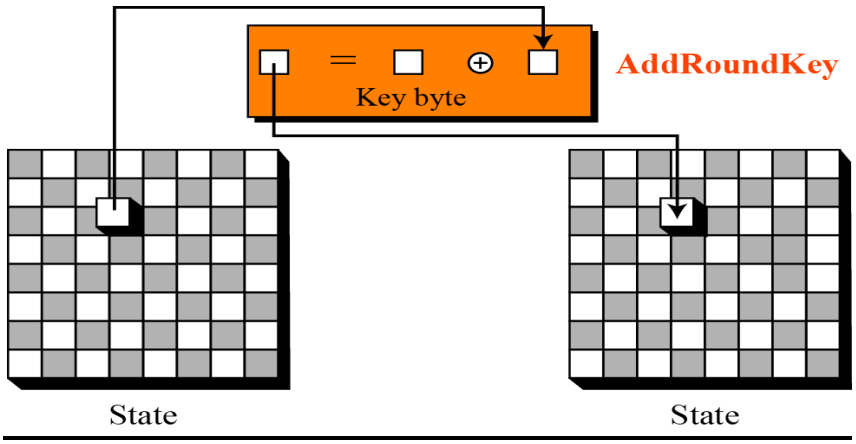
ShiftColumns



MixRows transformation in the Whirlpool cipher



AddRoundKey transformation in the Whirlpool cipher



Key expansion in the Whirlpool cipher

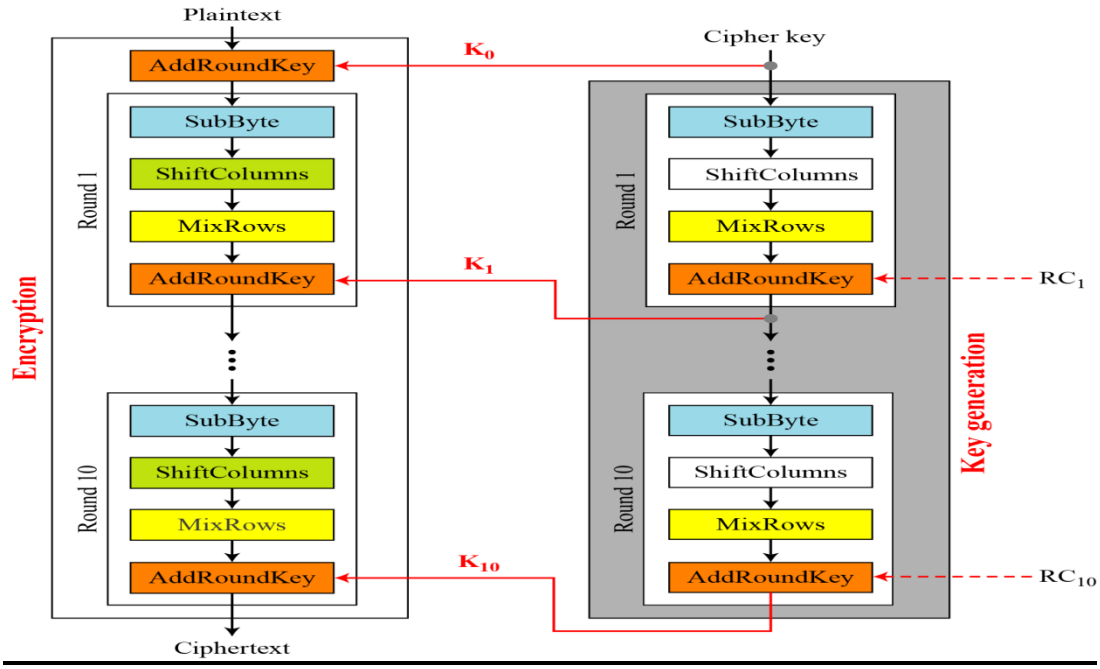


Table 12.5 *Main characteristics of the Whirlpool cipher*

Block size: 512 bits
Cipher key size: 512 bits
Number of rounds: 10
Key expansion: using the cipher itself with round constants as round keys
Substitution: SubBytes transformation
Permutation: ShiftColumns transformation
Mixing: MixRows transformation
Round Constant: cubic roots of the first eighty prime numbers

Although Whirlpool has not been extensively studied or tested, it is based on a robust scheme (Miyaguchi-Preneel), and for a compression function uses a cipher that is based on AES, a cryptosystem that has been proved very resistant to attacks. In addition, the size of the message digest is the same as for SHA-512. Therefore it is expected to be a very strong cryptographic hash function

DIGITAL SIGNATURES

A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature. Typically the signature is formed by taking the hash of the message and encrypting the message with the creator's private key. The signature guarantees the source and integrity of the message.

The digital signature standard (DSS) is an NIST standard that uses the secure hash algorithm (SHA).

Digital signature process

Properties

Message authentication protects two parties who exchange messages from any third-party. However, it does not protect the two parties against each other. Several forms of dispute between the two are possible.

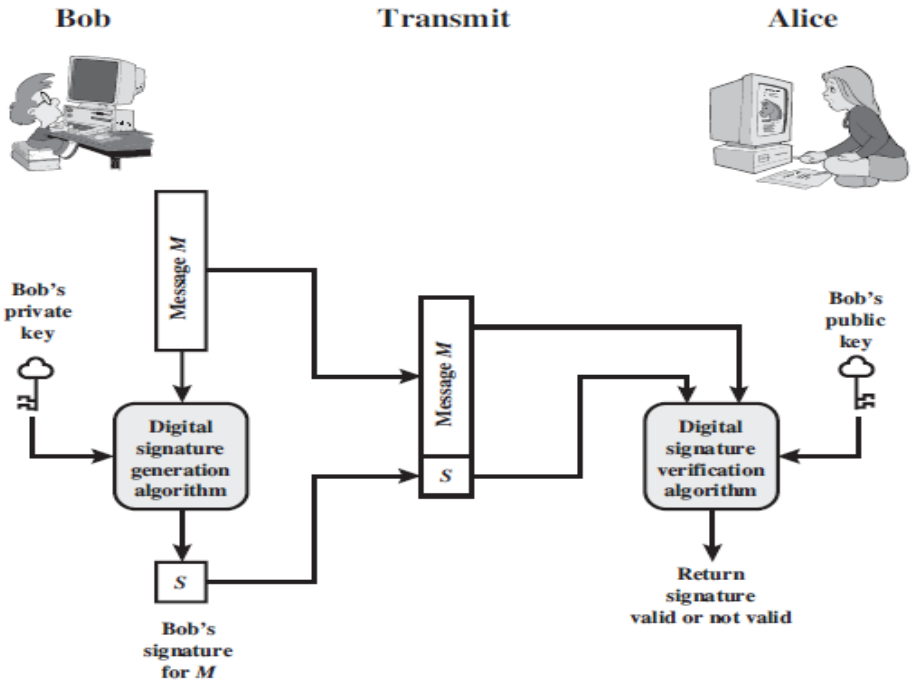


Figure 13.1 Generic Model of Digital Signature Process

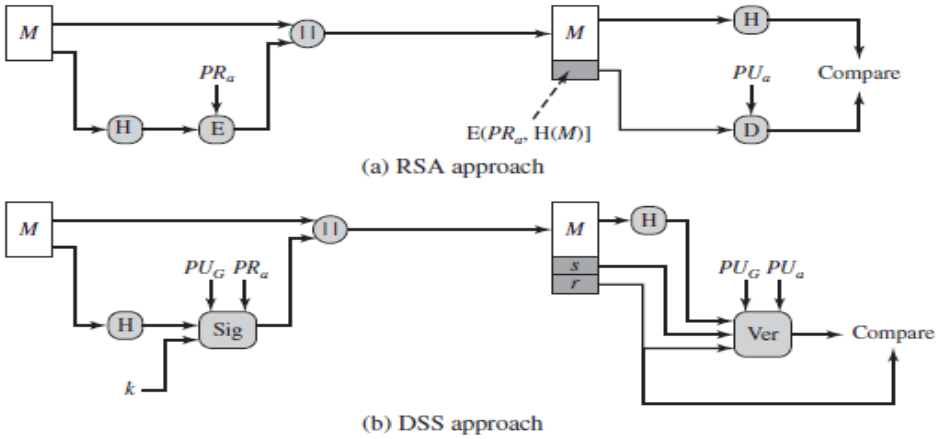


Figure 13.3 Two Approaches to Digital Signatures

In the RSA approach, the message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender's private key to form the signature. Both the message and the signature are then transmitted. The recipient takes the message and produces a hash code.

The recipient also decrypts the signature using the sender's public key. If the calculated hash code matches the decrypted signature, the signature is accepted as valid. Because only the sender knows the private key, only the sender could have produced a valid signature.

KEY MANAGEMENT AND DISTRIBUTION

Symmetric Key Distribution Using Symmetric EncryptionFor symmetric encryption to work, the two parties to an exchange must share thesame key, and that key must be protected from access by others. Therefore, the term thatrefers to the means of delivering a key to two parties who wish to exchange data, withoutallowing others to see the key. For two parties A and B, key distribution can be achieved in anumber of ways, as follows:

1. A can select a key and physically deliver it to B.

2. A third party can select the key and physically deliver it to A and B.

3. If A and B have previously and recently used a key, one party can transmit the newkey to the other, encrypted using the old key.

4. If A and B each has an encrypted connection to a third party C, C can deliver a key onthe encrypted links to A and B.

Physical delivery (1 & 2) is simplest - but only applicable when there is personalcontact between recipient and key issuer. This is fine for link encryption where devices &keys occur in pairs, but does not scale as number of parties who wish to communicate grows.

3 is mostly based on 1 or 2 occurring first.A third party, whom all parties trust, can be used as a trusted intermediary tomediate the establishment of secure communications between them (4). Must trustintermediary not to abuse the knowledge of all session keys. As number of parties grow,some variant of 4 is only practical solution to the huge growth in number of keys potentiallyneeded.

Key distribution centre:

 The use of a key distribution centeris based on the use of a hierarchy of keys. At aminimum, two levels of keys are used.

 Communication between end systems is encrypted using a temporary key, often referredto as a Session key.

 Typically, the session key is used for the duration of a logical connection and thendiscarded

 Master key is shared by the key distribution center and an end system or user and used to

encrypt the session key.

A Key Distribution Scenario

The key distribution concept can be deployed in a number of ways. A typical scenario is illustrated in Figure 14.3, which is based on a figure in [POPE79]. The scenario assumes that each user shares a unique master key with the key distribution center (KDC).

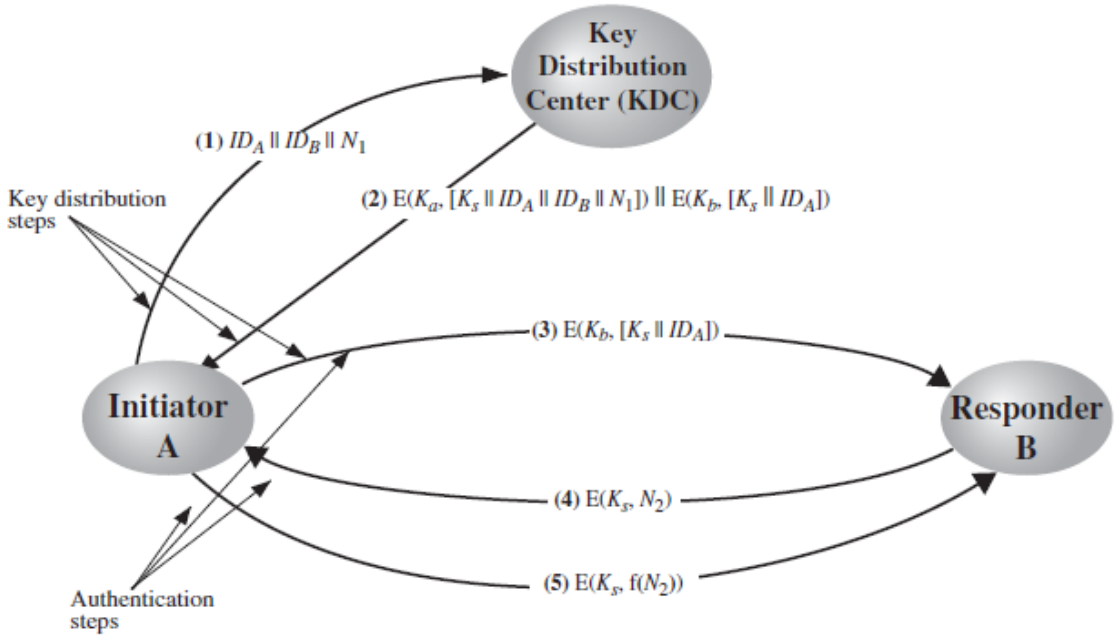


Figure 14.3 Key Distribution Scenario

Symmetric Key Distribution Usingasymmetric Encryption

Because of the inefficiency of public key cryptosystems, they are almost never usedfor the direct encryption of sizable block of data, but are limited to relatively smallblocks. One of the most important uses of a public-key cryptosystem is to encryptsecret keys for distribution.Wesee many specific examples of this in Part Five. Here,we discuss general principles and typical approaches.

Simple Secret Key Distribution

An extremely simple scheme was put forward by Merkle [MERK79], as illustrated in Figure 14.7. If A wishes to communicate with B, the following procedure is employed:

1. A generates a public/private key pair $\{PU_a, PR_a\}$ and transmits a message to B consisting of PU_a and an identifier of A, ID_A.
2. B generates a secret key, K_s, and transmits it to A, which is encrypted with A's public key.
3. A computes $D(PR_a, E(PU_a, K_s))$ to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of K_s.
4. A discards PU_a and PR_a and B discards PU_a.

A and B can now securely communicate using conventional encryption and the session key K_s. At the completion of the exchange, both A and B discard K_s.

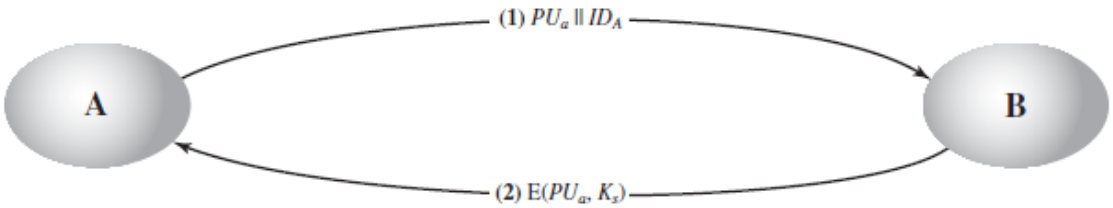


Figure 14.7 Simple Use of Public-Key Encryption to Establish a Session Key

REMOTE USER-AUTHENTICATION PRINCIPLES

In most computer security contexts, user authentication is the fundamental buildingblock and the primary line of defense. User authentication is the basis for mosttypes of access control and for user accountability

AnAuthentication process consists of two steps

- **Identification step:** Presenting an identifier to the security system. (Identifiers should be assigned carefully, because authenticated identities are the basis for other security services, such as access control service.)
- **Verification step:** Presenting or generating authentication information that corroborates the binding between the entity and the identifier.

Identification is the means by which a user provides a claimed identity to the system; user authentication is the means of establishing the validity of theclaim .Note that user authentication is distinct from message authentication.

There are four general means of authenticating a user's identity, which can beused alone or in combination:

• **Something the individual knows:** Examples include a password, a personal identification number (PIN), or answers to a prearranged set ofquestions.

• Something the individual possesses: Examples include cryptographic keys,electronic keycards, smart cards, and physical keys. This type of authenticatoris referred to as a token.

• **Something the individual is (static biometrics):** Examples include recognitionby fingerprint, retina, and face.

• **Something the individual does (dynamic biometrics):** Examples include recognition by voice pattern, handwriting characteristics, and typing rhythm.

KERBEROS:

- ✓ **Introduction:**

Kerberos is a computer network authentication protocol which works on the basis of "tickets" to allow nodes communication over a non secure network.

(Or) It is a secure method (service) to authenticate a request for a service in a computer network.

It was developed in "Athena Project" at MIT.

It provide authentication by using Secret-Key cryptography.

- ✓ **Use of Kerberos:**

Kerberos is used for decreasing the burden for server, means; Kerberos will takes responsibility of authentication.

It is designed for providing for strong authentication for client/server applications by using secret-key.

- ✓ **Versions:**

Kerberos Version4

Kerberos Version5

Characteristics of KERBEROS:

- ✓ It is secure: it never sends a password unless it is encrypted.
- ✓ Only a single login is required per session. Credentials defined at login are then passed between resources without the need for additional logins.
- ✓ The concept depends on a trusted third party – a Key Distribution Center (KDC). The KDC is aware of all systems in the network and is trusted by all of them.
- ✓ It performs mutual authentication, where a client proves its identity to a server and a server proves its identity to the client.

Requirements Kerberos:

- ✓ **Secure**: Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
- ✓ **Reliable:** Kerberos should be highly reliable and should employ distributed server architecture with one system able to back up another.
- ✓ **Transparent:** Ideally, the user should not be aware that authentication is taking place beyond the requirement to enter a password.
- ✓ **Scalable:** The system should be capable of supporting large numbers of clients and servers.

Kerberos Version 4

Version 4 of Kerberos makes use of DES

A SIMPLE AUTHENTICATION DIALOGUE

In an unprotected network environment, any client can apply to any server for service. The obvious security risk is that ofimpersonation. An opponent can pretend to be another client and obtainunauthorized privileges on server machines. To counter this threat, servers must beable to confirm the identities of clients who request service. Each server can berequired to undertake this

task for each client/server interaction, but in an openenvironment, this places a substantial burden on each server.

An alternative is to use an Authentication server (AS) that knows the passwords of all users and stores these in a centralized database. In addition, the AS shares a unique secret key with each server. These keys have been distributed physically or in some other secure manner. Consider the following hypothetical dialogue:

(1) C → AS: $ID_C \| P_C \| ID_V$

(2) AS → C: $Ticket$

(3) C → V: $ID_C \| Ticket$

$Ticket = E(K_v, [ID_C \| AD_C \| ID_V])$

where

C = client
AS = authentication server
V = server
ID_C = identifier of user on C
ID_V = identifier of V
P_C = password of user on C
AD_C = network address of C
K_v = secret encryption key shared by AS and V

1. The user logs on to a workstation and requests access to server V. The client module C in the user's workstation requests the user's password and then sends a message to the AS that includes the user's ID, the server's ID, and the user's password.
2. The AS checks its database to see if the user has supplied the proper password for this user ID and whether this user is permitted access to server V. If both tests are passed, the AS accepts the user as authentic and must now convince the server that this user is authentic. To do so, the AS creates a **ticket** that contains the user's ID and network address and the server's ID. This ticket is encrypted using the secret key shared by the AS and this server.This ticket is then sent back to C.
3. With this ticket, C can now apply to V for service. C sends a message to V containing C's ID and the ticket. V decrypts the ticket and verifies that the user ID in the ticket is the same as the unencrypted user ID in the message.

Problems:

The first problem is that the earlier scenario involved a plaintext transmission of the password [message (1)]. An eavesdropper could capture the password and use any services accessible to the victim

An opponent could capture the ticket transmitted in message (2), then use the name ID_C *a*nd transmit a message of form (3) another workstation. The server would receive a valid ticket that matches the user ID and grant access to the user on that other workstation. To prevent this attack, the AS includes in the ticket the network address from which the original request came.

The second problem is we would like to minimize thenumber of times that a user has to enter a password. Suppose each ticket can beused only once. If user C logs on to a workstation in the morning and wishes tocheck his or her mail at a mail server, C must supply a password to get a ticket forthe mail server. If C wishes to check the mail several times during the day, eachattempt requires reentering the password. We can improve matters by saying thattickets are reusable. For a single logon session, the workstation can store the mailserver ticket after it is received and use it on behalf of the user .

A MORE SECURE AUTHENTICATION DIALOGUE

The main problem in ***A SIMPLE AUTHENTICATION DIALOGUE, the user*** must enter password for every individual service.

Kerberos overcome this by using a new server, known as Ticket granting server (TGS).

Now in Kerberos we have two servers; AS and TGS.

Once per user logon session:

(1) $C \rightarrow AS: \quad ID_C \| ID_{tgs}$

(2) $AS \rightarrow C: \quad E(K_c, Ticket_{tgs})$

Once per type of service:

(3) $C \rightarrow TGS: \quad ID_C \| ID_V \| Ticket_{tgs}$

(4) $TGS \rightarrow C: \quad Ticket_v$

Once per service session:

(5) $C \rightarrow V: \quad ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$
$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$

The new service, TGS, issues tickets to users who have been authenticated to AS. Thus, the user first requests a ticket-granting ticket ($Ticket_{tgs}$) from the AS.

The client module in the user workstation saves this ticket. Each time the user requires access to a new service, the client applies to the TGS, using the ticket to authenticate itself. The TGS then grants a ticket for the particular service. The client saves each service-granting ticket and uses it to authenticate its user to a server each time a particular service is requested. Let us look at the details of this scheme:

1. The client requests a ticket-granting ticket on behalf of the user by sending its user's ID to the AS, together with the TGS ID, indicating a request to use the TGS service.
2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password ( ), which is already stored at the AS. When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message. If the correct password is supplied, the ticket is successfully recovered.
3. The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.
4. The TGS decrypts the incoming ticket using a key shared only by the AS and the TGS $(K_{tgs})$and verifies the success of the decryption by the presence of its ID. It checks to make sure that the lifetime has not expired. Then it compares the user ID and network address with the incoming information to authenticate the user. If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested service.
5. The client requests access to a service on behalf of the user. For this purpose, the client transmits a message to the server containing the user's ID and the service-granting ticket. The server authenticates by using the contents of the ticket.

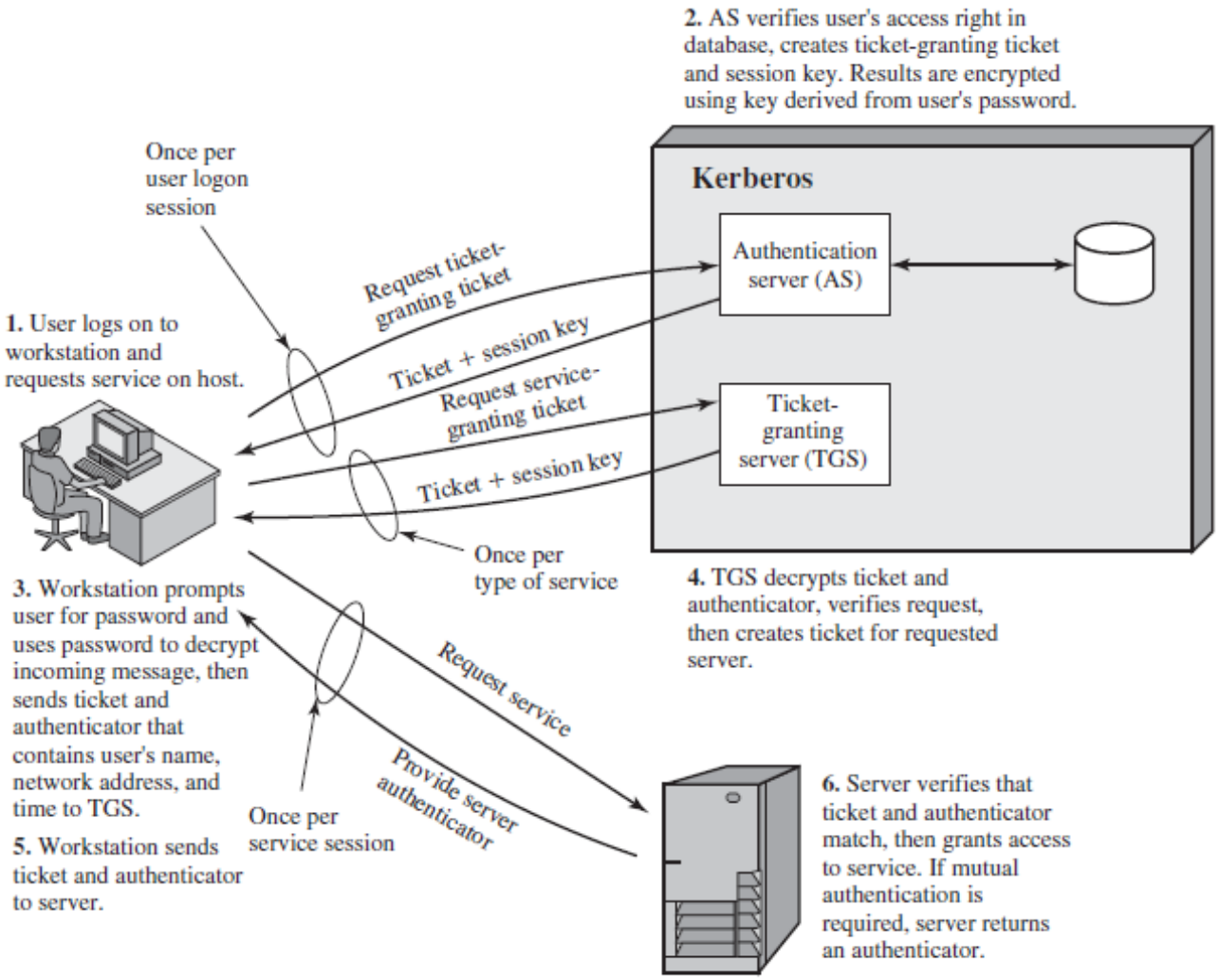


Figure 15.1 Overview of Kerberos

Table 15.1 Summary of Kerberos Version 4 Message Exchanges

(1)	$\mathbf{C} \rightarrow \mathbf{AS} \quad ID_c \| ID_{tgs} \| TS_1$
(2)	$\mathbf{AS} \rightarrow \mathbf{C} \quad E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$
	$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3)	$\mathbf{C} \rightarrow \mathbf{TGS} \quad ID_v \| Ticket_{tgs} \| Authenticator_c$
(4)	$\mathbf{TGS} \rightarrow \mathbf{C} \quad E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$
	$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$
	$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$
	$Authenticator_c = E(K_{c,tgs}, [ID_C \| AD_C \| TS_3])$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5)	$\mathbf{C} \rightarrow \mathbf{V} \quad Ticket_v \| Authenticator_c$
(6)	$\mathbf{V} \rightarrow \mathbf{C} \quad E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)
	$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$
	$Authenticator_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$

(c) Client/Server Authentication Exchange to obtain service

UNIT-V

WEB SECURITY

Usage of internet for transferring or retrieving the data has got many benefits like speed, reliability, security etc. Much of the Internet's success and popularity lies in the fact that it is an open global network. At the same time, the fact that it is open and global makes it not very secure. The unique nature of the Internet makes exchanging information and transacting business over it inherently dangerous.

For the exchange of information and for commerce to be secure on any network, especially the Internet, a system or process must be put in place that satisfies requirements for confidentiality, access control, authentication, integrity, and non-repudiation. These requirements are achieved on the Web through the use of encryption and by employing digital signature technology. There are many examples on the Web of the practical application of encryption. One of the most important is the SSL protocol.

A summary of types of security threats faced in using the Web is given below:

Web Security Threats:

Table 16.1 provides a summary of the types of security threats faced when using the Web. One way to group these threats is in terms of passive and active attacks. Passive attacks include eavesdropping on network traffic between browser and server and gaining access to information on a Web site that is supposed to be restricted. Active attacks include impersonating another user, altering messages in transit between client and server, and altering information on a Web site. Another way to classify Web security threats is in terms of the location of the threat: Web server, Web browser, and network traffic between browser and server.

Table 16.1 A Comparison of Threats on the Web

	Threats	Consequences	Countermeasures
Integrity	• Modification of user data • Trojan horse browser • Modification of memory • Modification of message traffic in transit	• Loss of information • Compromise of machine • Vulnerabilty to all other threats	Cryptographic checksums
Confidentiality	• Eavesdropping on the net • Theft of info from server • Theft of data from client • Info about network configuration • Info about which client talks to server	• Loss of information • Loss of privacy	Encryption, Web proxies
Denial of Service	• Killing of user threads • Flooding machine with bogus requests • Filling up disk or memory • Isolating machine by DNS attacks	• Disruptive • Annoying • Prevent user from getting work done	Difficult to prevent
Authentication	• Impersonation of legitimate users • Data forgery	• Misrepresentation of user • Belief that false information is valid	Cryptographic techniques

SECURE SOCKET LAYER

- Secure Socket Layer (SSL) provides security services between TCP and applications that use TCP. The Internet standard version is called Transport Layer Service (TLS).
- SSL/TLS provides confidentiality using symmetric encryption and message integrity using a message authentication code.
- SSL/TLS includes protocol mechanisms to enable two TCP users to determine the security mechanisms and services they will use.
- Netscape originated SSL.

SSL Architecture

SSL is designed to make use of TCP to provide a reliable end-to-end secure service.

SSL is not a single protocol but rather two layers of protocols, as illustrated inFigure 16.2.

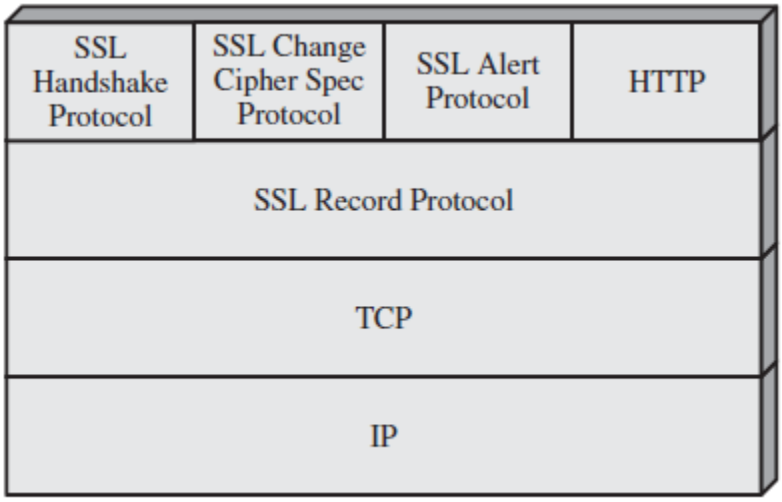


Figure 16.2 SSL Protocol Stack

The SSL Record Protocol provides basic security services to various higher-layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/server interaction, can operate on top of SSL. Three higher-layer protocols are defined as part of SSL: the Handshake Protocol, The Change Cipher Spec Protocol, and the Alert Protocol. These SSL-specific protocols are used in the management of SSL exchanges.

Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows.

- **Connection:** A connection is a transport (in the OSI layering model definition) that provides a suitable type of service.Every connection is associated with one session.
- **Session:** An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters which can be shared among multiple connections..

A session state is defined by the following parameters.

- **Session identifier:** An arbitrary byte sequence chosen by the server to identify an active or resumable session state.
- **Peer certificate:** An X509.v3 certificate of the peer. This element of the state may be null.
- **Compression method:** The algorithm used to compress data prior to encryption.

- **Cipher spec:** Specifies the bulk data encryption algorithm (such as null,AES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation. It also defines cryptographic attributes such as the hash_size.
- **Master secret:** 48-byte secret shared between the client and server.
- **Is resumable:** A flag indicating whether the session can be used to initiate new connections.

A connection state is defined by the following parameters

- **Server and client random**: Byte sequences that are chosen by the server and client for each connection.
- **Server write MAC secret:** The secret key used in MAC operations on data sent by the server.
- **Client write MAC secret:** The secret key used in MAC operations on data sent by the client.
- **Server write key:** The secret encryption key for data encrypted by the server and decrypted by the client.
- **Client write key:** The symmetric encryption key for data encrypted by the client and decrypted by the server.
- **Initialization vectors**: When a block cipher in CBC mode is used, an initialization vector (IV) is maintained for each key. This field is first initialized by the SSL Handshake Protocol.
- **Sequence numbers:** Each party maintains separate sequence numbers for transmitted and received messages for each connection.

SSL Record Protocol

The SSL Record Protocol provides two services for SSL connections:

- **Confidentiality:** The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.
- **Message Integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

Figure 16.3 indicates the overall operation of the SSL Record Protocol. The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled before being delivered to higher-level users.

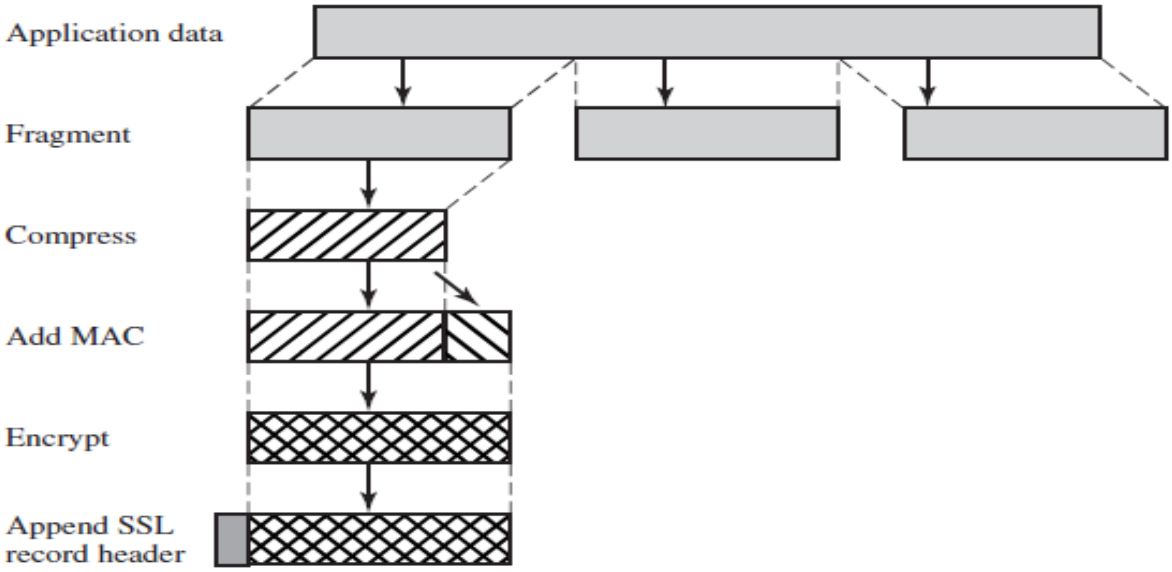


Figure 16.3 SSL Record Protocol Operation

- The first step is **fragmentation**. Each upper-layer message is fragmented into blocks of 2^{14} bytes (16384 bytes) or less.
- Next, **compression** is optionally applied. Compression must be lossless and may not increase the content length by more than 1024 bytes.1In SSLv3 (as well as the current version of TLS), no compression algorithm is specified, so the default compression algorithm is null.
- The next step in processing is to compute a **message authentication code** over the compressed data. For this purpose, a shared secret key is used.
- The next step is perform encryption and adds a header

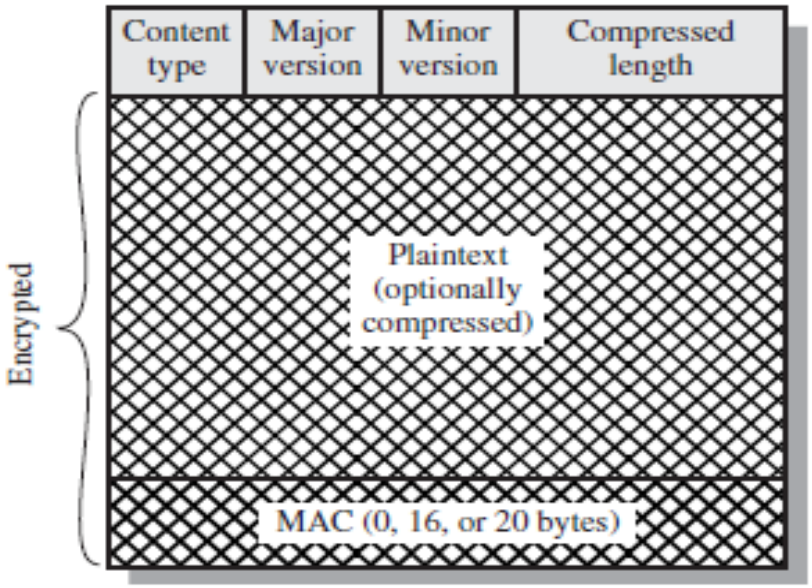


Figure 16.4 SSL Record Format

SSL Handshake Protocol

This phase is used to initiate a logical connection between client and server

The Handshake Protocol consists of a series of messages exchanged by client and server. All of these have the format shown in Figure 16.5c. Each message has three fields:

- **Type (1 byte):** Indicates one of 10 messages. Table 16.2 lists the defined message types.
- **Length (3 bytes):** The length of the message in bytes.
- **Content (≥ 0 bytes):** The parameters associated with this message; these are listed in Table 16.2.

Table 16.2 SSL Handshake Protocol Message Types

Message Type	Parameters
`hello_request`	null
`client_hello`	version, random, session id, cipher suite, compression method
`server_hello`	version, random, session id, cipher suite, compression method
`certificate`	chain of X.509v3 certificates
`server_key_exchange`	parameters, signature
`certificate_request`	type, authorities
`server_done`	null
`certificate_verify`	signature
`client_key_exchange`	parameters, signature
`finished`	hash value

- It consists of 4 phases
    1. Establish Security Capabilities
    2. Server Authentication and Key Exchange
    3. Client Authentication and Key Exchange
    4. Finish

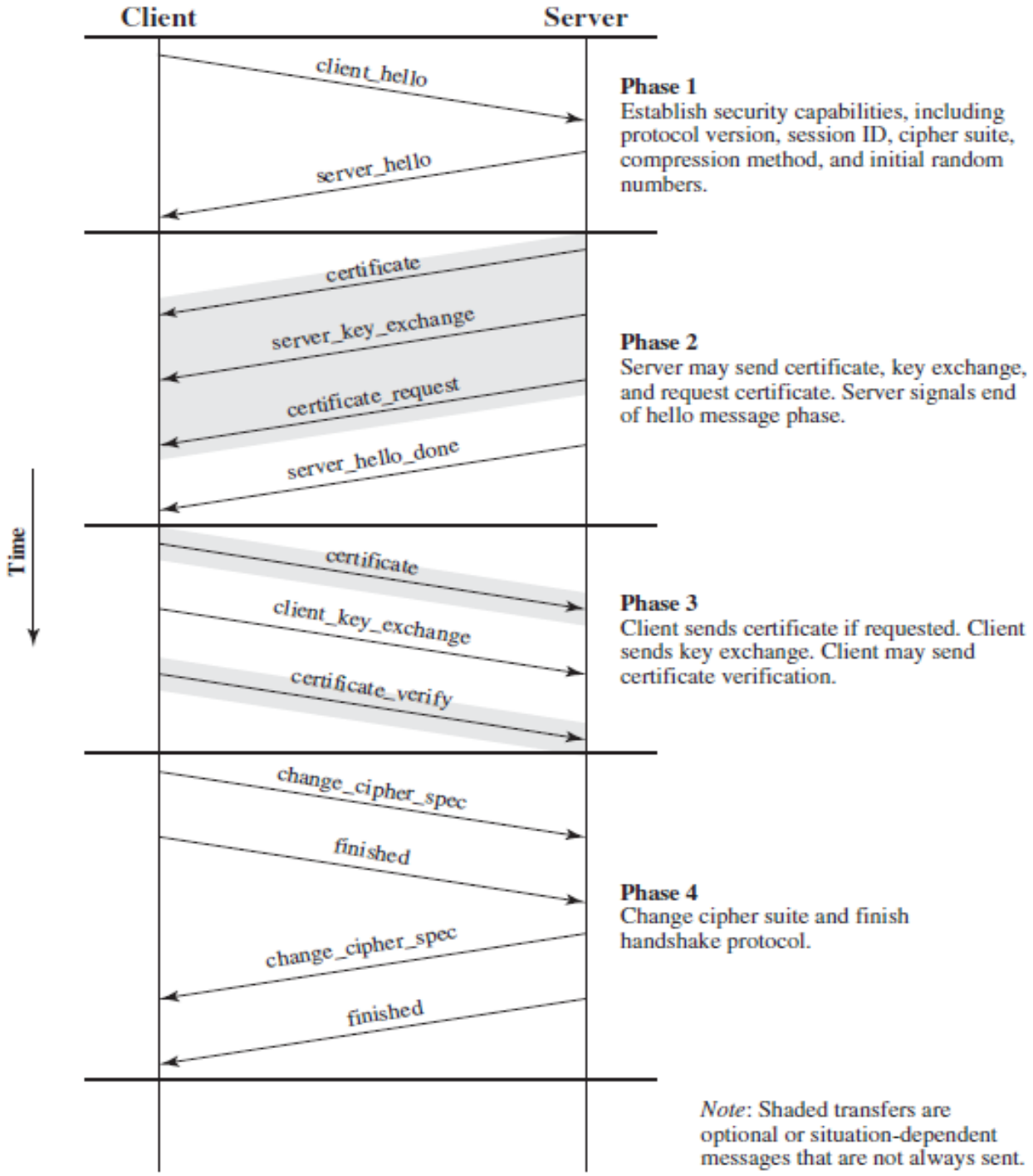


Figure 16.6 Handshake Protocol Action

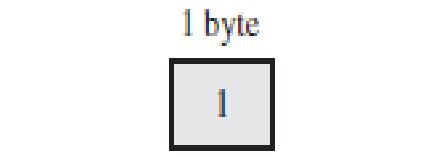
The exchange is initiated by the client, which sends a **client_hello message** with the following parameters:

- **Version:** The highest SSL version understood by the client.

- **Random:** A client-generated random structure consisting of a 32-bit timestamp and 28 bytes generated by a secure random number generator. These values serve as nonces and are used during key exchange to prevent replay attacks.
- **Session ID:** A variable-length session identifier. A nonzero value indicates that the client wishes to update the parameters of an existing connection or to create a new connection on this session. A zero value indicates that the client wishes to establish a new connection on a new session.
- **CipherSuite:** This is a list that contains the combinations of cryptographic algorithms supported by the client, in decreasing order of preference. Each element of the list (each cipher suite) defines both a key exchange algorithm and a CipherSpec; these are discussed subsequently.
- **Compression Method**: This is a list of the compression methods the client supports.

Change Cipher Spec Protocol

- The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol, and it is the simplest.
- This protocol consists of a single message (Figure 16.5a), which consists of a single byte with the value 1.
- The sole purpose of this message is to cause the pending state to be copied into the current state,

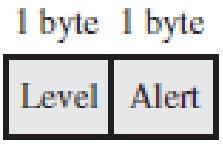


(a) Change Cipher Spec Protocol

Alert Protocol

- The Alert Protocol is used to convey SSL-related alerts to the peer entity
- Each message in this protocol consists of two bytes (Figure 16.5b). The first byte takes the value warning (1) or fatal (2) to convey the severity of the message.

- If the level is fatal, SSL immediately terminates the connection
- The second byte contains a code that indicates the specific alert.



(b) Alert Protocol

Example Alerts

fatal: unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter

warning: close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown

unexpected_message: An inappropriate message was received.

bad_record_mac: An incorrect MAC was received.

TRANSPORT LAYER SECURITY

TLS was released in response to the Internet community's demands for a standardized protocol. TLS (Transport Layer Security), defined in RFC 2246, is a protocol for establishing a secure connection between a client and a server.

TLS (Transport Layer Security) is capable of authenticating both the client and the server and creating a encrypted connection between the two.

Many protocols use TLS (Transport Layer Security) to establish secure connections, including HTTP, IMAP, POP3, and SMTP.

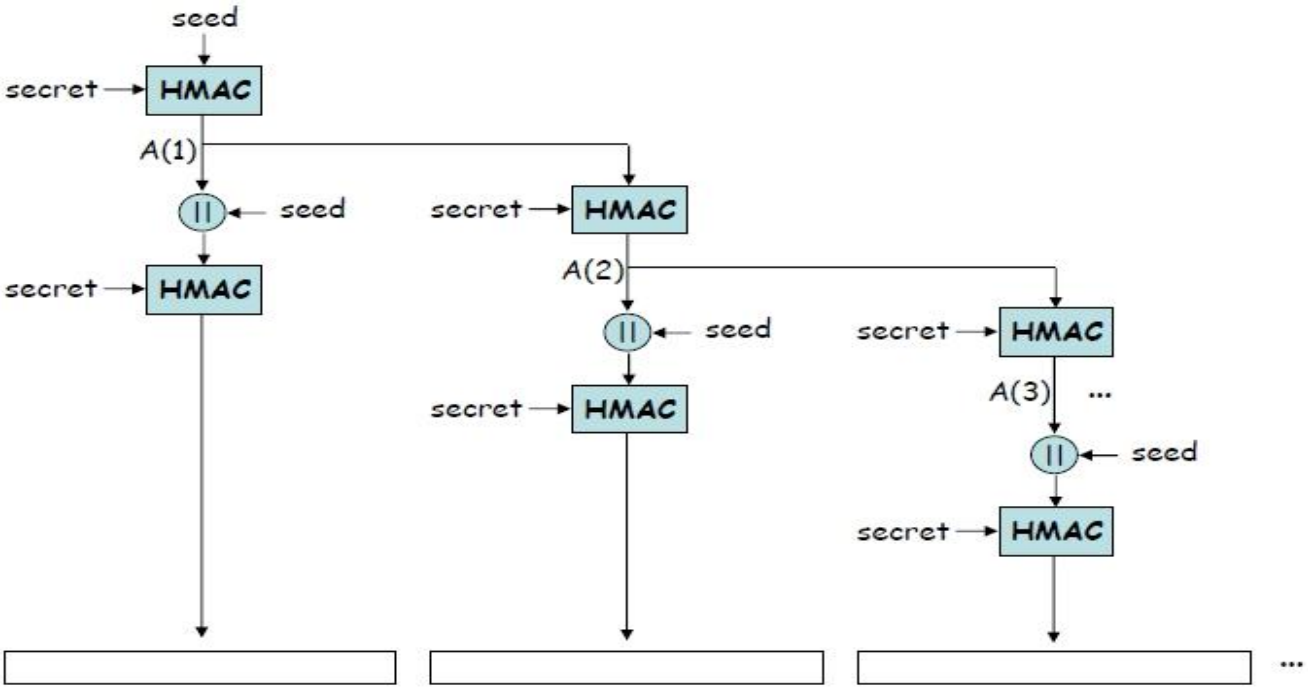
The TLS Handshake Protocol first negotiates key exchange using an asymmetric algorithm such as RSA or Diffie- Hellman .

TLS is very similar to SSLv3.

There are some minor differences ranging from protocol version numbers to generation of key material.

Version Number: The TLS Record Format is the same as that of the SSL Record Format and the fields in the header have the same meanings. The one difference is in version values. For the current version of TLS, the Major Version is 3 and the Minor Version is 3.

Message Authentication Code: Two differences arise one being the actual algorithm and the other being scope of MAC calculation. TLS makes use of the HMAC algorithm defined in RFC 2104. SSLv3 uses the same algorithm, except that the padding bytes are concatenated with the secret key rather than being XORed with the secret key padded to the block length. For TLS, the MAC calculation encompasses the fields.




```
HMAC hash(MAC write secret, seq num || TLSCompressed.type ||
          TLSCompressed.version || TLSCompressed.length ||
          TLSCompressed.fragment)
```

TheMACcalculationcoversallofthefieldscoveredbytheSSLv3calculation,plusthefield TLSCompressed. version, whichis the version of the protocolbeing employed.

Pseudorandom Function:TLSmakes use of apseudorandom function referred to asPRF to expandsecrets intoblocks of data forpurposes of key generation orvalidation.ThePRFis based on the followingdataexpansionfunction:

```
P hash(secret, seed) = HMAC hash(secret, A(1)  || seed) ||
                       HMAC hash(secret, A(2)  || seed) ||
                       HMAC hash(secret, A(3) || seed) || ...
```

where A() is defined as

```
A(0) = seed
A(i) = HMAC_hash (secret, A(i - 1))
```

Alert Codes

TLS supports all of the alert codes defined in SSLv3 with the exception ofno_certificate .A number of additional codes is defined in TLS. Some of them are

- **record_overflow**
- **unknown_ca**
- **access_denied**
- **protocol_version**
- **internal_error**
- **decrypt_error**

SECURE SHELL (SSH)

Secure Shell (SSH) is a protocol for secure network communications designed to be relatively simple and inexpensive to implement.The initial version, SSH1 was focused on providing a secure remote logon facility to replace TELNET and other remote logon schemes that provided no security. SSH also provides a more general client/server capability and can be used for such network functions as file transfer ande-mail. A new version, SSH2, fixes a number of security flaws in the original scheme.

SSH2 is documented as a proposed standard in IETF RFCs 4250 through 4256.

SSH client and server applications are widely available for most operating systems.

It has become the method of choice for remote login and X tunneling and israpidly becoming one of the most pervasive applications for encryption technologyoutside of embedded systems.

SSH is organized as three protocols that typically run on top of TCP(Figure 16.8):

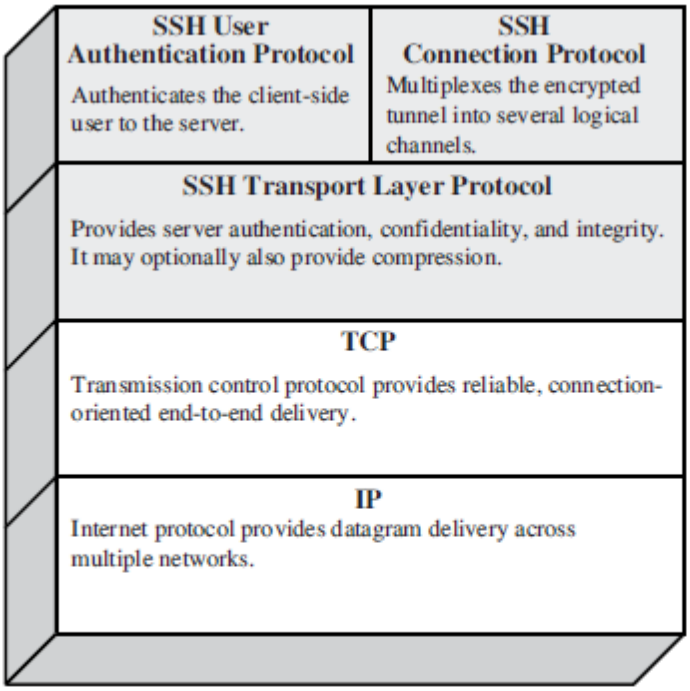


Figure 16.8 SSH Protocol Stack

1) Transport Layer Protocol: Provides server authentication, data confidentiality,and data integrity with forward secrecy (i.e., if a key is compromised duringone session, the knowledge does not affect the security of earlier sessions).Thetransport layer may optionally provide compression..

Once the connection is established, the client and server exchange data,referred to as packets, in the data field of a TCP segment. Each packet is in thefollowing format (Figure 16.10).

• **Packet length:** Length of the packet in bytes, not including the packet lengthand MAC fields.

• **Padding length:** Length of the random padding field.

• **Payload:** Useful contents of the packet. Prior to algorithm negotiation, thisfield is uncompressed. If compression is negotiated, then in subsequent packets,this field is compressed.

• **Random padding:** Once an encryption algorithm has been negotiated this field is added.

- **Message authentication code (MAC):** If message authentication has been negotiated, this field contains the MAC value.

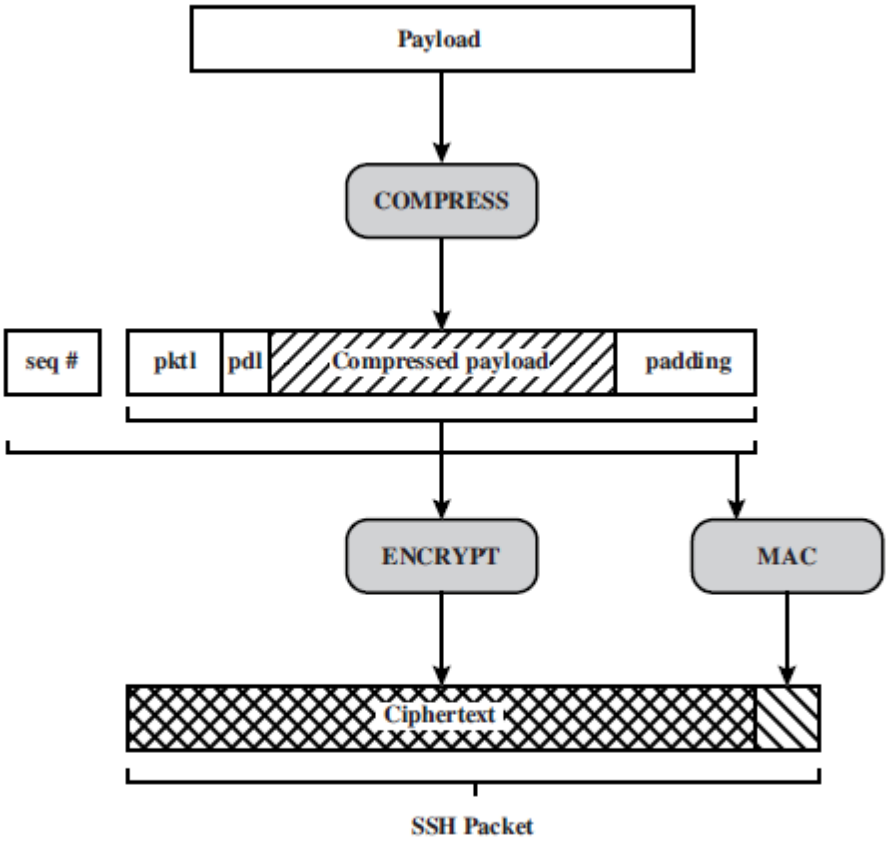


Figure 16.10 SSH Transport Layer Protocol Packet Formation

2) User Authentication Protocol: Authenticates the user to the server.

Authentication Methods:The server may require one or more of the following authentication methods

- **Public key:** The details of this method depend on the public-key algorithm chosen. In essence, the client sends a message to the server that contains the client's public key, with the message signed by the client's private key.When the server receives this message, it checks whether the supplied key is acceptable for authentication and, if so, it checks whether the signature is correct.

• **Password:** The client sends a message containing a plaintext password,which is protected by encryption by the Transport Layer Protocol.

• **host based:** Authentication is performed on the client's host rather than theclient itself

3) Connection Protocol: Multiplexes multiple logical communications channelsover a single, underlying SSH connection.

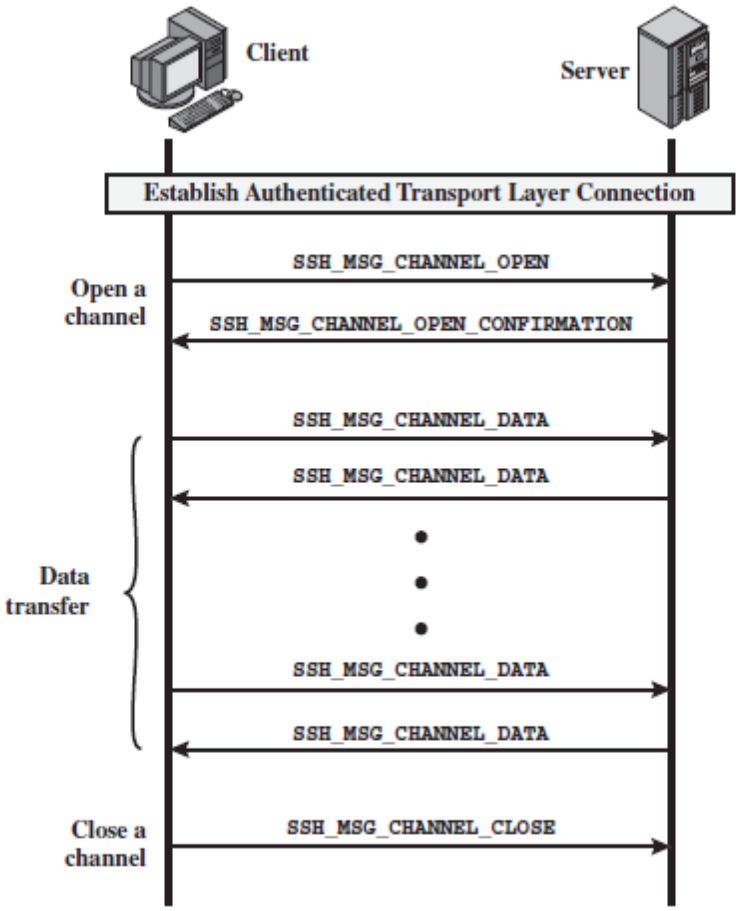


Figure 16.11 Example SSH Connection Protocol Message Exchange

ELECTRONIC MAIL SECURITY

The protection of email from unauthorized access and inspection is known as **electronic privacy**. There are mainly two methods for proving security for electronic mails

- Pretty Good Privacy
- S/MIME

Pretty Good Privacy:

In virtually all distributed environments, electronic mail is the most heavily used network based application.

Introduction:

- PGP is data encryption and decryption computer program that provides privacy (Confidentiality) and authentication for data communication.
- It was created by Phil Zimmermann in 1991

Use of PGP:

- It is used in Electronic mail
- File storage applications.
- PGP is an open-source, freely available software package for e-mail security. It provides **authentication** through the use of digital signature, **confidentiality** through the use of symmetric block encryption, **compression** using the ZIP algorithm, and **e-mail compatibility** using the radix-64 encoding scheme.
- PGP incorporates tools for developing a public-key trust model and public-key certificate management

PGP has grown explosively and is now widely used, because of following reasons:

1. It is available free worldwide in versions that run on a variety of platforms, including Windows, UNIX, Macintosh, and many more. In addition, the commercial version satisfies users who want a product that comes with vendor support.
2. It is based on algorithms that have survived extensive public review and are considered extremely secure. Specifically, the package includes RSA, DSS, and Diffie-Hellman for public-key encryption; CAST-128, IDEA, and 3DES for symmetric encryption; and SHA-1 for hash coding.

3. It has a wide range of applicability, from corporations that wish to select and enforce a standardized scheme for encrypting files and messages to individuals who wish to communicate securely with others worldwide over the Internet and other networks.
4. It was not developed by, nor is it controlled by, any governmental or standards organization. For those with an instinctive distrust of "the establishment," this makes PGP attractive.

NOTATIONS

The following symbols are used in PGP

K_s = session key used in symmetric encryption scheme
PR_a = private key of user A, used in public-key encryption scheme
PU_a = public key of user A, used in public-key encryption scheme
EP = public-key encryption
DP = public-key decryption
EC = symmetric encryption
DC = symmetric decryption
H = hash function
|| = concatenation
Z = compression using ZIP algorithm
R64 = conversion to radix 64 ASCII format[1]

PGP SERVICES

1) authentication
2) confidentiality
3) compression
4) e-mail compatibility

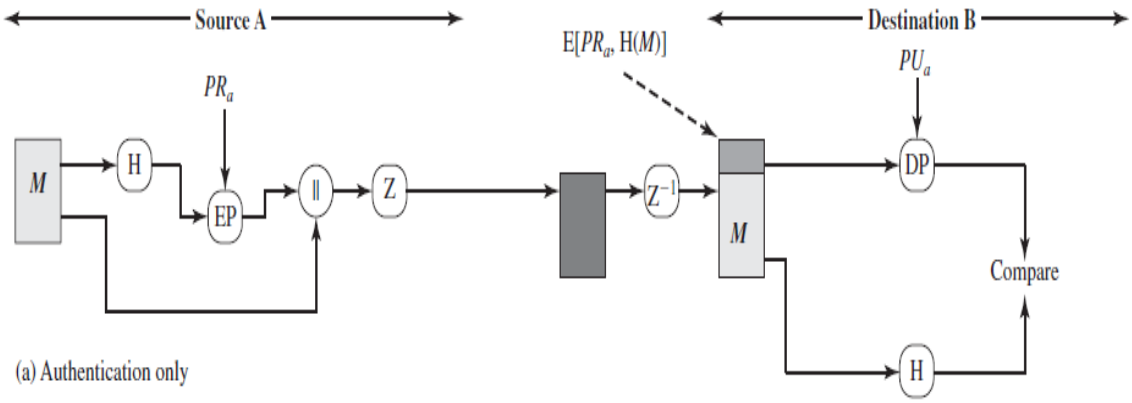
Function	Algorithms Used	Description

Digital signature	DSS/SHA or RSA/SHA	A hash code of a message is created using SHA-1.This message digest is encrypted using DSS or RSA with the sender's private key and included with the message.
Message encryption	CAST or IDEA or Three-key Triple DES with Diffie-Hellman or RSA	A message is encrypted using CAST-128 or IDEA or 3DES with a one-time session key generated by the sender. The session key is encrypted using Diffie-Hellman or RSA with the recipient's public key and included with the message.
Compression	ZIP	A message may be compressed for storage or transmission using ZIP.
E-mail compatibility	Radix-64 conversion	To provide transparency for e-mail applications, an encrypted message may be converted to an ASCII string using radix-64 conversion.

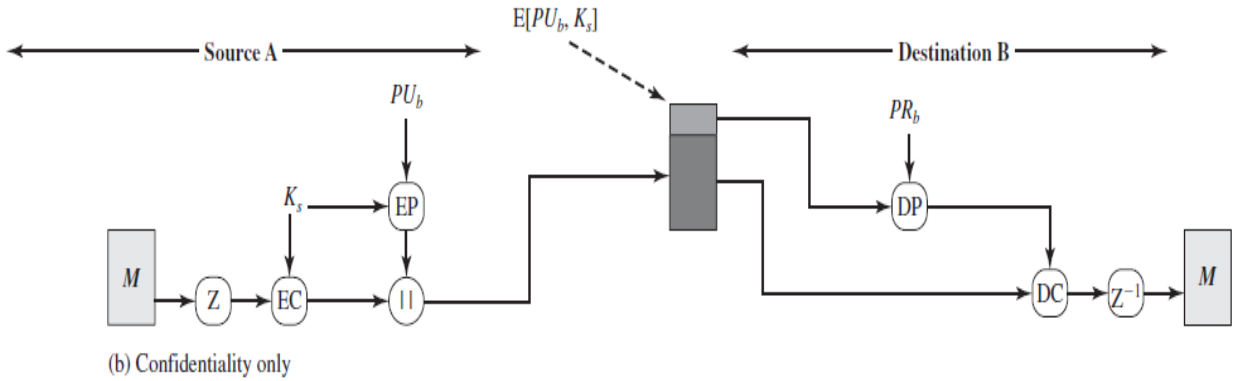
Providing authentication by using PGP:

The sequence steps for providing authentication by using PGP

1. The sender creates a message.
2. SHA-1 is used to generate a 160-bit hash code of the message.
3. The hash code is encrypted with RSA using the sender's private key, and the result is prepended to the message.
4. The receiver uses RSA with the sender's public key to decrypt and recover the hash code.
5. The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic.



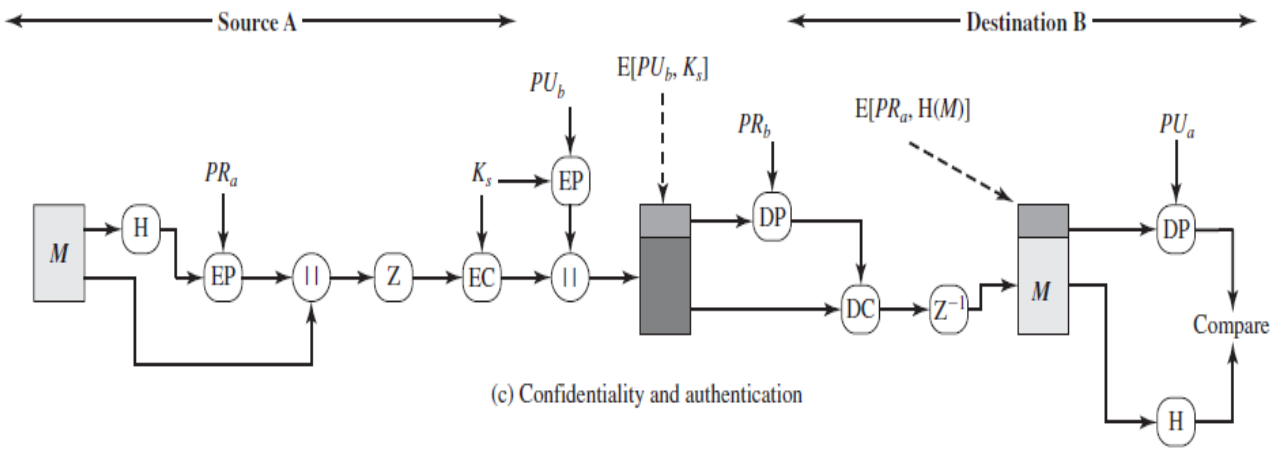
Confidentiality by using PGP



Steps for providing confidentiality:

1. The sender generates a message and a random 128-bit number to be used as a session key for this message only.
2. The message is encrypted using CAST-128 (or IDEA or 3DES) with the session key.
3. The session key is encrypted with RSA using the recipient's public key and is prepended to the message.
4. The receiver uses RSA with its private key to decrypt and recover the session key.
5. The session key is used to decrypt the message.

PGP for both authentication and confidentiality:



(c) Confidentiality and authentication

COMPRESSION

As a default, PGP compresses the message after applying thesignature but before Encryption.This has the benefit of saving space both for e-mailtransmissionandforfilestorage.

The signature is generated before compression for two reasons

- so can store uncompressed message & signature for later verification
- Message encryption is applied after compression to strengthen cryptographicsecurity. Because the compressed message has less redundancy than theoriginal plaintext, cryptanalysis is more difficult.

PGP uses ZIP compression algorithm

E-MAIL COMPATIBILITY

When PGP is used, at least part of the block to be transmittedis encrypted. If only the signature service is used, then the message digest isencrypted (with the sender's private key). If the confidentiality service is used, themessage plus signature (if present) are encrypted (with a one-time symmetric key).Thus, part or the entire resulting block consists of a stream of arbitrary 8-bit octets.

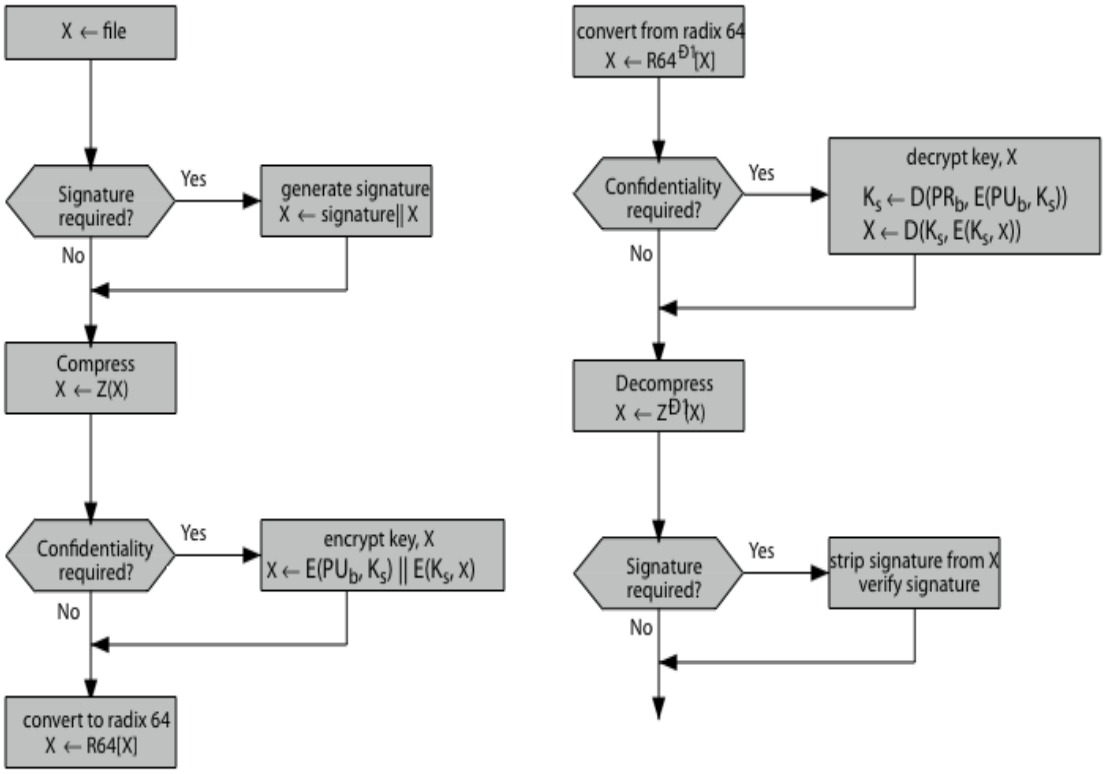
However, many electronic mail systems only permit the use of blocks consisting ofASCII text.To accommodate this restriction, PGP provides the service of convertingthe raw 8-bit binary stream to a stream of printable ASCII characters.

The scheme used for this purpose is radix-64 algorithm

- It maps 3 bytes to 4 printable chars
- It also appends a CRC to detect transmission errors

PGP also segments messages if too big

PGP Operation – Summary



(a) Generic Transmission Diagram (from A)

(b) Generic Reception Diagram (to B)

S/MIME

- Secure/Multipurpose Internet Mail Extension is a security enhancement to the MIME internet email standard.
- S/MIME is for industry standard for commercial and organizational use.
- It defined in number of documents that is RFC 2630, RFC 2632, RFC 2633

E-mail format standard, RFC 822, which is still in common use. The most recent version of this format specification is RFC 5322 (Internet Message Format).

RFC 5322 (Internet Message Format).

RFC 5322 defines a format for text messages that are sent using electronic mail. It has been the standard for Internet-based text mail messages and remains in common use.

Message Structure

- A message consists of some number of header lines (*the header*) followed by unrestricted text (*the body*).
- A header line usually consists of a keyword, followed by a colon, followed by the keyword's arguments

  Example

  Date: October 8, 2009 2:15:49 PM EDT
  From: William Stallings <ws@shore.net>
  Subject: The Syntax in RFC 5322
  To: Smith@Other-host.com
  Cc: Jones@Yet-Another-Host.com

  Hello. This section begins the actual
  message body, which is delimited from the
  message heading by a blank line.

Multipurpose Internet Mail Extensions:

Multipurpose Internet Mail Extension (MIME) is an extension to the RFC 5322 framework that is intended to address some of the problems and limitations of the use of Simple Mail Transfer Protocol (SMTP).

The following are limitations of the SMTP/5322 scheme.

1. SMTP cannot transmit executable files or other binary objects. A number of schemes are in use for converting binary files into a text form that can be used by SMTP mail systems.
2. SMTP cannot transmit text data that includes national language characters, because these are represented by 8-bit codes, and SMTP is limited to 7-bit ASCII.
3. SMTP servers may reject mail message over a certain size.
4. SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.
5. SMTP gateways to X.400 electronic mail networks cannot handle non-textual data included in X.400 messages.
6. Some SMTP implementations do not adhere completely to the SMTP standards defined in RFC 821. Common problems include:
    a. Deletion, addition, or reordering of carriage return and linefeed
    b. Truncating or wrapping lines longer than 76 characters
    c. Removal of trailing white space (tab and space characters)
    d. Padding of lines in a message to the same length
    e. Conversion of tab characters into multiple space characters

MIME is intended to resolve these problems in a manner that is compatible with existing RFC 5322 implementations. The specification is provided in RFCs 2045 through 2049.

MIME has 5 header fields

The five header fields defined in MIME are

1. **MIME-Version**: Must have the parameter value 1.0.
2. **Content-Type**: Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the datato the user
3. **Content-Transfer-Encoding**: Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.

4. **Content-ID**: Used to identify MIME entities uniquely in multiple contexts.
5. **Content-Description**: A text description of the object with the body; this is useful when the object is not readable (e.g., audio data).

Mime Content Types

Describes the data contained in the body with sufficient detail

Table 18.3 MIME Content Types

Type	Subtype	Description
Text	Plain	Unformatted text; may be ASCII or ISO 8859.
	Enriched	Provides greater format flexibility.
Multipart	Mixed	The different parts are independent but are to be transmitted together. They should be presented to the receiver in the order that they appear in the mail message.
	Parallel	Differs from Mixed only in that no order is defined for delivering the parts to the receiver.
	Alternative	The different parts are alternative versions of the same information. They are ordered in increasing faithfulness to the original, and the recipient's mail system should display the "best" version to the user.
	Digest	Similar to Mixed, but the default type/subtype of each part is message/rfc822.
Message	rfc822	The body is itself an encapsulated message that conforms to RFC 822.
	Partial	Used to allow fragmentation of large mail items, in a way that is transparent to the recipient.
	External-body	Contains a pointer to an object that exists elsewhere.
Image	jpeg	The image is in JPEG format, JFIF encoding.
	gif	The image is in GIF format.
Video	mpeg	MPEG format.
Audio	Basic	Single-channel 8-bit ISDN mu-law encoding at a sample rate of 8 kHz.
Application	PostScript	Adobe Postscript format.
	octet-stream	General binary data consisting of 8-bit bytes.

Example

Here is a simple example of a multipart message containing two parts--**both consisting of simple text**

From: Nathaniel Borenstein nsb@bellcore.com>

To: Ned Freed <ned@innosoft.com>

Subject: Sample message

MIME-Version: 1.0

Content-type: multipart/mixed; boundary="simple boundary“

This is the preamble. It is to be ignored, though it

is a handy place for mail composers to include an

explanatory note to non-MIME conformant readers.

—simple boundary

Content-type: text/plain; charset=us-ascii

This is explicitly typed plain ASCII text. It DOES end with a line break.

S/MIME Functionality

In terms of general functionality, S/MIME is very similar to PGP. Both offer the ability to sign and/or encrypt messages. In this subsection, we briefly summarize S/MIME capability.

S/MIME provides the following functions.

- **Enveloped data:** This consists of encrypted content of any type and encrypted-content encryption keys for one or more recipients.
- **Signed data:** A digital signature is formed by taking the message digest of the content to be signed and then encrypting that with the private key of the signer. The content plus signature are then encoded using base64 encoding. A signed data message can only be viewed by a recipient with S/MIME capability.
    - encoded (message + signed digest)

- **Clear-signed data:** As with signed data, a digital signature of the content is formed. However, in this case, only the digital signature is encoded using base64.As a result, recipients without S/MIME capability can view the message content, although they cannot verify the signature.
    - clear text message + encoded (signed digest)

- **Signed and enveloped data:** Signed-only and encrypted-only entities may be nested, so that encrypted data may be signed and signed data or clear-signed data may be encrypted.
    - nesting of signed & encrypted entities

CRYPTOGRAPHIC ALGORITHMS

S/MIME uses the following terminology taken from RFC 2119 (Key Words for use in RFCs to Indicate Requirement Levels) to specify the requirement level:

- **MUST**: The definition is an absolute requirement of the specification. An implementation must include this feature or function to be in conformance with the specification.
- **SHOULD**: There may exist valid reasons in particular circumstances to ignore this feature or function, but it is recommended that an implementation include the feature or function.

Table 18.6 Cryptographic Algorithms Used in S/MIME

Function	Requirement
Create a message digest to be used in forming a digital signature.	MUST support SHA-1. Receiver SHOULD support MD5 for backward compatibility.
Encrypt message digest to form a digital signature.	Sending and receiving agents MUST support DSS. Sending agents SHOULD support RSA encryption. Receiving agents SHOULD support verification of RSA signatures with key sizes 512 bits to 1024 bits.
Encrypt session key for transmission with a message.	Sending and receiving agents SHOULD support Diffie-Hellman. Sending and receiving agents MUST support RSA encryption with key sizes 512 bits to 1024 bits.
Encrypt message for transmission with a one-time session key.	Sending and receiving agents MUST support encryption with tripleDES. Sending agents SHOULD support encryption with AES. Sending agents SHOULD support encryption with RC2/40.
Create a message authentication code.	Receiving agents MUST support HMAC with SHA-1. Sending agents SHOULD support HMAC with SHA-1.

S/MIME incorporates three public-key algorithms. The Digital Signature Standard (DSS) is the preferred algorithm for digital signature.

CRYPTOGRAPHY AND NETWORK SECURITY
UNIT-6

Syllabus: IP Security & Intrusion Detection Systems

IP Security: IP Security Overview, IP Security Architecture, Authentication Header, Encapsulating Security Payload, Combining Security Associations and Key Management.

Intrusion detection: Overview, Approaches for IDS/IPS, Signature based IDS, Host based IDS/IPS.

6.1 IP SECURITY OVERVIEW:

IPSec is an Internet Engineering Task Force (IETF) standard suite of protocols that provides data authentication, integrity, and confidentiality as data is transferred between communication points across IP networks. IPSec provides data security at the IP packet level. A packet is a data bundle that is organized for transmission across a network, and it includes a header and payload (the data in the packet). IPSec emerged as a viable network security standard because enterprises wanted to ensure that data could be securely transmitted over the Internet. IPSec protects against possible security exposures by protecting data while in transit.

IPSEC SECURITY FEATURES:

IPSec is the most secure method commercially available for connecting network sites. IPSec was designed to provide the following security features when transferring packets across networks:

- Authentication: Verifies that the packet received is actually from the claimed sender.
- Integrity: Ensures that the contents of the packet did not change in transit.
- Confidentiality: Conceals the message content through encryption.

IPSEC ELEMENTS:

IPSec contains the following elements:

- **Encapsulating Security Payload (ESP):** Provides confidentiality, authentication, and integrity.
- **Authentication Header (AH):** Provides authentication and integrity.
- **Internet Key Exchange (IKE):** Provides key management and Security Association (SA) management.

APPLICATIONS OF IPSEC:

IPSec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet. Examples of its use include the following:

- Secure branch office connectivity over the Internet
- Secure remote access over the Internet
- **Establishing extranet and intranet connectivity with partners**: IPSec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.
- **Enhancing electronic commerce security:** Even though some Web and electronic commerce applications have built-in security protocols, the use of IPSec enhances that security.

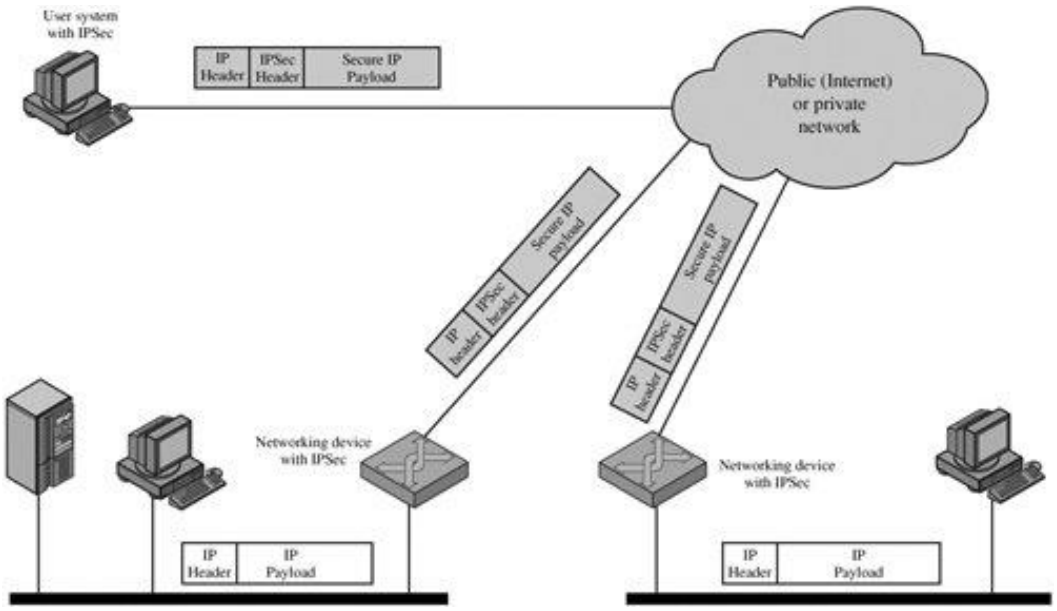


Figure. An IP Security Scenario

BENEFITS OF IPSEC:

- IPSec provides strong security within and across the LANs.
- Firewall uses IPSec to restrict all those incoming packets which are not using IP. Since firewall is the only way to enter into an organization, restricted packets cannot enter.
- IPSec is below the transport layer (TCP, UDP) and so is transparent to applications.
- There is no need to change software on a user or server system when IPSec is implemented in the firewall or router. Even if IPSec is implemented in end systems, upper-layer software, including applications, is not affected.
- IPSec can be transparent to end users.
- IPSec can provide security for individual users if needed.

6.2 IP SECURITY ARCHITECTURE:

Mainly the IPSec is constituted by three major components.

- IPSec Documents
- IPSec Services
- Security Associations(SA)

IPSec Documents:

The IPSec specification consists of numerous documents. The most important of these, issued in November of 1998, are RFCs 2401, 2402, 2406, and 2408:

- RFC 2401: An overview of a security architecture
- RFC 2402: Description of a packet authentication extension to IPv4 and IPv6
- RFC 2406: Description of a packet encryption extension to IPv4 and IPv6
- RFC 2408: Specification of key management capabilities

The header for authentication is known as the Authentication header(AH); that for encryption is known as the **Encapsulating Security Payload (ESP)** header.

The documents are divided into seven groups, as depicted in Figure

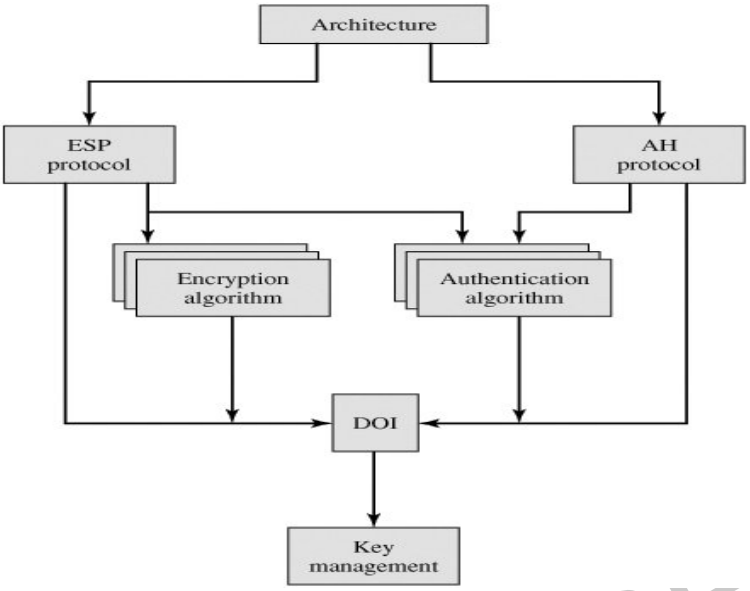


Figure. IPSec Document Overview

- **Architecture:** Covers the general concepts, security requirements, definitions, and mechanisms defining IPSec technology.
- **Encapsulating Security Payload (ESP):** Covers the packet format and general issues related to the use of the ESP for packet encryption and, optionally, authentication.
- **Authentication Header (AH):** Covers the packet format and general issues related to the use of AH for packet authentication.
- **Encryption Algorithm:** A set of documents that describe how various encryption algorithms are used for ESP.
- **Authentication Algorithm:** A set of documents that describe how various authentication algorithms are used for AH and for the authentication option of ESP.
- **Key Management:** Documents that describe key management schemes.
- **Domain of Interpretation (DOI):** Contains values needed for the other documents to relate to each other. These include identifiers for approved encryption and authentication algorithms, as well as operational parameters such as key lifetime.

IPSec Services:

IPSec provides security services at the IP layer by selecting required security protocols, algorithms and cryptographic keys as per the services requested.

Two protocols are used to provide security:

- an authentication protocol designated by the header of the protocol, **Authentication Header (AH)**
- and a combined encryption/authentication protocol designated by the format of the packet for that protocol, **Encapsulating Security Payload (ESP).**

The services are

- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets
- Confidentiality
- Limited traffic flow confidentiality

	AH	ESP (encryption only)	ESP (encryption plus authentication)
Access control	✔	✔	✔
Connectionless integrity	✔		✔
Data origin authentication	✔		✔
Rejection of replayed packets	✔	✔	✔
Confidentiality		✔	✔
Limited traffic flow confidentiality		✔	✔

Security Associations:

A key concept that appears in both the authentication and confidentiality mechanisms for IP is the security association (SA). An association is a one-way relationship between a sender and a receiver that affords security services to the traffic carried on it. If a peer relationship is needed, for two-way secure exchange, then two security associations are required. Security services are afforded to an SA for the use of AH or ESP, but not both.

A security association is uniquely identified by three parameters:

- **Security Parameters Index (SPI):** A bit string assigned to this SA and having local significance only. SPI is located in AH and ESP headers. SPI enables the receiving system under which the packet is to process.
- **IP Destination Address:** It is the end point address of SA which can be end user system or a network system.
- **Security Protocol Identifier:** security protocol identifier indicates whether the associations is an AH or ESP.

SA Parameters:

The implementation of IPSec contain SA database which identifies the parameters related to SA.

- **Sequence Number Counter**: A 32-bit value used to generate the Sequence Number field in AH or ESP headers.
- **Sequence Counter Overflow**: A flag indicating whether overflow of the Sequence Number Counter should generate an auditable event and prevent further transmission of packets on this SA.
- **Anti-Replay Window**: Used to determine whether an inbound AH or ESP packet is a replay
- **AH Information**: Authentication algorithm, keys, key lifetimes, and related parameters being used with AH.
- **ESP Information**: Encryption and authentication algorithm, keys, initialization values, key lifetimes, and related parameters being used with ESP (required for ESP implementations).
- **Lifetime of This Security Association**: A time interval or byte count after which an SA must be replaced with a new SA or terminated.

- **IPSec Protocol Mode:** This parameter represents the type of mode used for IPSec implementation. The mode may be a Tunnel or transport.
- **Path MTU:** Any observed path maximum transmission unit (maximum size of a packet that can be transmitted).

SA Selectors:

- IPSec provides flexibility in providing services to the users according to their needs. For this purpose, SA's are used. Different combinations of SA's can give different user configurations. IPSec is also capable of differentiating traffic i.e., which traffic is allowed to pass and which traffic should be forwarded the IPSec protection. The property of IPSec requires the traffic to be associated with a security association. To associate a particular SA to IP traffic IPSec maintains a database called Security Policy Database(SPD).
- SPD is table entries which maps a set of IP traffic to a single or more SAs.
- Selectors are basically used to define policy that specifies which packet should be forwarded and which packet should be rejected to filter outgoing traffic.

A sequence of steps is performed on the outgoing traffic,

- Compare the values of the appropriate fields in the packet (the selector fields) against the SPD to find a matching SPD entry, which will point to zero or more SAs.
- Determine the SA if any for this packet and its associated SPI. **Security Parameter Index(SPI)** is one of the fields of IPSec header which is a unique identifier to identify a security association.
- Do the required IPSec processing (i.e., AH or ESP processing).

The following selectors determine an SPD entry:

- **Destination IP Address:** This may be a single IP address, an enumerated list or range of addresses.
- **Source IP Address:** This may be a single IP address, an enumerated list or range of addresses.
- **UserID:** A user identifier from the operating system. This is not a field in the IP or upper-layer headers but is available if IPSec is running on the same operating system as the user.
- **Data Sensitivity Level:** Used for systems providing information flow security (e.g., Secret or Unclassified).
- **Transport Layer Protocol:** Obtained from the IPv4 Protocol or IPv6 Next Header field. This may be an individual protocol number, a list of protocol numbers, or a range of protocol numbers.
- **Source and Destination Ports:** These may be individual TCP or UDP port values, an enumerated list of ports, or a wildcard port.

Transport and Tunnel Modes:

- Both AH and ESP support two modes of use: **transport and tunnel mode**.
- The operation of these two modes is best understood in the context of a description of AH and ESP.

Transport Mode:

Transport mode provides protection primarily for upper-layer protocols. That is, transport mode protection extends to the payload of an IP packet. Transport mode is used for end-to-end communication between two hosts. ESP in transport mode encrypts and optionally authenticates the IP payload but not the IP header. AH in transport mode authenticates the IP payload and selected portions of the IP header.

Tunnel Mode:

Tunnel mode provides protection to the entire IP packet. To achieve this, after the AH or ESP fields are added to the IP packet, the entire packet plus security fields is treated as the payload of new "outer" IP packet with a new outer IP header. The entire original, or inner, packet travels through a "tunnel" from one point of an IP network to another; no routers along the way are able to examine the inner IP header. Because the original packet is encapsulated, the new, larger packet may have totally different source and destination addresses, adding to the security.

Tunnel mode is used when one or both ends of an SA are a security gateway, such as a firewall or router that implements IPSec. ESP in tunnel mode encrypts and optionally authenticates the entire inner IP packet, including the inner IP header. AH in tunnel mode authenticates the entire inner IP packet and selected portions of the outer IP header.

6.3AUTHENTICATION HEADER(AH):

The Authentication Header provides support for data integrity and authentication of IP packets. Data integrity service insures that data inside IP packets is not altered during the transit. The authentication feature enables an end system to authenticate the user or application and filter traffic accordingly. It also prevents the **address spoofing attacks** (A technique used to gain unauthorized access to computers, whereby the intruder sends messages to a computer with an IP **address** indicating that the message is coming from a trusted host).Authentication is based on the use of a message authentication code (MAC) i.e.; two communication parties must share a secret key.

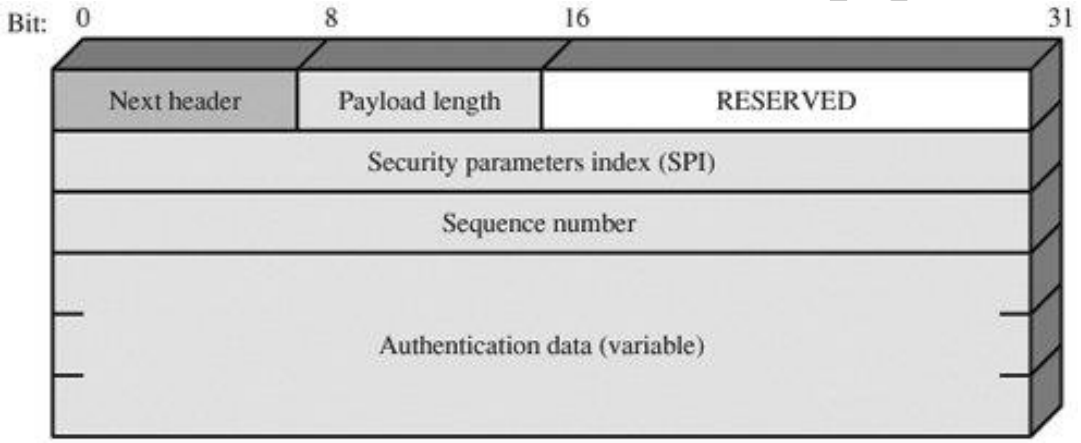


Figure. IPSec Authentication Header

The Authentication Header consists of the following fields

1. **Next Header (8 bits):** Identifies the type of header that immediately following the AH.
2. **Payload Length:** Length of Authentication Header in 32-bit words.
3. **Reserved (16 bits):** For future use.
4. **Security Parameters Index (32 bits):** Identifies a security association.
5. **Sequence Number (32 bits):** A monotonically increasing counter value.
6. **Authentication Data (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value (ICV), or MAC, for this packet.

Anti-Replay Service:

A replay attack is one in which an attacker obtains a copy of an authenticated packet and later transmits it to the intended destination. The receipt of duplicate, authenticated IP packets may disrupt service in some way or may have some other undesired consequence. The **Sequence Number** field is designed to stop such attacks.

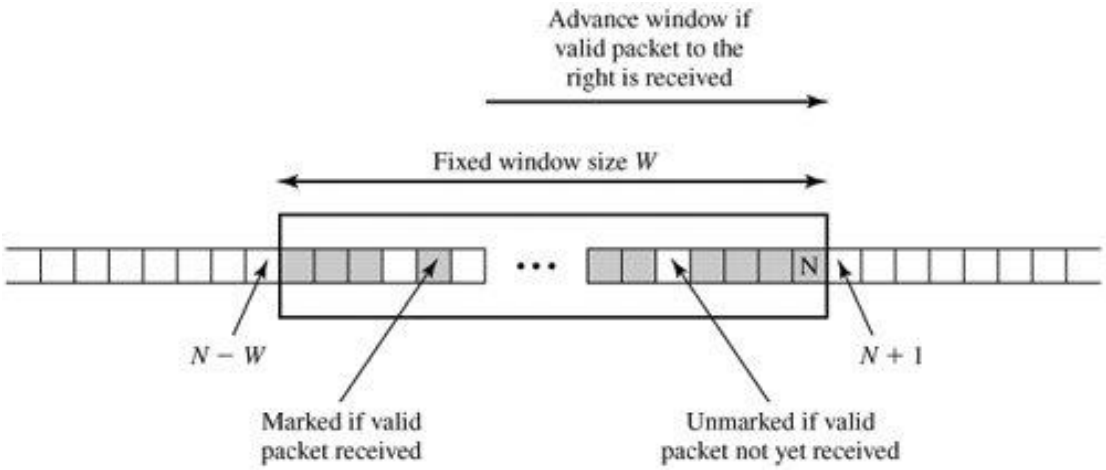


Figure. Antireplay Mechanism

When a new SA is established, the sender initializes a sequence number counter to 0. Each time that a packet is sent on this SA, the sender increments the counter and places the value in the Sequence Number field. Thus, the first value to be used is 1. If anti-replay is enabled (the default), the sender must not allow the sequence number to cycle past 2^{32}-1 back to zero. Otherwise, there would be multiple valid packets with the same sequence number. If the limit of 2^{32}-1 is reached, the sender should terminate this SA and negotiate a new SA with a new key. IP is a connectionless, unreliable service, the protocol does not guarantee that packets will be delivered in order and does not guarantee that all packets will be delivered. Therefore, the IPSec authentication document dictates that the receiver should implement a window of size W, with a default of W = 64. The right edge of the window represents the highest sequence number, N, so far received for a valid packet. For any packet with a sequence number in the range from N-W + 1 to N that has been correctly received (i.e., properly authenticated), the corresponding slot in the window is marked (Figure).

Inbound processing proceeds as follows when a packet is received:

1. If the received packet falls within the window and is new, the MAC is checked. If the packet is authenticated, the corresponding slot in the window is marked.
2. If the received packet is to the right of the window and is new, the MAC is checked. If the packet is authenticated, the window is advanced so that this sequence number is the right edge of the window, and the corresponding slot in the window is marked.
3. If the received packet is to the left of the window, or if authentication fails, the packet is discarded; this is an auditable event.

Integrity Check Value:

The Authentication Data field holds a value referred to as the Integrity Check Value. The ICV is a message authentication code or a truncated version of a code produced by a MAC algorithm.

Transport and Tunnel Modes:

There are two ways in which the IPSec authentication service can be used. In one case, **authentication is provided directly** between a server and client workstations; the workstation can be either on the same network as the server or on an external network. As long as the workstation and the server share a protected secret key, the authentication process is secure. This case uses a **transport mode SA**. In the other case, a **remote workstation authenticates itself to the corporate firewall**, either for access to the entire internal network or because the requested server does not support the authentication feature. This case uses a **tunnel mode SA**.

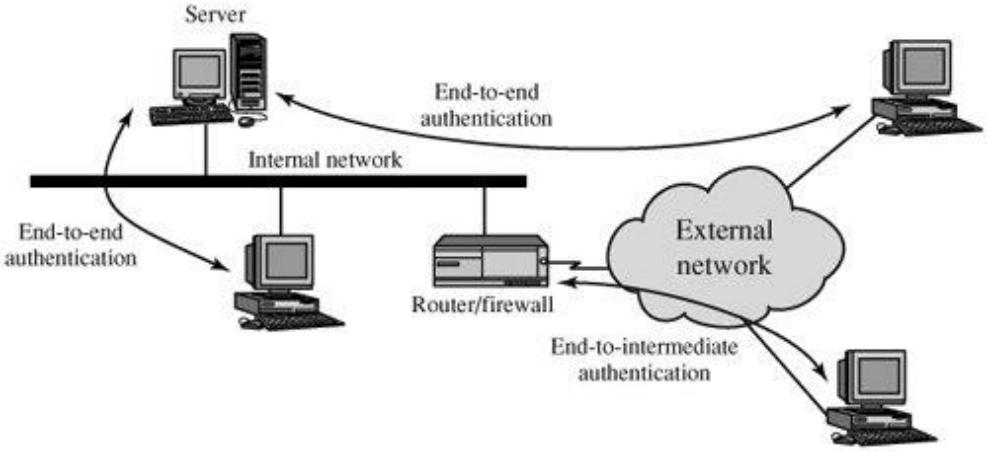


Figure. End-to-End versus End-to-Intermediate Authentication

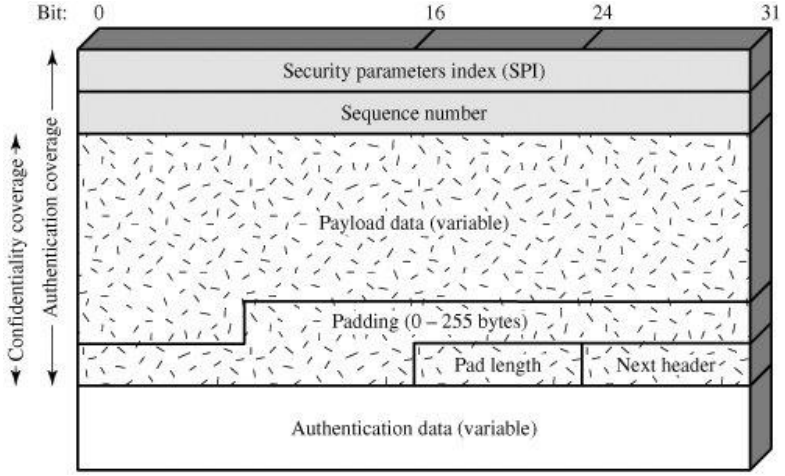
6.4. Encapsulating Security Payload(ESP):

The Encapsulating Security Payload provides confidentiality services, including confidentiality of message contents and limited traffic flow confidentiality. As an optional feature, ESP can also provide an authentication service.

ESP Format:

It contains the following fields:

1. **Security Parameters Index (32 bits):** Identifies a security association.
2. **Sequence Number (32 bits):** A monotonically increasing counter value; this provides an anti-replay function, as discussed for AH.
3. **Payload Data (variable):** This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption.
4. **Padding (0-255 bytes):** The purpose of this field is discussed later.
5. **Pad Length (8 bits):** Indicates the number of pad bytes immediately preceding this field.
6. **Next Header (8 bits):** Identifies the type of data contained in the payload data field by identifying the first header in that.
7. **Authentication Data (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value computed over the ESP packet minus the Authentication Data field.



Encryption and Authentication Algorithms:

The Payload Data, Padding, Pad Length, and Next Header fields are encrypted by the ESP.

Various algorithms used for encryption are: Three-key triple DES, RC5, IDEA, Three-key triple IDEA, CAST, Blowfish

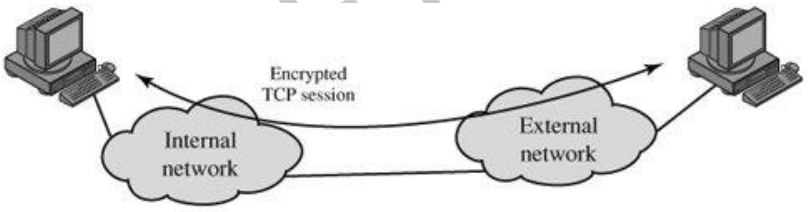
Padding:

The Padding field serves several purposes:

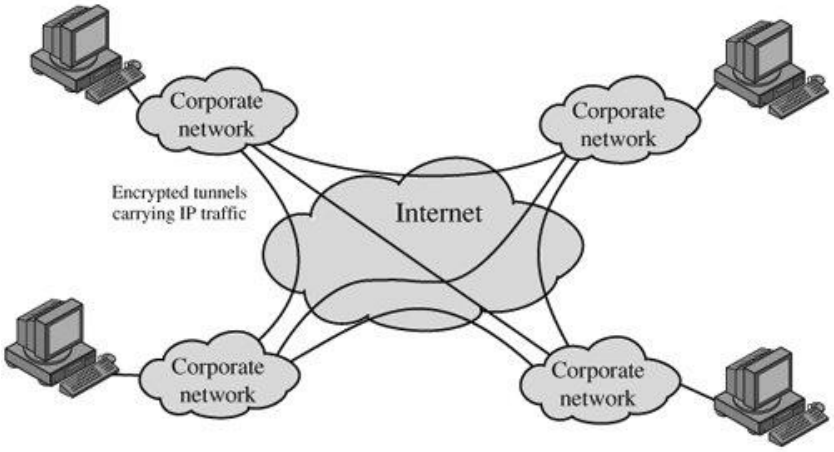
1. If an encryption algorithm requires the plaintext to be a multiple of some number of bytes. The Padding field is used to expand the plaintext to the required length.
2. The ESP format requires that the Pad Length and Next Header fields be right aligned within a 32-bit word. Equivalently, the ciphertext must be an integer multiple of 32 bits. The Padding field is used to assure this alignment.
3. Additional padding may be added to provide partial traffic flow confidentiality by concealing the actual length of the payload.

Transport and Tunnel Modes:

Figure shows two ways in which the IPSec ESP service can be used. In the upper part of the figure, encryption (and optionally authentication) is provided directly between two hosts. Figure(b) shows how tunnel mode operation can be used to set up a *virtual private network*. In this example, an organization has four private networks interconnected across the Internet. Hosts on the internal networks use the Internet for transport of data but do not interact with other Internet-based hosts. By terminating the tunnels at the security gateway to each internal network, the configuration allows the hosts to avoid implementing the security capability. The former technique is support by a transport mode SA, while the latter technique uses a tunnel mode SA.



(a) Transport-level security



(b) A virtual private network via tunnel mode

6.5 COMBINING SECURITY ASSOCIATIONS:

An individual SA can implement either the AH or ESP protocol but not both. Sometimes a particular traffic flow will call for the services provided by both AH and ESP. Further, a particular traffic flow may require IPSec services between hosts and, for that same flow, separate services between security gateways, such as firewalls. In all of these cases, multiple SAs must be employed for the same traffic flow to achieve the desired IPSec services. The term ***security association bundle*** refers to a sequence of SAs through which traffic must be processed to provide a desired set of IPSec services.

Security associations may be combined into bundles in two ways:

- **Transport adjacency:** Refers to applying more than one security protocol to the same IP packet, without invoking tunneling.
- **Iterated tunneling:** Refers to the application of multiple layers of security protocols effected through IP tunneling.

6.6 KEY MANAGEMENT:

The key management portion of IPSec involves the determination and distribution of secret keys. A typical requirement is four keys for communication between two applications: transmit and receive pairs for both AH and ESP.

The IPSec Architecture document mandates support for two types of key management:

- **Manual:** A system administrator manually configures each system with its own keys and with the keys of other communicating systems. This is suitable for small, relatively static environments.
- **Automated:** An automated system enables the on-demand creation of keys for SAs and facilitates the use of keys in a large distributed system.

The default automated key management protocol for IPSec is referred to as ISAKMP/Oakley and consists of the following elements:

1. **Oakley Key Determination Protocol**
2. **Internet Security Association and Key Management Protocol (ISAKMP)**

Oakley Key Determination Protocol:

Oakley is a key exchange protocol based on the Diffie-Hellman algorithm but providing added security. Oakley is generic in that it does not dictate specific formats.

The Diffie-Hellman algorithm has **two** attractive features:

1. Secret keys are created only when needed.
2. The exchange requires no preexisting infrastructure other than an agreement on the global parameters.

However, there are a number of weaknesses to Diffie-Hellman, as pointed out in

3. It does not provide any information about the identities of the parties.
4. It is subject to a man-in-the-middle attack

It is computationally intensive. As a result, it is vulnerable to a clogging attack, in which an opponent requests a high number of keys.

Oakley is designed to retain the advantages of Diffie-Hellman while countering its weaknesses.

Features of Oakley:

The Oakley algorithm is characterized by five important features:

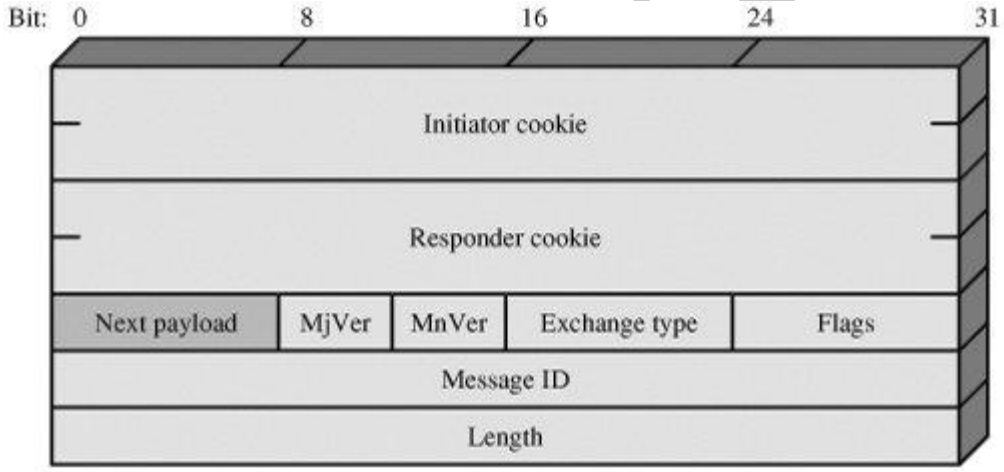
- It employs a mechanism known as cookies to thwart clogging attacks.
- It enables the two parties to negotiate a group; this, in essence, specifies the global parameters of the Diffie-Hellman key exchange.
- It uses nonces to ensure against replay attacks.
- It enables the exchange of Diffie-Hellman public key values.
- It authenticates the Diffie-Hellman exchange to thwart man-in-the-middle attacks.

Internet Security Association and Key Management Protocol (ISAKMP):

ISAKMP provides a framework for Internet key management and provides the specific protocol support, including formats, for negotiation of security attributes.

ISAKMP Header Format:

An ISAKMP message consists of an ISAKMP header followed by one or more payloads. All of this is carried in a transport protocol. The specification dictates that implementations must support the use of UDP for the transport protocol.



(a) ISAKMP header

It consists of the following fields:

1. **Initiator Cookie (64 bits):** Cookie of entity that initiated SA establishment, SA notification, or SA deletion.
2. **Responder Cookie (64 bits):** Cookie of responding entity; null in first message from initiator.
3. **Next Payload (8 bits):** Indicates the type of the first payload in the message
4. **Major Version (4 bits):** Indicates major version of ISAKMP in use.
5. **Minor Version (4 bits):** Indicates minor version in use.
6. **Exchange Type (8 bits):** Indicates the type of exchange.
7. **Flags (8 bits):** Indicates specific options set for this ISAKMP exchange.
8. **Message ID (32 bits):** Unique ID for this message.
9. **Length (32 bits):** Length of total message (header plus all payloads) in octets.

6.7. Intrusion Detection/Prevention System:

Intrusion

A set of actions aimed to compromise the security goals, namely Integrity, confidentiality, or availability, of a computing and networking resource. It is act of gaining unauthorized access to a system so as to cause loss.

Intrusion detection

The process of identifying and responding to intrusion activities

Intrusion prevention

Extension of ID with exercises of access control to protect computers from exploitation

Terminology:

- **True Positives**: These are alerts that something is not right when it is actually not right.
- **True negatives**: these are alerts that something is right when it is actually right.
- **False positives**: these are alerts indicating that something is not right with a packet when actually it is right.
- **False Negatives**: these are alerts that something is right when actually it is wrong.

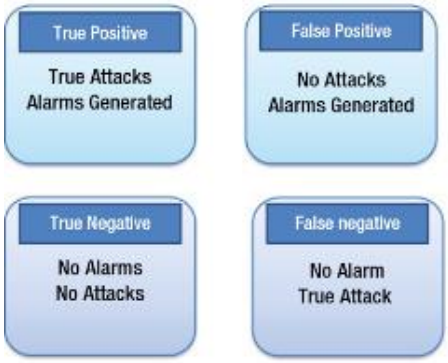
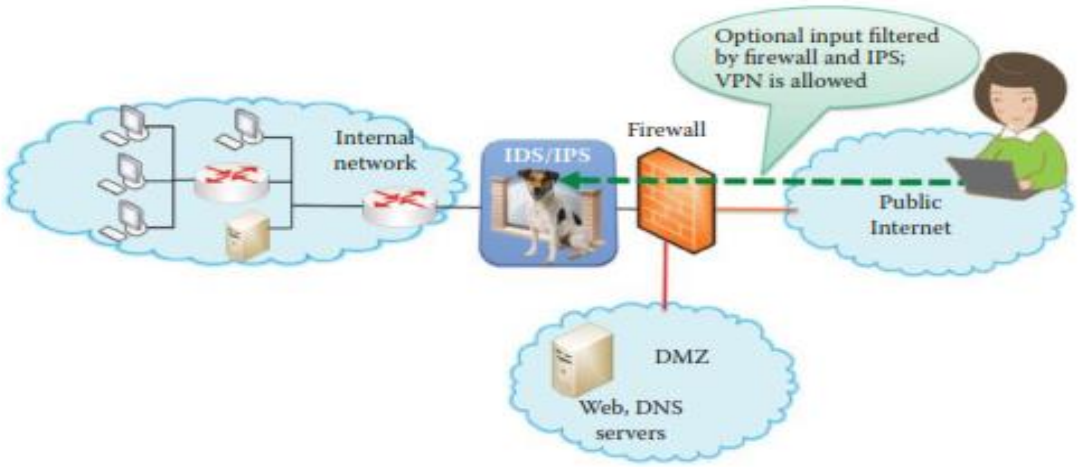


Figure 11-1. Definitions of IDS/IPS Alerts

6.8 OVERVIEW:

An intrusion detection/prevention system (IDS/IPS) is another element in which it employed to provide deep packet inspection at the entrance of important network. Intrusion Detection System/Intrusion Prevention System is positioned behind the firewall, as shown in Figure.



The IDS/IPS provides deep packet inspection for the payload, IDS is based on out-of-band detection of intrusions and their reporting, and IPS is in-band filtering to block intrusions. IDS is performed through

a wiretap, and is clearly an out-of-band operation. In contrast, IPS is performed inline. And by preventing intrusions, IPSs eliminate the need for keeping and reading extensive intrusion-incident logs, which contributes to IDSs' considerable CPU, memory, and I/O overhead.

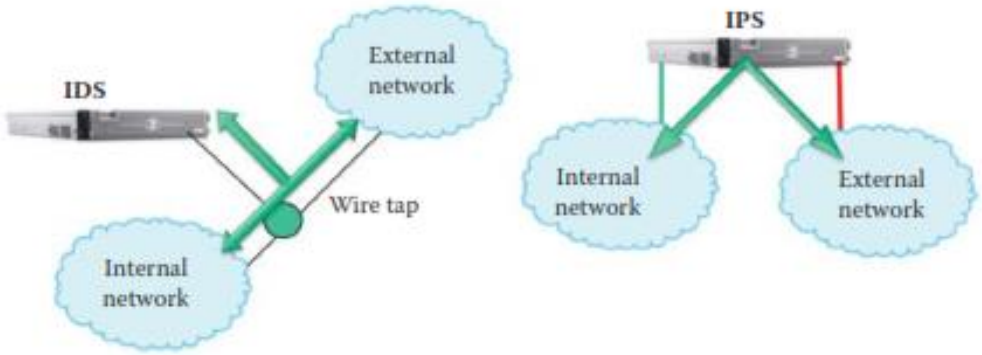


FIGURE 19.2 An illustration of out-of-band IDS vs. in-band IPS.

IDS/IPS BUILDING BLOCKS:

A block diagram that outlines the functions of an IDS/IPS system is shown in Figure. As indicated, the observable activities are preprocessed and forwarded to the detection engine that uses a Signature/Anomaly model. This information is then forwarded to the classification decision engine that uses classification algorithms to provide the alerts or blocking actions

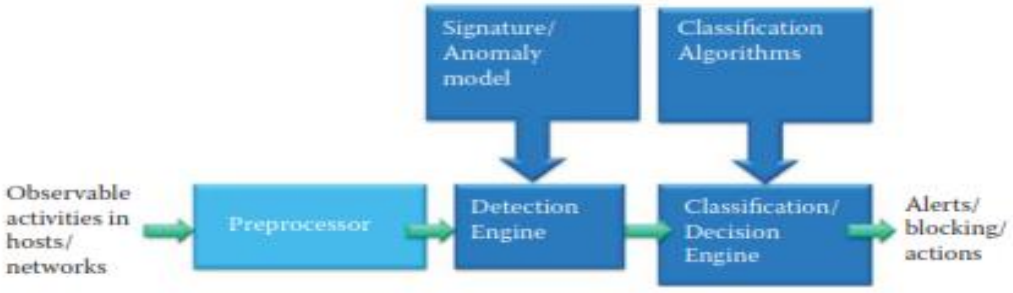


FIGURE 19.3 An IDS/IPS system processes activities and generates alerts and blockings.

HOST-BASED OR NETWORK-BASED IDS/IPS:

IDS/IPS can be either **host-based or network-based**, in which case it is labeled as HIDS/HIPS or NIDS/NIPS, respectively. In the HIDS/HIPS case, the **monitoring and blocking activity is performed on a single host**. HIDS/HIPS has the advantage that it provides better visibility into the behavior of individual applications running on that host. HIDS/HIPS monitoring also includes attacks by genuine users/insiders. These include illegitimate use of root privileges; unauthorized access to resources and data. In the NIDS/NIPS, it is often located behind a router or firewall that provides the guarded entrance to a critical asset. At this location traffic is monitored and packet headers and payloads are examined using the knowledge base in NIDS/NIPS. The advantage of this location is that a single NIDS/NIPS can protect many hosts as well as detect global patterns.

There are **various types of IPS products**.

- **Host-based application** firewalls perform the IPS function independently of the operating system and block the entry of application-level and web based intrusions, much like network firewalls bar entry to unwanted traffic.
- A **network-based IPS** blocks network-level intrusions, such as denial-of-service attacks, and may use anomaly detection to recognize threats based on their behavior.
- Combining network- and host-based IPSs provides the best protection against all types of intrusions.

6.9 THE APPROACHES USED FOR IDS/IPS:

The approaches to **intrusion detection** can generally be classified as either **anomaly/behavior based** or **signature-based**.

- Anomaly-based detectors generate the normal behavior/pattern of the protected system, and deliver an anomaly alarm if the observed behavior at an instant does not conform to expected behavior.
- Anomaly-based IDS/IPS are more likely to generating false positives due to the dynamic nature of networks, applications and exploits.
- According to the type of processing, anomaly detection techniques can be classified into three main categories: **statistical-based, knowledge-based, and machine learning-based**.

STATISTICAL-BASED IDS/IPS:

In the statistical-based IDS/IPS, the behavior of the system is represented from the **captured network traffic activity** and a profile representing its stochastic behavior is created. This profile is based on metrics such as the **traffic rate, the number of packets for each protocol, the rate of connections, the number of different IP addresses**, etc. This method employs the collected profile that relates to the behavior of genuine users and is then used in statistical tests to determine if the behavior under detection is genuine or not. During the anomaly detection process, one corresponding to the currently captured profile is compared with the previously trained statistical profile. As the network events occur, the current profile is determined and an anomaly score estimated by comparison of the two behaviors. The score normally indicates the degree of deviation for a specific event.

Advantages

- First, they do not require prior knowledge about the normal activity of the target system; instead, they have the ability to learn the expected behavior of the system from observations.
- Second, statistical methods can provide accurate notification of malicious activities occurring over long periods of time.

Drawbacks

- setting the values of the thresholds, parameters/metrics that is a difficult task, especially because the balance between false positives and false negatives is affected.
- Not all behaviors can be modeled by using stochastic methods.

KNOWLEDGE-/EXPERT-BASED IDS/IPS:

Knowledge-based IDS/IPS captures the **normal behavior from available information, including expert knowledge, protocol specifications, network traffic instances,** etc. The normal behavior is represented as a **set of rules**. Attributes and classes are identified from the training data or specifications. Then a set of classification rules, parameters or procedures are generated. The rules are used for detecting anomaly behaviors. Specification-based anomaly methods require that the

model is manually constructed by human experts in terms of a set of rules (the specifications) that describing the system behavior. Specification-based techniques have been shown to produce a low rate of false alarms, but are not as effective as other anomaly detection methods in detecting novel attacks, especially when it comes to network probing and denial-of-service attacks. The most significant advantages of knowledge/expert-based detection are the low false alarm rate and the fact that they may detect zero-day and mutated attacks. The main drawback is that the development of high-quality rules is time-consuming and labor-intensive.

MACHINE LEARNING-BASED IDS/IPS:

Machine learning IDS/IPS schemes are based on the establishment of an explicit or implicit model that allows the patterns analyzed to be categorized. Machine learning is different from statistical-based methods because machine learning discovers the characteristics for building a model of behaviors. As more learning is performed, the model will become more accurate. The discovery and learning process is the advantage of machine learning; however, it requires a significant amount of computational resources.

6.10. SIGNATURE BASED IDS:

This mechanism proceeds against known threats. A signature is a known pattern of a threat, such as:

- An e-mail with an attachment containing a malware with an interesting subject.
- A “remote login" by an admin user, which is a clear violation of an organization's policy.

Signature-based detection is the simplest form of detection because it just the traffic with the signature database. If a match found then the alert is generated, if a match is not found then the traffic flows without any problem. In signature-based detection, detection is based on comparing the traffic with the known signatures for possible attacks. They can only detect known threats and hence, are not efficient in detecting unknown threats. To detect an attack, the signature matching has to be precise, otherwise, even if the attack has a small variation from the known threat signature, then the system will not detect. Hence, it is very easy for the attackers to compromise and breach into the trusted network.

Signature database needs to be updated constantly, almost on a daily basis from the anti-virus labs. If the signature is not up to date, chances are that the IDS systems will fail to detect some of the intrusion attacks. The other disadvantage is that they have very little information about previous requests when processing the current ones.

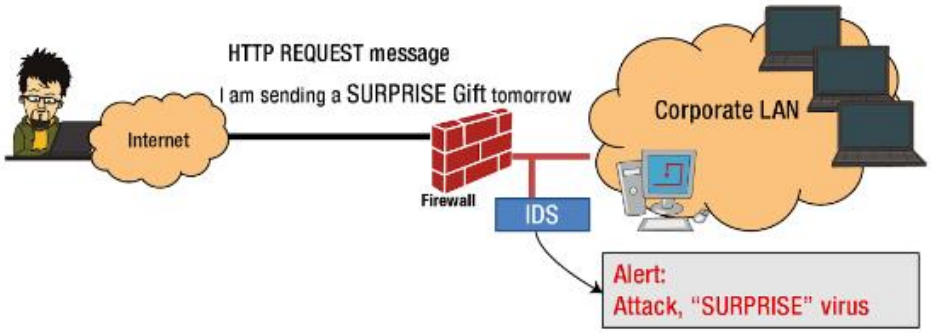


Figure. Signature based detection

Signature based detection can offer very specific detection threats by comparing network traffic with the threat signature database. The detection can be enhanced if the network traffic inside the network can be made to learn specific patterns, thus reducing false positives. Signature detection engines tend

to degrade in performance over a period of time as more and more signatures are added to the database. It takes more time for engine to do a pattern search as the signature database is always growing as more and more definitions are added to it. Hence a robust platform is needed for signature detection considering this growth.

6.11. HOST-BASED IDS:

Host-based Intrusion Detection System refers to the detection intrusion Single system. This is normally software-based deployment where an agent, as shown in Figure, is installed on the local host that monitors and reports the application activity. HIDS monitors the access to the system and its application and sends alerts for any unusual activities. It **constantly monitors event logs, system logs, application logs**, user policy enforcement, rootkit detection, file integrity and other intrusions to the system. **It constantly monitors these logs and creates a baseline.** If any log entries appear, HIDS Checks the data against the baseline and if entries are found outside of this baseline. HIDS triggers an alert, if any unauthorized activity is detected, HIDS can alert the user or block the activity or perform any other decision based the policy that is configured on the system.

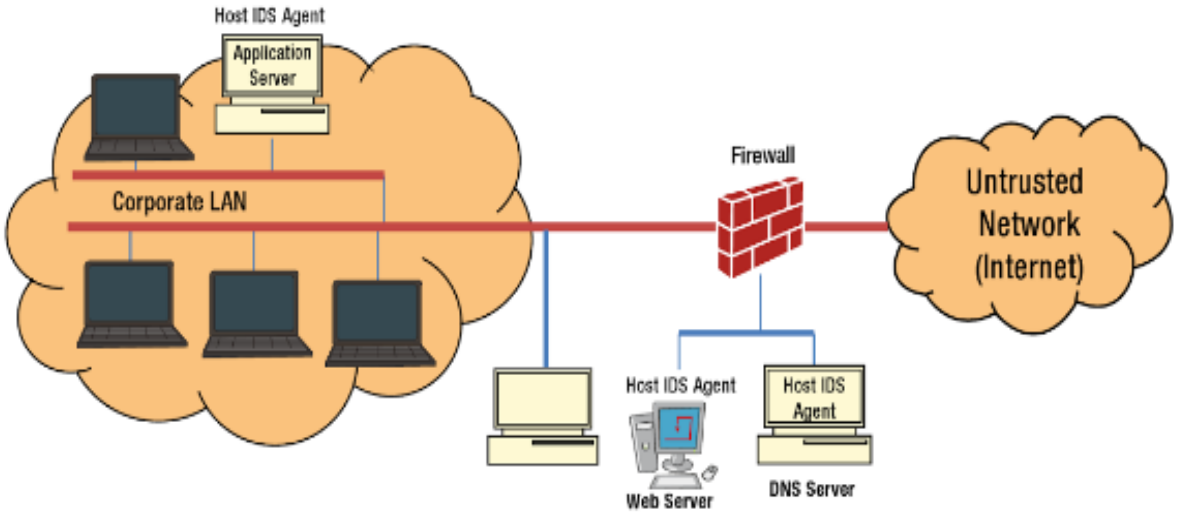


Figure 11-2. Host-Based Intrusion Detection System

Most of the HIDS products have ability to prevent attacks also. However, it is initially deployed in the monitor mode and then there is of the System activity, a baseline is and then HIDS is deployed prevention mode. The functionality HIDS depends the logs generated by the System and the fact that the intruders leave evidence of their activities. Generally, hackers get access to the System and install malicious tools so that future access becomes easier. If these tools change the operating system configurations, or entries of some windows registry, it is logged in the systems/event log, thus triggering an alert by the HIDS system. HIDS is generally installed on servers, or end point devices to protect the system from intrusion. The function of HIDS solely depends on the audit trails generated by the system, If hackers manage to turn off these logs, if you have a HIDS agent running. it may not trigger any alerts. This is the biggest disadvantage of HIIDS.

Advantages of HIDS are:

- System level protection. Protects from attacks directed to the system
- Any unauthorized activity on the system (configuration changes, file changes, registry changes, etc.) are detected and an alert is generated for further action

Disadvantages

- HIDS functionality works only if the Systems generate logs and match against the pre-defined policies. If for some reason, Systems do not generate logs, HIDS may not function properly.
- If hackers bring down the HIDS server, then HIDS is of no use. This is true for any vulnerability Software.

HOST-BASED IDS/IPS:

Many host security products contain integrated host-based IDS/IPS systems (HIDS/HIPS), anti-malware and a firewall. These HIDS/HIPS systems have both advantages and weaknesses. They are capable of protecting mobile hosts from an attack when outside the protected internal network, and they can defend local attacks, such as malware in removable devices. They also protect against attacks from network and encrypted attacks in which the encrypted data stream terminates at the host being protected. They have the capability of detecting anomalies of host software execution, e.g., system call patterns. HIDS/HIPS builds a dynamic database of system objects that can be monitored.

On the negative side, if an attacker takes over a host, the HIDS/HIPS and NAC (Network Access Control) agent software can be compromised and disabled, and the audit logs are modified to hide the malware. In addition, HIDS/HIPS has only a local view of the attack, and host-based anomaly detection has a high false alarm rate.