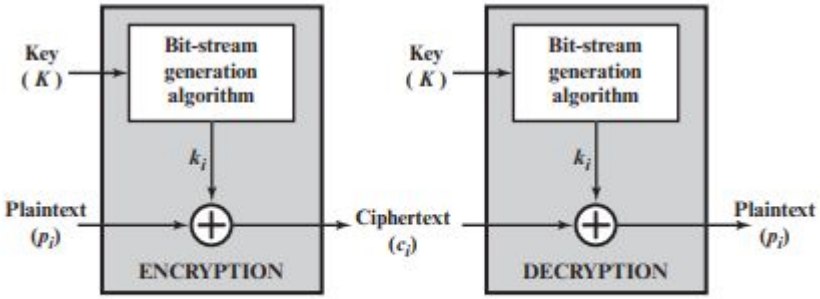
MODULE 2: Cryptography: Key Management, Distribution and User Authentication

-by
Asst Prof Rohini Sawant

STREAM & BLOCK CIPHERS

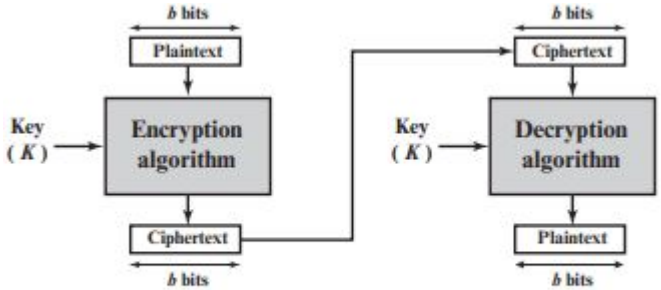
- A stream cipher is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the autokeyed Vigenère cipher and the Vernam cipher.
- the bit-stream generator is a key-controlled algorithm and must produce a bit stream that is cryptographically strong.
- That is, it must be computationally impractical to predict future portions of the bit stream based on previous portions of the bit stream.
- The two users need only share the generating key, and each can produce the keystream.



(a) Stream cipher using algorithmic bit-stream generator

STREAM & BLOCK CIPHERS

- 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.
- Far more effort has gone into analyzing block ciphers.
- In general, they seem applicable to a broader range of applications than stream ciphers.
- The vast majority of network-based symmetric cryptographic applications make use of block ciphers.



(b) Block cipher

Stream Cipher	Block Cipher
Stream cipher operates on smaller Units of Plaintext	Block cipher operates on larger block of data
Faster than block cipher	Slower than Stream Cipher
Stream cipher processes the input element continuously producing output one element at a time	Block cipher processes the input one block of element at a time, producing an output block for each input block
Require less code	Requires more code
Only one time of key used.	Reuse of key is possible
Ex: One time pad	Ex: DES (Data Encryption Standard)
Application: SSL (secure connection on the web)	Application: Database, file encryption.
Stream cipher is more suitable for hardware implementation	Easier to implement in software.

CONFUSION & DIFFUSION

- The terms diffusion and confusion were introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system.
- Shannon's concern was to thwart cryptanalysis based on statistical analysis.
- In what Shannon refers to as a strongly ideal cipher, all statistics of the ciphertext are independent of the particular key used.
- Other than recourse to ideal systems, Shannon suggests two methods for frustrating statistical cryptanalysis: **Diffusion** and **Confusion**.
- In diffusion, the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext.
- This is achieved by having each plaintext digit affect the value of many ciphertext digits; generally, this is equivalent to having each ciphertext digit be affected by many plaintext digits.
- Example of Diffusion is Transposition Cipher.

CONFUSION & DIFFUSION

- On the other hand, confusion seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart attempts to discover the key.
- Thus, even if the attacker can get some handle on the statistics of the ciphertext, the way in which the key was used to produce that ciphertext is so complex as to make it difficult to deduce the key.
- The relations between CT and PT is obscured.
- Given a CT, no information about PT, Key, Encryption algorithm is known.
- This is achieved by the use of a complex substitution algorithm.

THE FEISTEL CIPHER

- Feistel proposed that we can approximate the ideal block cipher by utilizing the concept of a product cipher, which 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.
- The essence of the approach is to develop a block cipher with a key length of k bits and a block length of n bits, allowing a total of 2^k possible transformations, rather than the 2^n ! transformations available with the ideal block cipher.
- 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.

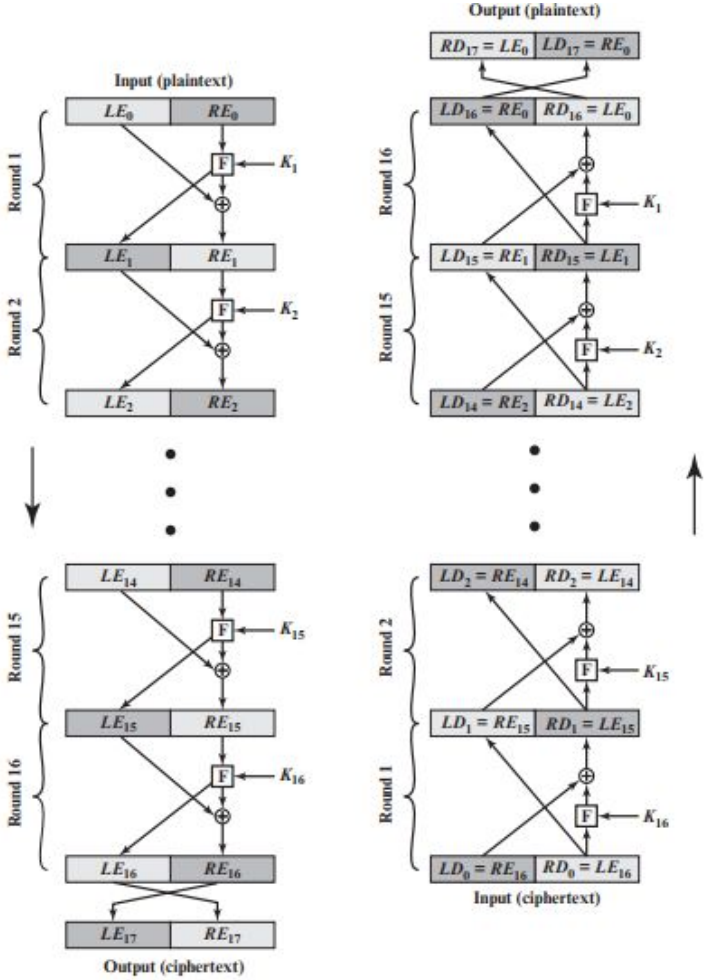


Figure 4.3 Feistel Encryption and Decryption (16 rounds)

THE FEISTEL CIPHER

- The left-hand side of Figure 4.3 depicts the encryption structure proposed by Feistel. The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key K.
- The plaintext block is divided into two halves, LE0 and RE0. The two halves of the data pass through n rounds of processing and then combine to produce the ciphertext block.
- In general, the subkeys Ki are different from K and from each other.
- In Figure 4.3, 16 rounds are used, although any number of rounds could be implemented.

THE FEISTEL CIPHER

- 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.
- The round function has the same general structure for each round but is parameterized by the round subkey Ki .
- Following this substitution, a permutation is performed that consists of the interchange of the two halves of the data.
- This structure is a particular form of the substitution-permutation network (SPN) proposed by Shannon.

THE FEISTEL STRUCTURE 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.
- 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.

THE FEISTEL STRUCTURE DESIGN FEATURES

- Round function F: Again, greater complexity generally means greater resistance to cryptanalysis.
- Fast software encryption/decryption: In many cases, encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation. Accordingly, the speed of execution of the algorithm becomes a concern
- Ease of analysis: Although we would like to make our algorithm as difficult as possible to cryptanalyze, there is great benefit in making the algorithm easy to analyze.
- **The process of decryption with a Feistel cipher is essentially the same as the encryption process. The rule is as follows: Use the ciphertext as input to the algorithm, but use the subkeys Ki in reverse order. That is, use Kn in the first round, Kn-1 in the second round, and so on, until K1 is used in the last round.**

DATA ENCRYPTION STANDARD

- The Data Encryption Standard (DES) was the most widely used encryption scheme. DES was issued in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), as Federal Information Processing Standard.
- It is a Symmetric Block Cipher.
- The algorithm is also referred to as the Data Encryption Algorithm (DEA).
- It was redundant after the invasion of Advanced Encryption Standard (AES) in 2001.
- For DES, data are encrypted in 64-bit blocks using a 56-bit key. The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used to reverse the encryption.

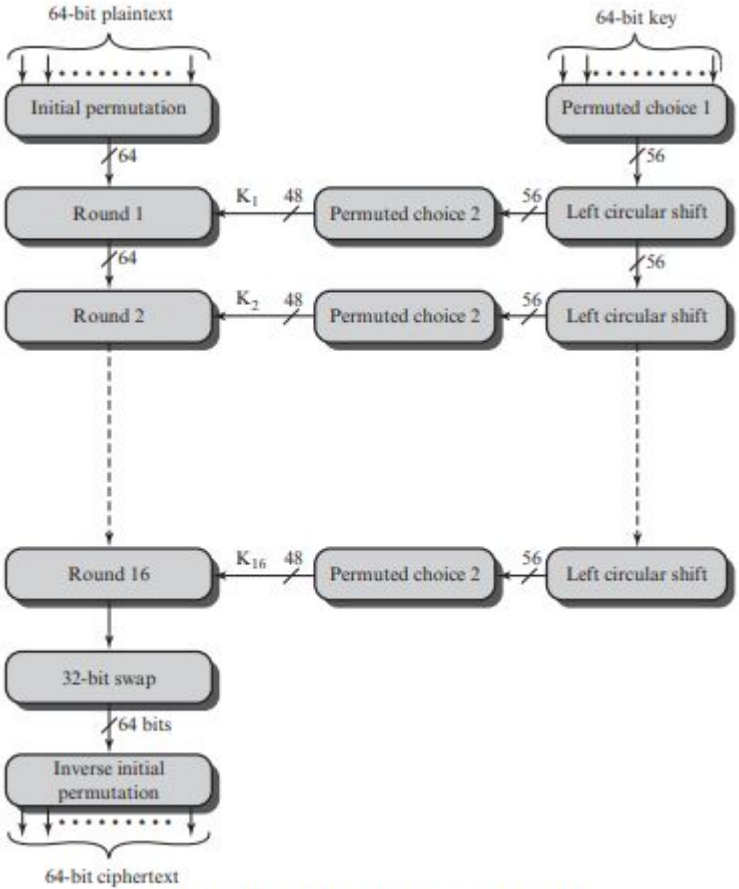


Figure 4.5 General Depiction of DES Encryption Algorithm

DATA ENCRYPTION STANDARD

- The overall scheme for DES encryption is illustrated in Figure 4.5. As with any encryption scheme, there are two inputs to the encryption function: the plaintext to be encrypted and the key.
- In this case, the plaintext must be 64 bits in length and the key is 56 bits in length.8 Looking at the left-hand side of the figure, we can see that the processing of the plaintext proceeds in three phases.
- First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input.
- 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.
- As with any Feistel cipher, decryption uses the same algorithm as encryption, except that the application of the subkeys is reversed. Additionally, the initial and final permutations are reversed

DATA ENCRYPTION STANDARD

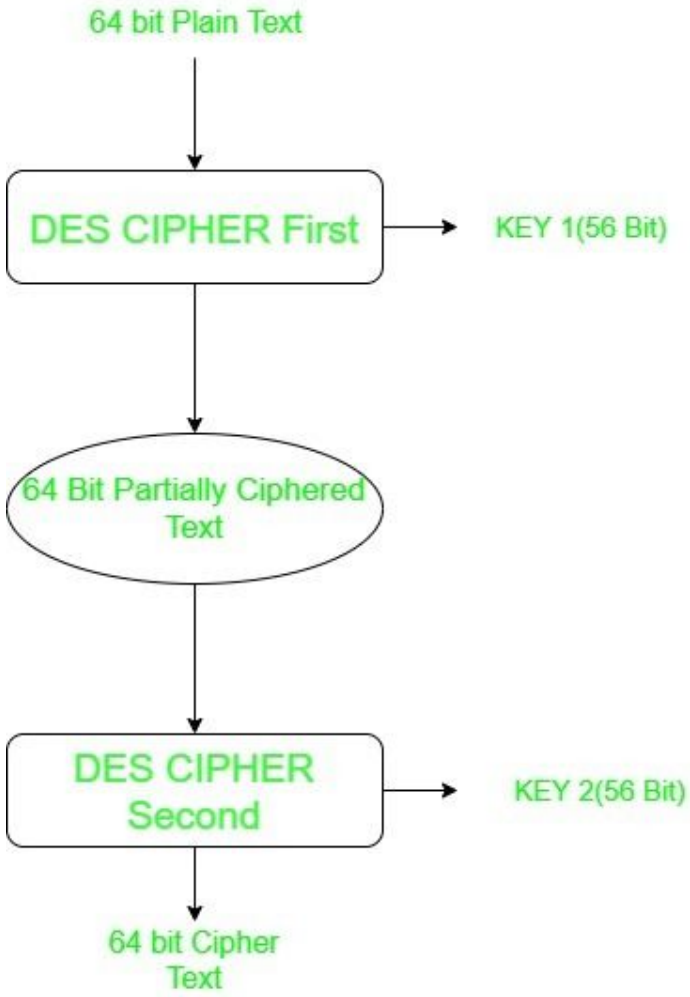
- Finally, the preoutput is passed through a permutation [IP-1] that is the inverse of the initial permutation function, to produce the 64-bit ciphertext. With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher, as shown in Figure 4.3
- The right-hand portion of Figure 4.5 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.

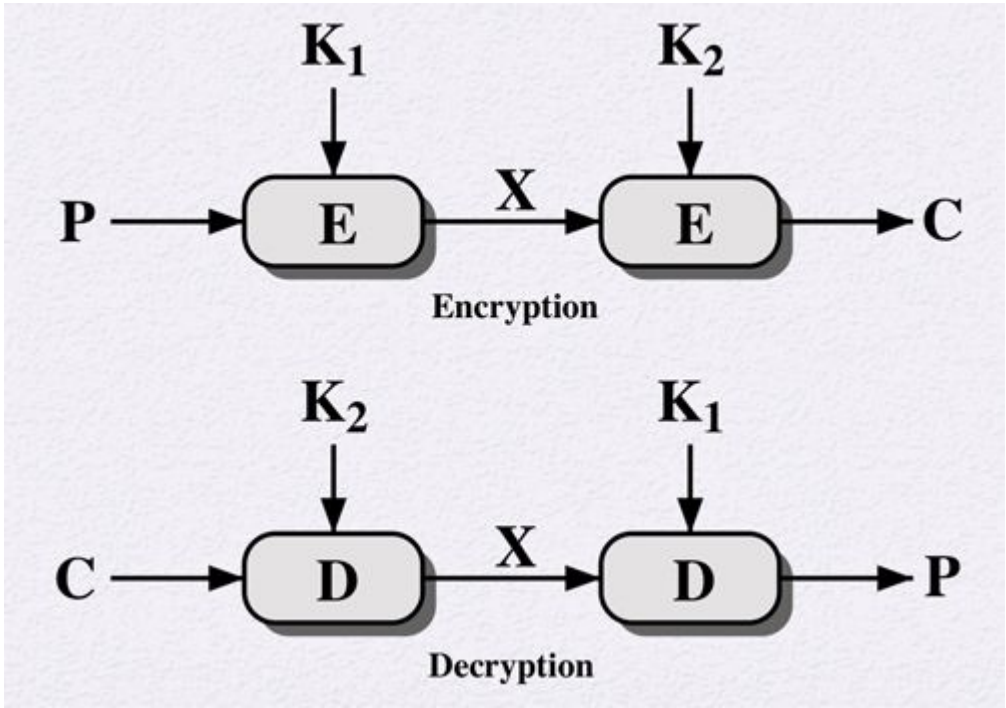
WEAKNESS IN DES

- DES has been proven to be susceptible to Crpytanalysis.
- 56-bit keys have a keyspace of $2^{56.}$
- As we know the DES uses 56 bit key to encrypt any plain text which can be easily be cracked by using modern technologies.
- To prevent this from happening double DES and triple DES were introduced which are much more secure than the original DES because it uses 112 and 168 bit keys respectively.
- They offer much more security than DES.

DOUBLE DES

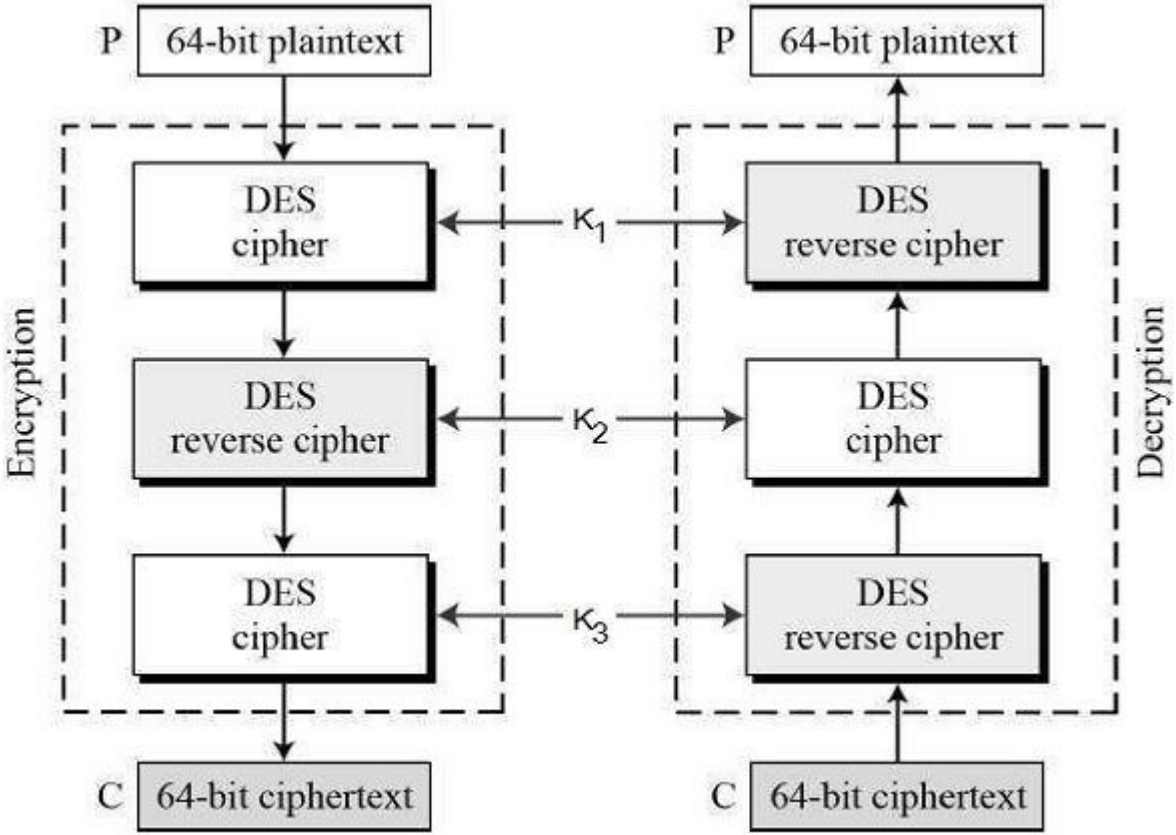
- Double DES is a encryption technique which uses two instance of DES on same plain text.
- In both instances it uses different keys to encrypt the plain text. Both keys are required at the time of decryption.
- The 64 bit plain text goes into first DES instance which then converted into a 64 bit middle text using the first key and then it goes to second DES instance which gives 64 bit cipher text by using second key.
- Meet-in-the middle attack which can be used to break through double DES.


64 bit Plain Text
DES CIPHER First
KEY 1(56 Bit)
64 Bit Partially Ciphered Text
DES CIPHER Second
KEY 2(56 Bit)
64 bit Cipher Text


K1
K2
P
E
X
E
C
Encryption
K2
K1
C
D
X
D
P
Decryption

TRIPLE DES

- Triple DES is a encryption technique which uses three instance of DES on same plain text. It uses there different types of key choosing technique in first all used keys are different and in second two keys are same and one is different and in third all keys are same.
- Before using 3TDES, user first generate and distribute a 3TDES key K, which consists of three different DES keys K1, K2 and K3.
- This means that the actual 3TDES key has length 3×56 = 168 bits.
- Triple DES systems are significantly more secure than single DES, but these are clearly a much slower process than encryption using single DES.


P
64-bit plaintext
DES
cipher
DES
reverse cipher
DES
cipher
C
64-bit ciphertext
Encryption
K1
K2
K3
P
64-bit plaintext
DES
reverse cipher
DES
cipher
DES
reverse cipher
C
64-bit ciphertext
Decryption

BLOCK CIPHER MODES OF OPERATION

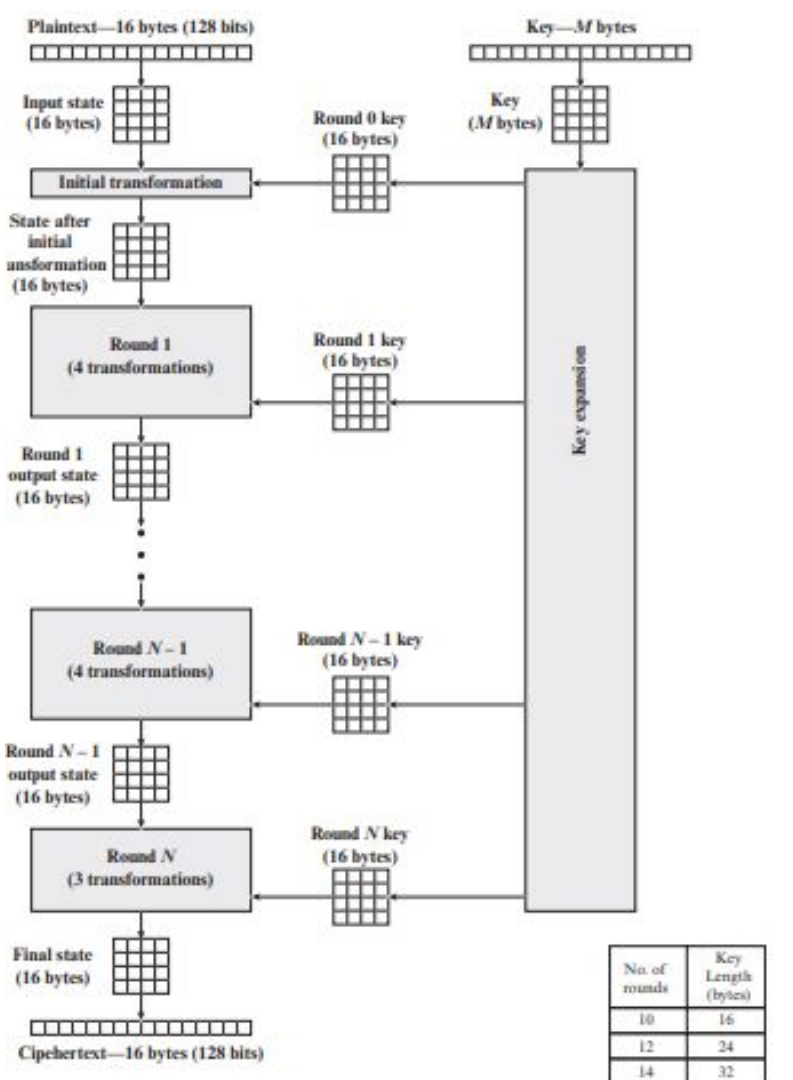
- Encryption algorithms are divided into two categories based on the input type, as a block cipher and stream cipher.
- **Block cipher** is an encryption algorithm that takes a fixed size of input say *b* bits and produces a ciphertext of *b* bits again.
- If the input is larger than *b* bits it can be divided further.
- For different applications and uses, there are several modes of operations for a block cipher.

Table 7.1 Block Cipher Modes of Operation

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of plaintext bits is encoded independently using the same key.	• Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next block of plaintext and the preceding block of ciphertext.	• General-purpose block-oriented transmission • Authentication
Cipher Feedback (CFB)	Input is processed s bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	• General-purpose stream-oriented transmission • Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.	• Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	• General-purpose block-oriented transmission • Useful for high-speed requirements

ADVANCED ENCRYPTION STANDARD

- The Advanced Encryption Standard (AES) was published by the National Institute of Standards and Technology (NIST) in 2001.
- AES is a symmetric block cipher that is intended to replace DES.
- It can work with three key sizes-128,192,256 bits,
- AES is considered highly secure due to its long key sizes and is still used in industries.
- Based on the key length i.e 16, 24, or 32 bytes (128, 192, or 256 bits),the algorithm is referred to as AES-128, AES-192, or AES-256.



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

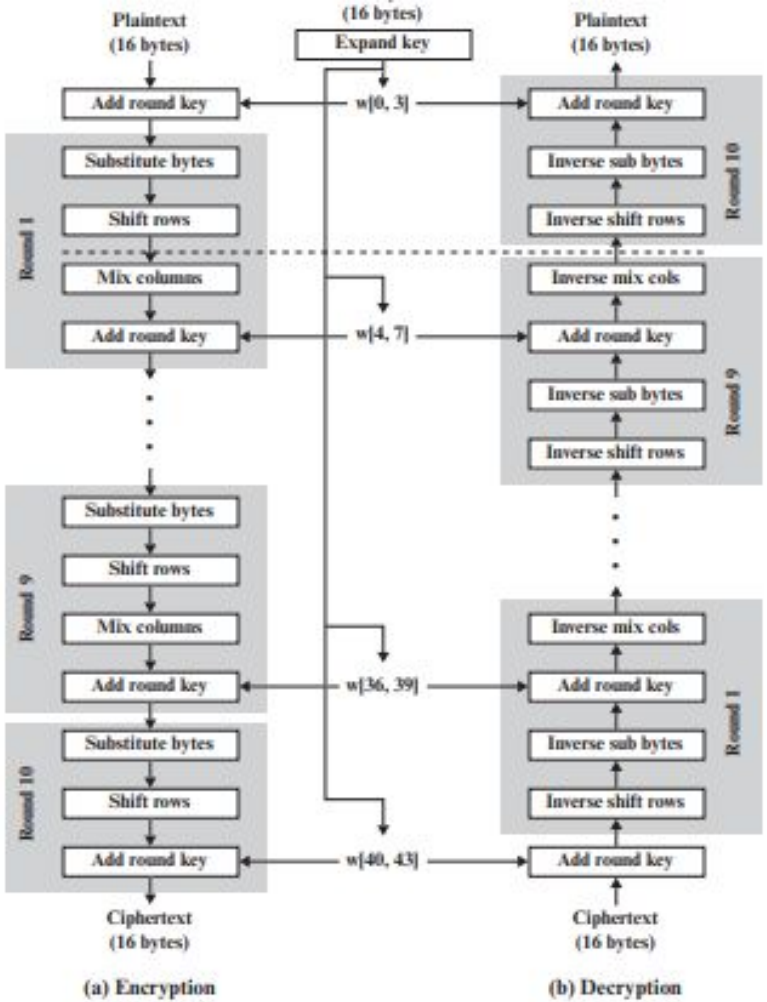


Figure 6.3 AES Encryption and Decryption

ADVANCED ENCRYPTION STANDARD

- Figure 6.1 shows the overall structure of the AES encryption process. The cipher takes a plaintext block size of 128 bits, or 16 bytes.
- 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 6.2a.
- Similarly, the key is depicted as a square matrix of bytes. This key is then expanded into an array of key schedule words. Figure 6.2b 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.

ADVANCED ENCRYPTION STANDARD

- 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.
- 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.

Table 6.1 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

ADVANCED ENCRYPTION STANDARD

- 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 over GF(28).
- AddRoundKey: A simple bitwise XOR of the current block with a portion of the expanded key.

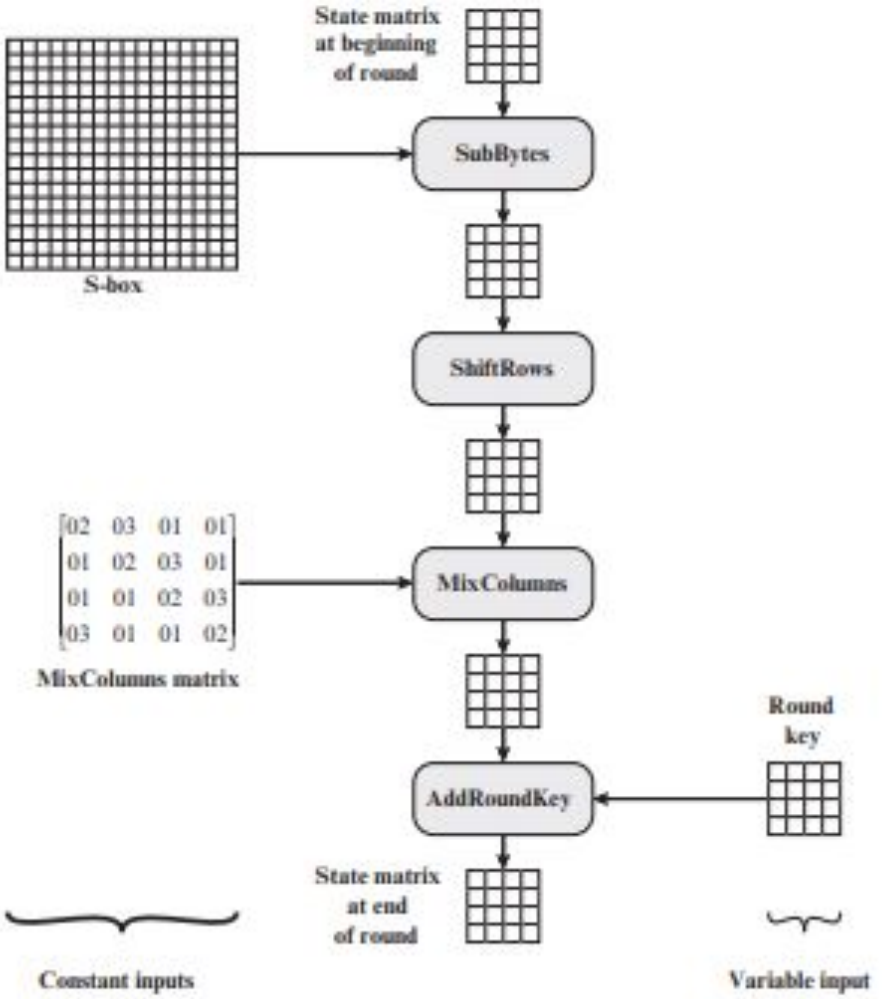
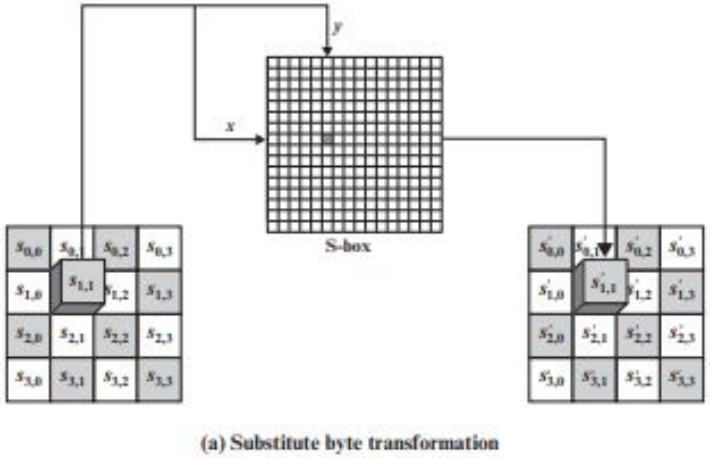


Figure 6.8 Inputs for Single AES Round

SUBSTITUTE BYTES TRANSFORMATION

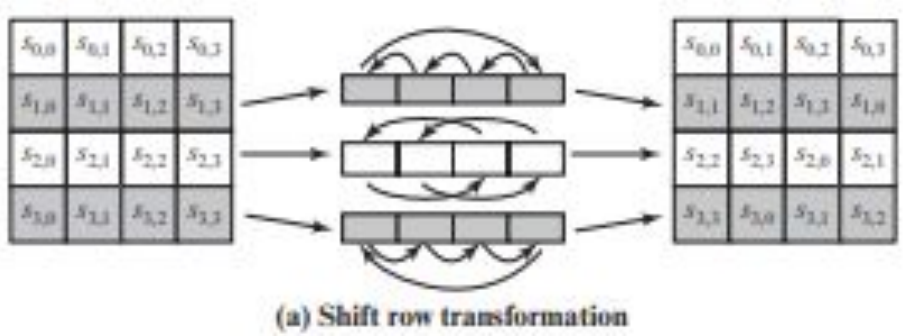
- AES defines a 16 * 16 matrix of byte values, called an S-box (Table 6.2a), that contains a permutation of all possible 256 8-bit values.
- Each individual byte of State is mapped into a new byte in the following way: The leftmost 4 bits of the byte are used as a row value and the rightmost 4 bits are 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 value {95} references row 9, column 5 of the S-box, which contains the value {2A}. Accordingly, the value {95} is mapped into the value {2A}.



(a) Substitute byte transformation

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. For the fourth row, a 3-byte circular left shift is performed. The following is an example of ShiftRows



(a) Shift row transformation

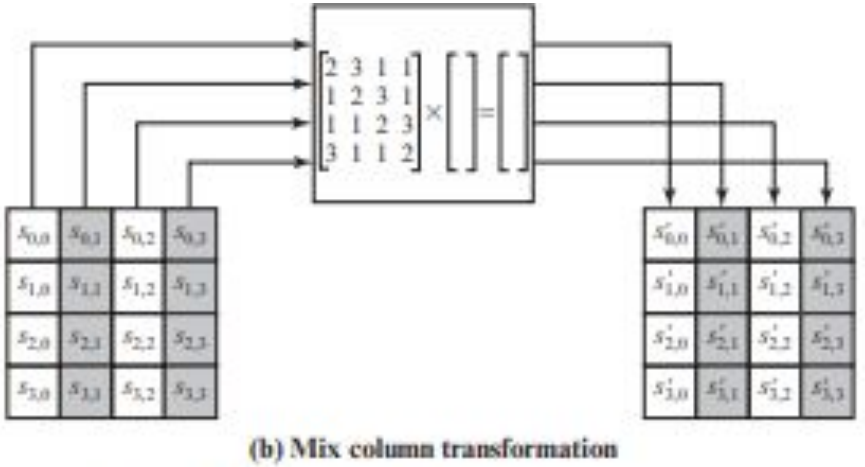
87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

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

MIX COLUMN TRANSFORMATION

Each byte of a column is mapped into a new value that is a function of all four bytes in that column. The transformation can be defined by the following matrix multiplication on State



(b) Mix column transformation

Figure 6.7 AES Row and Column Operations

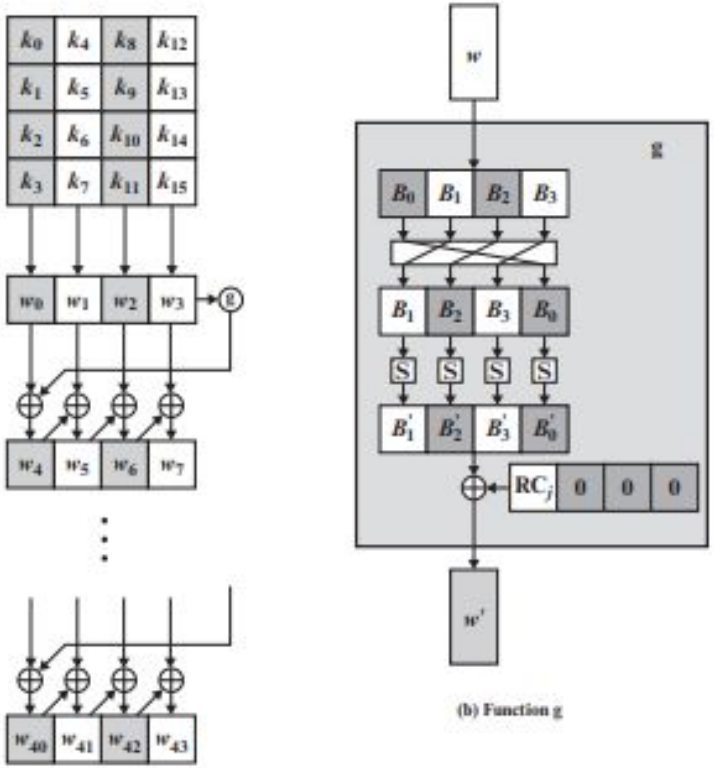
In AddRoundKey, the 128 bits of State are bitwise XORed with the 128 bits of the round key.

AES KEY EXPANSION

- The AES key expansion algorithm takes as input a four-word (16-byte) key and produces a linear array of 44 words (176 bytes). This is sufficient to provide a fourword round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher.
- 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].
- SubWord performs a byte substitution on each byte of its input word, using the S-box (Table 6.2a).
- The result of steps 1 and 2 is XORed with a round constant, Rcon[j].
- The round constant is a word in which the three rightmost bytes are always 0. [Rcon[j] = (RC[j], 0, 0, 0)]

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

AES KEY EXPANSION



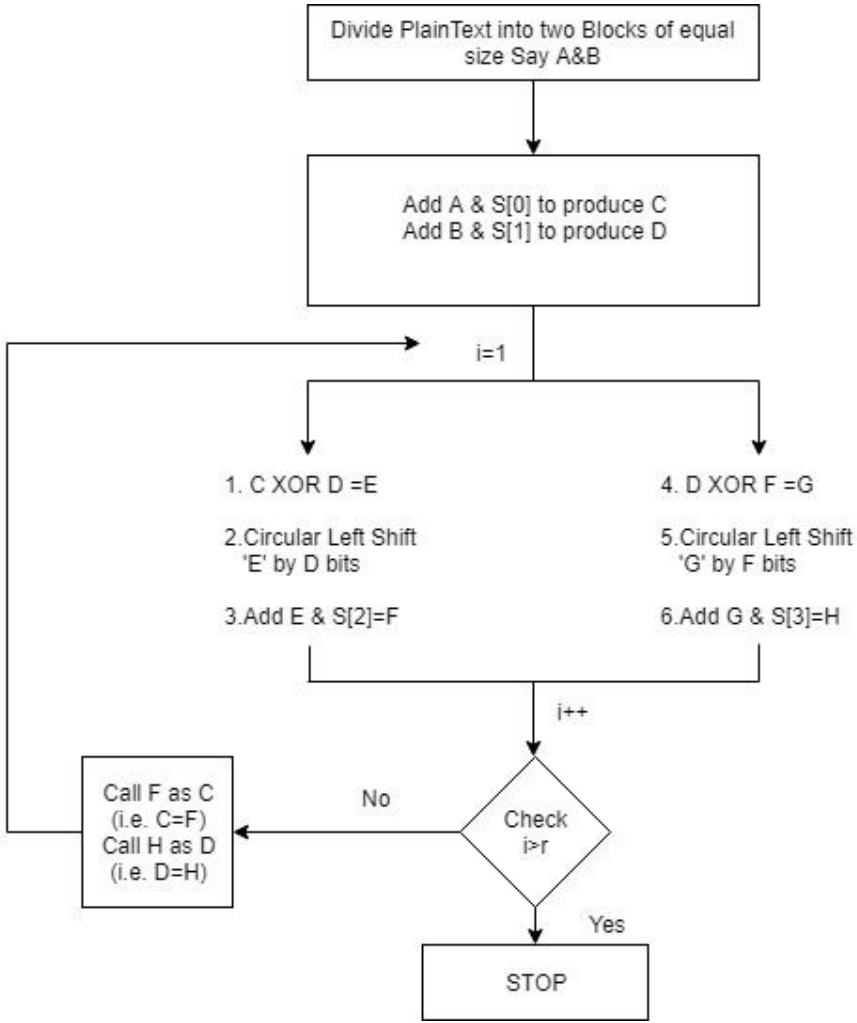
(a) Overall algorithm

(b) Function g

Figure 6.9 AES Key Expansion

RC5 (RIVEST CIPHER 5)

- RC5 is a **Block Cipher** with a variety of parameters: block size, key size, and number of rounds.
- It was invented by Ron Rivest and analyzed by RSA Laboratories.
- There are three operations: XOR, addition, and rotations.
- RC5 has a variable-length block.
- Once w, r, k (word size, number of rounds, number of keys) are finalized then they remain same for all the rounds.
- Plain text can be 32 bits, 64 bits or 128 bits
- Number of rounds can be between 0-255
- Key size can be between 0 to 255 bytes.
- The main feature of RC5 is that it is quite fast as it only uses primitive computer operations (addition, XOR,shift).
- Another important feature of RC5 is that it requires less memory for execution and is therefore suitable for desktop computers, smart cards and other devices that have small memory capacity.


Divide PlainText into two Blocks of equal size Say A&B
Add A & S[0] to produce C
Add B & S[1] to produce D
i=1
1. C XOR D =E
2.Circular Left Shift 'E' by D bits
3.Add E & S[2]=F
4. D XOR F =G
5.Circular Left Shift 'G' by F bits
6.Add G & S[3]=H
i++
Check i>r
No
Call F as C (i.e. C=F)
Call H as D (i.e. D=H)
Yes
STOP

- We initialize the counter to 1 and perform some permutation and combination using addition and XOR

The algorithm works into two phases:

a. First it starts with phase one

b. Output of phase one become input of phase two

- We divide the plaintext block into two equal parts A and B
- Then they are XOR with two subkeys S{0} and S{1}
- C=A+S[0] AND D=B+S[1]
- for i = 1 to r do:
- 1. C ⊕ D = E
- 2. perform circular left shift on E by D bits
- 3. add E and S[2 * i] and store the result in F which is input for step 4
- 4. D ⊕ F = G
- 5. perform circular left shift on G by F bits
- 6. add G and S[2 * i + 1] and store the result in H
- 7. If i< r
- Call F as C and H as D and repeat the steps from 1 to 7
- else stop
- Once both the phases are completed the counter is incriminated and we check if it is greater than the number of rounds, if yes then the algorithm terminals and if no then the algorithm iterates.

Decryption: Decryption is a straightforward reversal of the encryption process

PUBLIC KEY CRYPTOGRAPHY

Asymmetric algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristic.

■ It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key. In addition, some algorithms, such as RSA, also exhibit the following characteristic.

■ Either of the two related keys can be used for encryption, with the other used for decryption.

PUBLIC KEY CRYPTOGRAPHY

A public-key encryption scheme has six ingredients (Figure 9.1a; compare with Figure 3.1).

■ Plaintext: This is the readable message or data that is fed into the algorithm as input.

■ Encryption algorithm: The encryption algorithm performs various transformations on the plaintext.

■ Public and private keys: 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.

■ Ciphertext: This is the encrypted message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.

■ Decryption algorithm: This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

PUBLIC KEY CRYPTOGRAPHY

The essential steps are the following.

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.

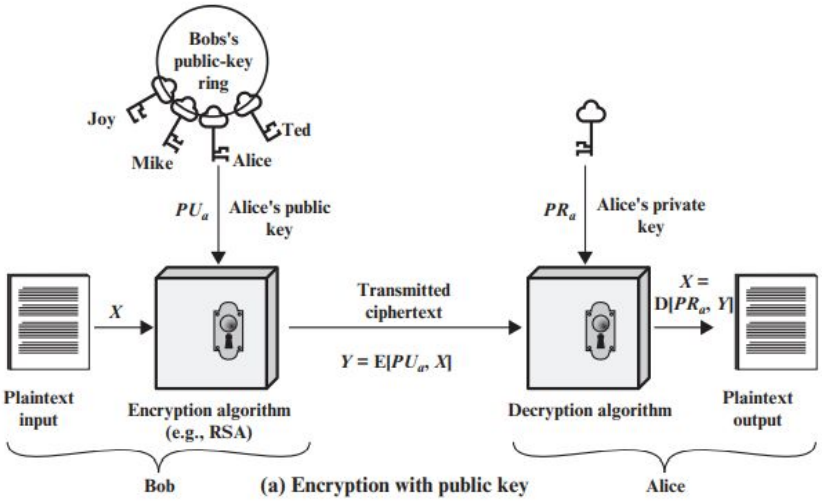
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As Figure 9.1a suggests, each user maintains a collection of public keys obtained from others.

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.

With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as a user's private key remains protected and secret, incoming communication is secure

To discriminate between the two, we refer to the key used in **symmetric encryption as a secret key**. The two keys used for **asymmetric encryption are referred to as the public key and the private key**. 2 Invariably, the private key is kept secret, but it is referred to as a private key rather than a secret key to avoid confusion with symmetric encryption



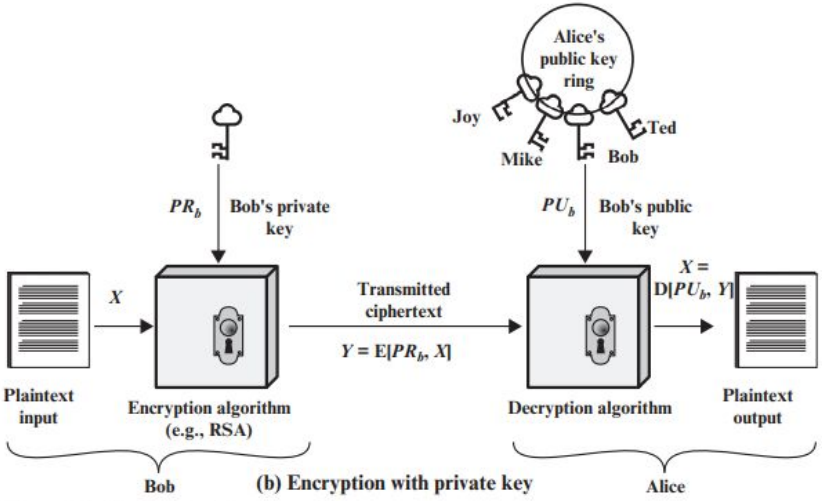


Figure 9.1 Public-Key Cryptography

CONVENTIONAL and PUBLIC KEY CRYPTOGRAPHY

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if the key is kept secret. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and a related algorithm for decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if one of the keys is kept secret. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

APPLICATIONS OF PUBLIC KEY CRYPTOGRAPHY

In broad terms, we can classify the use of public-key cryptosystems into three categories

■ Encryption/decryption: The sender encrypts a message with the recipient's public key, and the recipient decrypts the message with the recipient's private key.

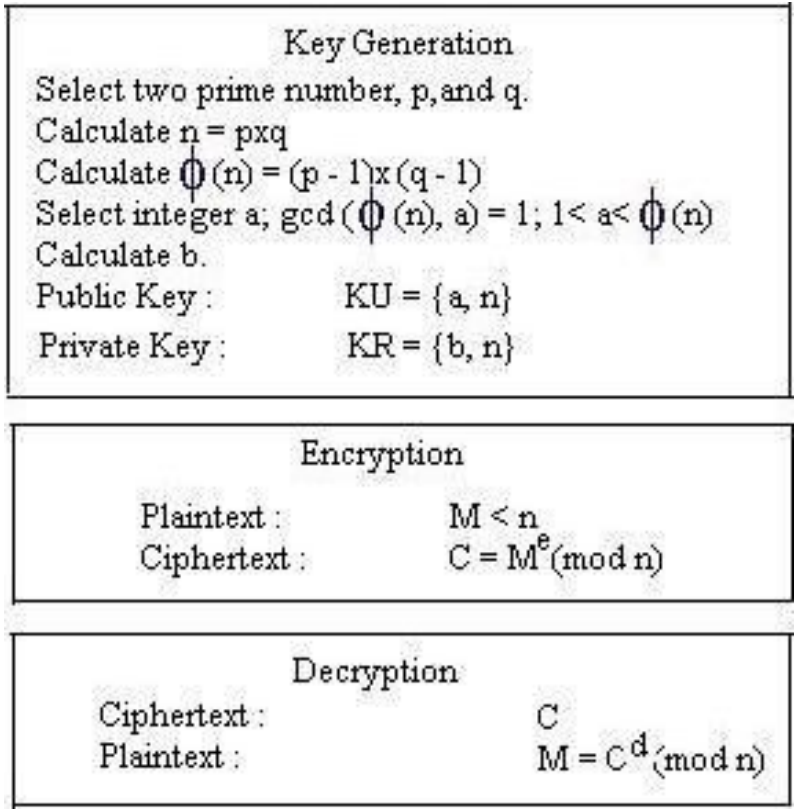
■ Digital signature: The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.

■ Key exchange: Two sides cooperate to exchange a session key, which is a secret key for symmetric encryption generated for use for a particular transaction (or session) and valid for a short period of time. Several different approaches are possible, involving the private key(s) of one or both parties;

RSA Algorithm

- One of the first successful responses to the challenge was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978.
- The Rivest-Shamir-Adleman (RSA) scheme has since that time reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption.
- The RSA scheme is a cipher in which the plaintext and 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 21024.
- RSA makes use of an expression with exponentials.

RSA Algorithm



b= a x b = 1 mod $\Phi(n)$

RSA Algorithm

1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 \times 11 = 187$.
3. Calculate $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.
4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$; we choose $e = 7$.
5. Determine d such that $de \equiv 1 \pmod{160}$ and $d < 160$. The correct value is $d = 23$, because $23 \times 7 = 161 = (1 \times 160) + 1$; d can be calculated using the extended Euclid's algorithm (Chapter 2).

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

SECURITY OF RSA

Five possible approaches to attacking the RSA algorithm are

- 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
- Hardware fault-based attack: This involves inducing hardware faults in the processor that is generating digital signatures.

SECURITY OF RSA

- The defense against the brute-force approach is the same for RSA as for other cryptosystems, namely, to use a large key space. Thus, the larger the number of bits in d, the better. However, because the calculations involved, both in key generation and in encryption/decryption, are complex, the larger the size of the key, the slower the system will run.
- THE FACTORING PROBLEM We can identify three approaches to attacking RSA mathematically.

1. Factor n into its two prime factors. This enables calculation of f(n) = (p - 1) * (q - 1), which in turn enables determination of d K e-1 (mod f(n)).

2. Determine f(n) directly, without first determining p and q. Again, this enables determination of d K e-1 (mod f(n)).

3. Determine d directly, without first determining f(n).

SECURITY OF RSA

- A timing attack is somewhat analogous to a burglar guessing the combination of a safe by observing how long it takes for someone to turn the dial from number to number.
- Countermeasures:

■ Constant exponentiation time: Ensure that all exponentiations take the same amount of time before returning a result. This is a simple fix but does degrade performance.

■ Random delay: Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack. Kocher points out that if defenders don't add enough noise, attackers could still succeed by collecting additional measurements to compensate for the random delays.

■ Blinding: Multiply the ciphertext by a random number before performing exponentiation. This process prevents the attacker from knowing what ciphertext bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack.

SECURITY OF RSA

- The basic RSA algorithm is vulnerable to a chosen ciphertext attack (CCA). CCA is defined as an attack in which the adversary chooses a number of ciphertexts and is then given the corresponding plaintexts, decrypted with the target's private key.
- Thus, the adversary could select a plaintext, encrypt it with the target's public key, and then be able to get the plaintext back by having it decrypted with the private key.
- A solution can be optimal asymmetric encryption padding (OAEP)

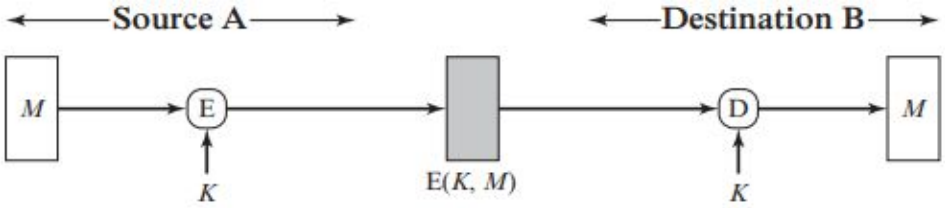
MESSAGE AUTHENTICATION FUNCTIONS

These may be grouped into three classes:

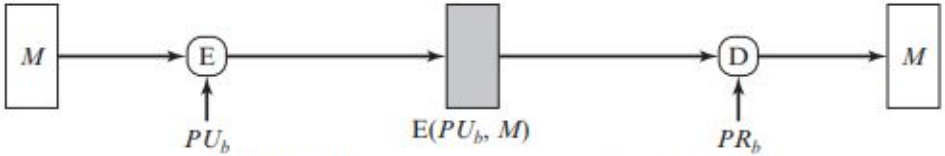
■ Hash function: A function that maps a message of any length into a fixed-length hash value, which serves as the authenticator

■ Message encryption: The ciphertext of the entire message serves as its authenticator

■ Message authentication code (MAC): A function of the message and a secret key that produces a fixed-length value that serves as the authenticator

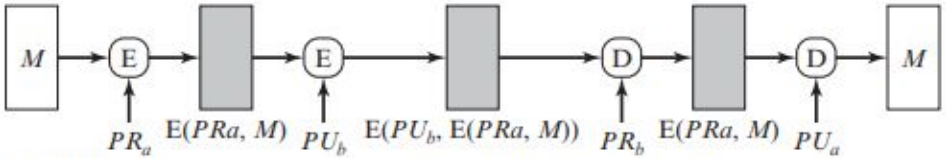


(a) Symmetric encryption: confidentiality and authentication



(b) Public-key encryption: confidentiality

(c) Public-key encryption: authentication and signature



(d) Public-key encryption: confidentiality, authentication, and signature

Figure 12.1 Basic Uses of Message Encryption

Message Authentication Code

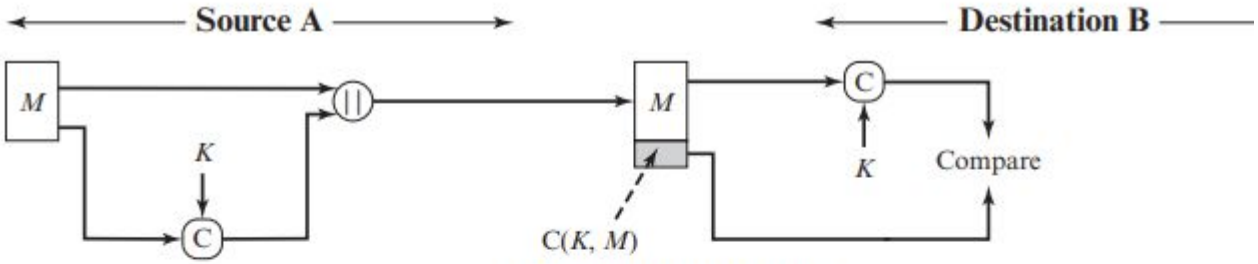
An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a **cryptographic checksum** or MAC, that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key K. When A has a message to send to B, it calculates the MAC as a function of the message and the key:

$$\text{MAC} = C(K, M)$$

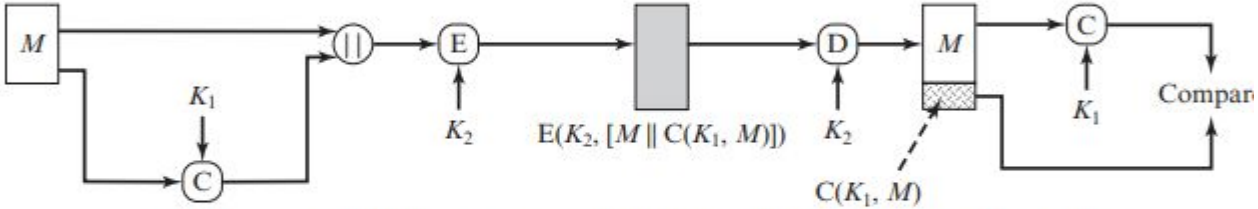
where

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

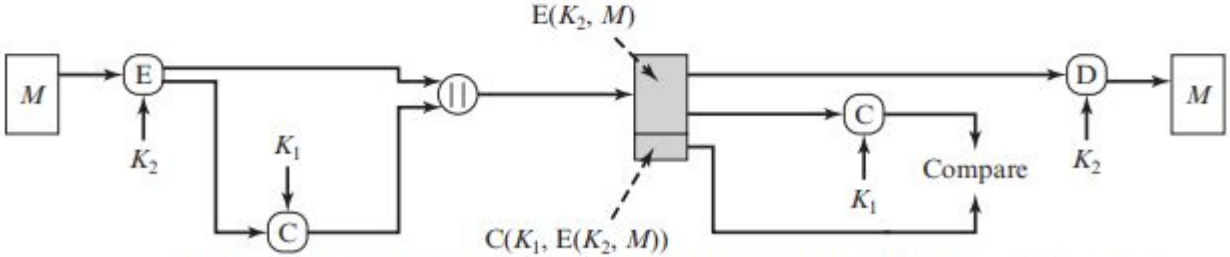
The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated MAC



(a) Message authentication



(b) Message authentication and confidentiality; authentication tied to plaintext

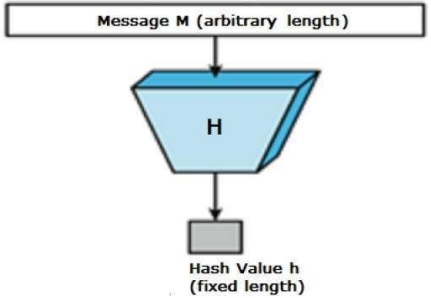


(c) Message authentication and confidentiality; authentication tied to ciphertext

Figure 12.4 Basic Uses of Message Authentication code (MAC)

HASHING

- Hash functions are extremely useful and appear in almost all information security applications.
- A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length but output is always of fixed length.
- A cryptographic hash function is an algorithm that takes an arbitrary amount of data input—a credential—and produces a fixed-size output of enciphered text called a hash value, or just “hash.”
- Values returned by a hash function are called message digest or simply hash values.
- A hash function is a versatile one-way cryptographic algorithm that maps an input of any size to a unique output of a fixed length of bits.
- When you hash data, the resulting digest is typically smaller than the input that it started with.
- With hashing, it doesn’t matter if you have a one-sentence message or an entire book — the result will still be a fixed-length chunk of bits



Process of Hashing

1. Create Information
2. Calculate the Hash Value
3. Encrypt the message
4. Send the Encrypted message and the Hash Value
5. Receive the Encrypted message and the Hash Value
6. Decrypt the message
7. Calculate its hash value at the receiving end
8. Compare the hashes
9. If matched, Process the information, else reject.

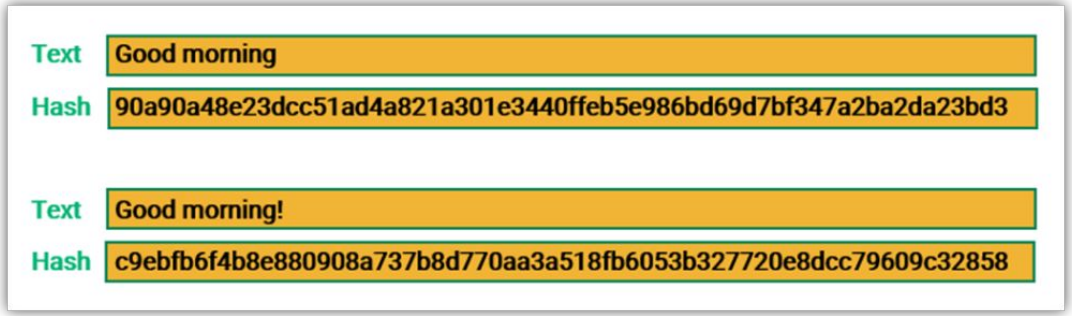
Characteristics of Hashing

Characteristics of Hash:

- One way only
- Any length Input, Fixed Length Output
- No Secrecy Involved
- Avalanche Effect
- Collisions are Possible.

SHA 256

- SHA-256, which stands for secure hash algorithm 256, is a cryptographic hashing algorithm (or function) that's used for message, file, and data integrity verification.
- Published in 2001, it was a joint effort between the NSA and NIST to introduce a successor to the SHA 1 family, which was slowly losing strength against brute force attacks.
- It's part of the SHA-2 family of hash functions and uses a 256-bit key to take a piece of data and convert it into a new, unrecognizable data string of a fixed length.
- This string of random characters and numbers, called a hash value, is also 256 bits in size.



SHA 256

Some of the standout features of the SHA algorithm are as follows:

- Message Length: The length of the cleartext should be less than 264 bits. The size needs to be in the comparison area to keep the digest as random as possible.
- Digest Length: The length of the hash digest should be 256 bits in SHA 256 algorithm, 512 bits in SHA-512, and so on. Bigger digests usually suggest significantly more calculations at the cost of speed and space.
- Irreversible: By design, all hash functions such as the SHA 256 are irreversible. You should neither get a plaintext when you have the digest beforehand nor should the digest provide its original value when you pass it through the hash function again.

SHA 256

SHA 256 follows the steps given below:

1. First, data is converted into binary. Binary code uses 0s and 1s to store information. For example, the letter ‘a’ is written as ‘01000001’ in this basic computer language.
2. The binary data is divided into blocks of 512 bits. If the block is smaller than 512, it’ll be expanded to that size by adding bits of “padding.” If it’s larger, it’ll be broken into blocks of 512 bits. (If the last block isn’t exactly 512 bits, padding is added to the last block to make it 512 bits.)
3. The message is further divided into smaller blocks that are 32 bits each.
4. Sixty-four iterations (rounds) of compression functions are performed, wherein the hash values generated above are rotated in a specific pattern and additional data gets added.
5. New hash values are created from the output of the previous operations.
6. In the last round, one final 256-bit hash value is produced — this hash digest is the end product of SHA 256.

SHA 256

Padding Bits: It adds some extra bits to the message, such that the length is exactly 64 bits short of a multiple of 512. During the addition, the first bit should be one, and the rest of it should be filled with zeroes.

Padding Length: You can add 64 bits of data now to make the final plaintext a multiple of 512. You can calculate these 64 bits of characters by applying the modulus to your original cleartext without the padding.

Initialising the Buffers: You need to initialize the default values for eight buffers to be used in the rounds as follows:

a = 0x6a09e667

b = 0xbb67ae85

c = 0x3c6ef372

d = 0xa54ff53a

e = 0x510e527f

f = 0x9b05688c

g = 0x1f83d9ab

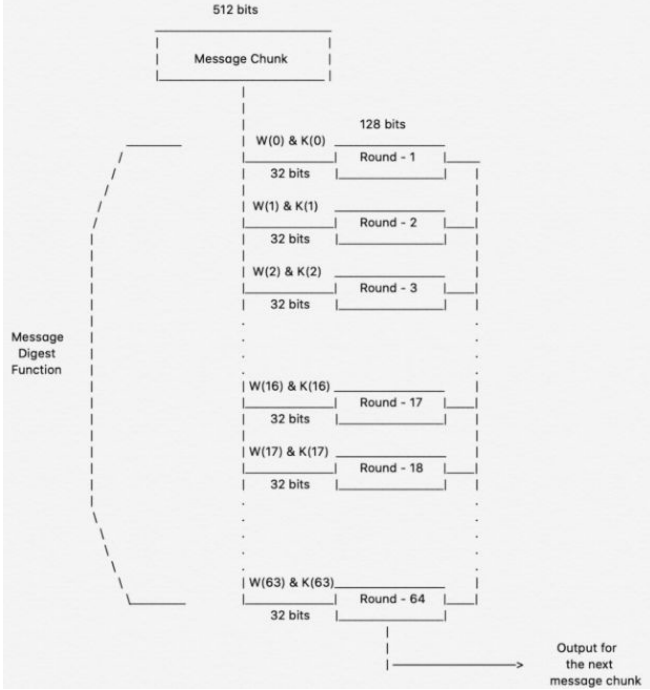
h = 0x5be0cd19

You also need to store 64 different keys in an array, ranging from K[0] to K[63]. They are initialized as follows:

```
k[0..63] :=
   0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
   0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
   0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
   0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
   0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
   0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
   0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
   0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208, 0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2
```

SHA 256

Compression Functions: The entire message gets broken down into multiple blocks of 512 bits each. It puts each block through 64 rounds of operation, with the output of each block serving as the input for the following block. The entire process is as follows:



SHA 512

- SHA-512, or Secure Hash Algorithm 512, is a hashing algorithm used to convert text of any length into a fixed-size string. Each output produces a SHA-512 length of 512 bits (64 bytes).
- This algorithm is commonly used for email addresses hashing, password hashing, and digital record verification. SHA-512 is also used in blockchain technology.
- The algorithm takes as input a message with a maximum length of less than 128 bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks. Figure 11.9 depicts the overall processing of a message to produce a digest.
- **Step 1: Append padding bits:** The message is padded so that its length is congruent to 896 modulo 1024 [length K 896(mod 1024)]. Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 1024. The padding consists of a single 1 bit followed by the necessary number of 0 bits.
- **STEP 2: Append length**. A block of 128 bits is appended to the message. This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding).
- The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length. In Figure 11.9, the expanded message is represented as the sequence of 1024-bit blocks M1, M2, c, MN, so that the total length of the expanded message is N * 1024bits.

Step 3 Initialize hash buffer. A 512-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h). These registers are initialized to the following 64-bit integers (hexadecimal values):

```
a = 6A09E667F3BCC908    e = 510E527FADE682D1
b = BB67AE8584CAA73B    f = 9B05688C2B3E6C1F
c = 3C6EF372FE94F82B    g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1    h = 5BE0CD19137E2179
```

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.

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, Hi-1.

Each round t makes use of a 64-bit value Wt, derived from the current 1024-bit block being processed (Mi). These values are derived using a message schedule described subsequently. Each round also makes use of an additive constant Kt, where 0 … t … 79 indicates one of the 80 rounds.

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

Digital Signature Scheme

- Informally, a **digital signature** is a technique for establishing the origin of a particular message in order to settle later disputes about what message (if any) was sent.
- The purpose of a digital signature is thus for an entity to **bind its identity to a message**.
- We use the term **signer** for an entity who creates a digital signature, and the term **verifier** for an entity who receives a signed message and attempts to check whether the digital signature is "correct" or not.
- Digital signatures have many **attractive properties** and it is very important to understand exactly what **assurances** they provide and what their **limitations** are.

Security requirements

We will define a **digital signature** on a message to be some data that provides:

- **Data origin authentication of the signer**
  - A digital signature validates the message in the sense that assurance is provided about the integrity of the message and of the identity of the entity that signed the message.
- **Non-repudiation**
  - A digital signature can be stored by anyone who receives the signed message as evidence that the message was sent and of who sent it. This evidence could later be presented to a third party who could use the evidence to resolve any dispute that relates to the contents and/or origin of the message.

Input to a digital signature

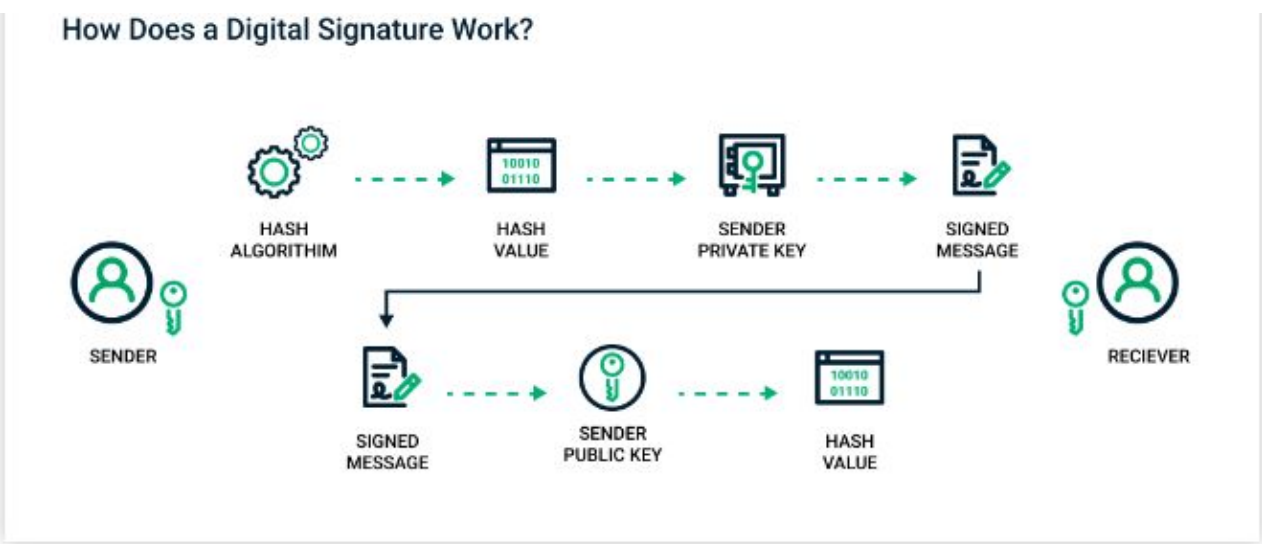
- **The message**
  - Since a digital signature needs to offer data origin authentication (and non-repudiation) it is clear that the digital signature itself must be a piece of data that depends on the message, and cannot be a completely separate identifier.
  - It may be **sent** as a separate piece of data to the message, but its computation must involve the message.
- **A secret parameter known only by the signer**
  - Since a digital signature needs to offer non-repudiation, its calculation must involve a secret parameter that is known only by the signer.
  - The only possible exception to this rule is if the other entity is totally trusted by all parties involved in the signing and verifying of digital signatures.

Properties of a digital signature

- **Easy for the signer to sign a message**
  - There is no point in having a digital signature scheme that involves the signer needing to use slow and complex operations to compute a digital signature.
- **Easy for anyone to verify a message**
  - Similarly we would like the verification of a digital signature to be as efficient as possible.
- **Hard for anyone to forge a digital signature**
  - It should be practically impossible for anyone who is not the legitimate signer to compute a digital signature on a message that appears to be valid. By "appears to be valid" we mean that anyone who attempts to verify the digital signature is led to believe that they have just successfully verified a valid digital signature on a message.

How it Works?

- A Digital Signature Scheme will have two components, a private signing algorithm which permits a user to securely sign a message and a public verification algorithm which permits anyone to verify that the signature is authentic.
- The signing algorithm needs to "bind" a signature to a message in such a way that the signature cannot be pulled out and used to sign another document, or have the original message modified and the signature remain valid.
- For practical reasons it would be necessary for both algorithms to be relatively fast and if small computers such as smart cards are to be used, the algorithms can not be too computationally complex.



Here is how sending a digital signature works:

1. The sender selects the file to be digitally signed in the document platform or application.
2. The sender's computer calculates the unique hash value of the file content.
3. This hash value is encrypted with the sender's private key to create the digital signature.
4. The original file along with its digital signature is sent to the receiver.
5. The receiver uses the associated document application, which identifies that the file has been digitally signed.
6. The receiver's computer then decrypts the digital signature using the sender's public key.

RSA Encryption: Algorithm

- Bob (Key generation):
  1. Generate two large random primes p and q.
  2. Compute n = pq.
  3. Select a small odd integer e relatively prime with (n).
  4. Compute (n) = (p − 1)(q − 1).
  5. Compute d = e−1 mod (n).

  PB = (e, n) is Bob's RSA public key.

  SB = (d, n) is Bob' RSA private key.

Alice (encrypt and send a message M to Bob):

1. Get Bob's public key PB = (e, n).
2. Compute C = Me mod n.

Bob (decrypt a message C received from Alice):

1. Compute M = Cd mod n.

RSA Signature Scheme: Algorithm

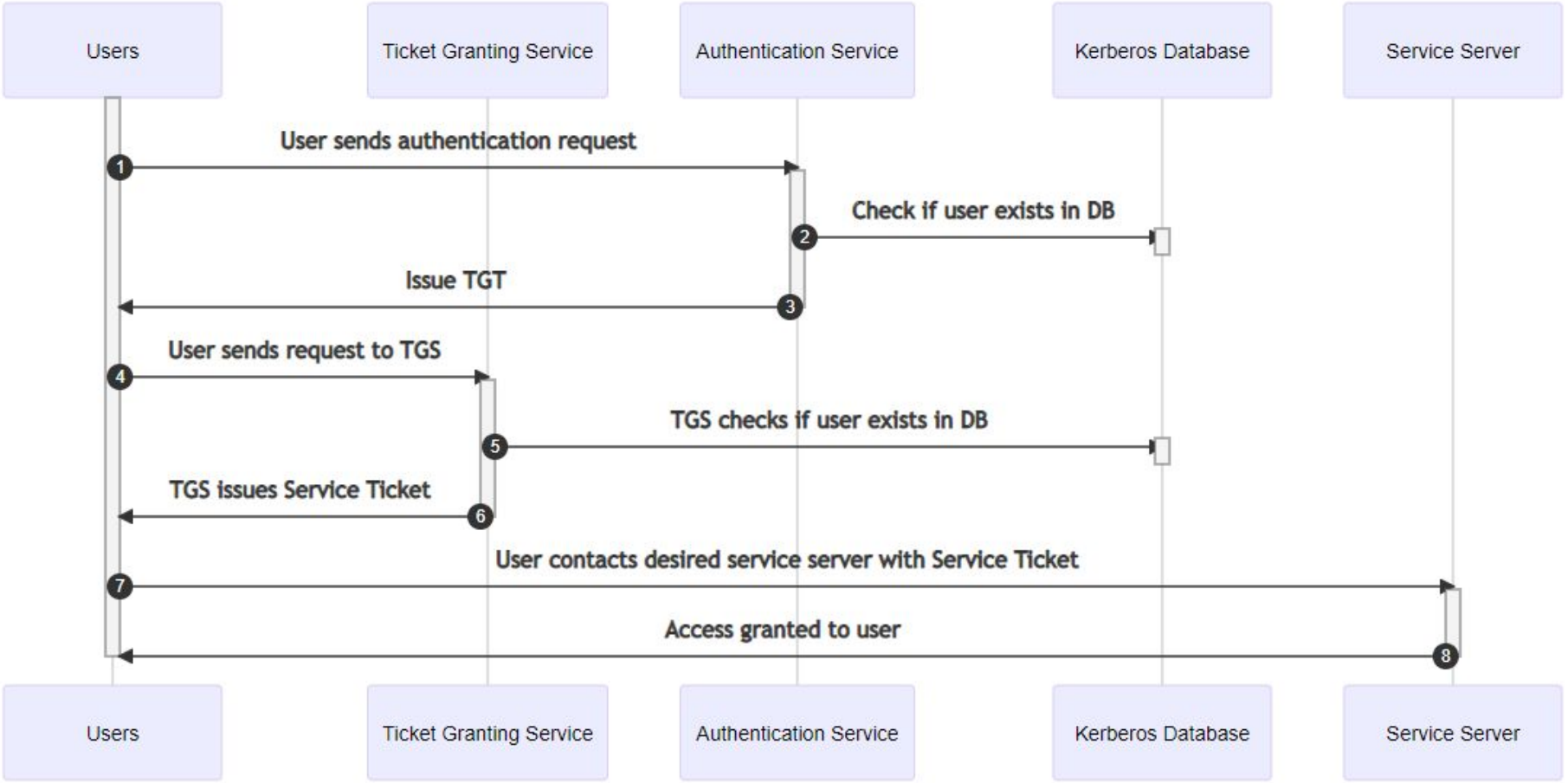
- Bob (Key generation): As before.
  - $P_B = (e, n)$ is Bob's RSA public key.
  - $S_B = (d, n)$ is Bob' RSA private key.
- Bob (sign a secret message M):
  1. Compute $S = M^d \bmod n$.
  2. Send M, S to Alice.
- Alice (verify signature S received from Bob):
  1. Receive M, S from Alice.
  2. Verify that $M \stackrel{?}{=} S^e \bmod n$.

- As we have previously noted, in order for Bob to sign a message m, he raises m to his private decryption exponent mod n. This is the signature algorithm.
- Anyone can verify this signature by raising md to Bob's public encryption exponent mod n. This is the verification algorithm.
- Application of the verification algorithm to a valid signature yields the message m.
- The verifier must know the message m in order to be sure that this is the message that Bob signed, so in this application Bob must send the ordered pair (m, md mod n).
- Some care must be taken in the construction of the message to be signed in this way. For instance, if m is the instruction to Bob's bank to issue a check to Alice, then if Alice intercepts the ordered pair, she can send the same pair to Bob's bank whenever she is a little low on cash.
- To prevent this kind of abuse, when it matters, messages should include dates and other such items which prevent the message from being reused.

KERBEROS

- Kerberos is a protocol for authenticating service requests between trusted hosts across an untrusted network, such as the internet. Kerberos support is built in to all major computer operating systems, including Microsoft Windows, Apple macOS, FreeBSD and Linux.
- Since Windows 2000, Microsoft has used the Kerberos protocol as the default authentication method in Windows, and it is an integral part of the Windows Active Directory (AD) service. Broadband service providers also use the protocol to authenticate cable modems and set-top boxes accessing their networks.
- Kerberos was developed for Project Athena at the Massachusetts Institute of Technology (MIT). The name was taken from Greek mythology; Kerberos (Cerberus) was a three-headed dog who guarded the gates of Hades. The three heads of the Kerberos protocol represent the following:\

1. the client or principal;
2. the network resource, which is the application server that provides access to the network resource;
3. a key distribution center (KDC), which acts as Kerberos' trusted third-party authentication service


Users
Ticket Granting Service
Authentication Service
Kerberos Database
Service Server
User sends authentication request
1
Check if user exists in DB
2
Issue TGT
3
User sends request to TGS
4
TGS checks if user exists in DB
5
TGS issues Service Ticket
6
User contacts desired service server with Service Ticket
7
Access granted to user
8
Users
Ticket Granting Service
Authentication Service
Kerberos Database
Service Server

When a user requests access to a service through the authentication service, they enter their username and password locally, and send the following information:

1. Security Identifier (SID)
2. Name of the requested service (for example, example.cool.hat)
3. User's IP address
4. Desired lifetime of the Ticket granting ticket (TGT). The default is 10 hours and can be changed via Group Policy. Authentication service issues a ticket granting ticket (TGT) if the user exists in the database. The first message sent back to the user contains:

1. Security identifier (SID)
2. TGS ID
3. Timestamp
4. User's IP address
5. TGT lifetime
6. TGT
7. Session key

After this message, another message will be sent containing:

1. TGS ID
2. Timestamp
3. Session key

The user sends the TGT to the TGS along with the Kerberos ID of the requested services. Another message is sent containing the "Authenticator", which is composed of the User ID and timestamp, encrypted with the user's session key.

The TGS will respond to the user with two messages if it finds the user's information within the Kerberos database. The first message will contain the following information, encrypted with the server's secret service key:

1. Service ticket
2. User's ID
3. User's IP address
4. Validity period
5. Service session key

A second message, encrypted with the user's session key (for example a locked box within a locked box, where the user can only unlock the first box), will contain the service session key.

The user sends the service ticket to the requested service along with the service request in two messages. The first message will be the first message from the previous step (encrypted with the server's secret service key). The second message will contain a new Authenticator with an updated timestamp, encrypted with the user's session key.

The service server decrypts the ticket using its own secret key to retrieve the user's session key, which is used to decrypt the authenticator. If the user's ID from previous messages matches, it will send a message encrypted with the user's session key to the user with the timestamp found in the new authenticator to confirm the service's identity.

PKI

- Public key infrastructure (PKI) is a catch-all term for everything used to establish and manage public key encryption, one of the most common forms of internet encryption.
- It is baked into every web browser in use today to secure traffic across the public internet, but organizations can also deploy it to secure their internal communications and access to connected devices.
- *Public Key Infrastructure* (PKI) is a technology for authenticating users and devices in the digital world. The basic idea is to have one or more trusted parties digitally sign documents certifying that a particular cryptographic key belongs to a particular user or device. The key can then be used as an **identity for the user** in digital networks.
- A public key infrastructure relies on digital signature technology, which uses **public key cryptography**. The basic idea is that the secret key of each entity is only known by that entity and is used for signing. This key is called the **private key**. There is another key derived from it, called the **public key**, which is used for verifying signatures but cannot be used to sign. This public key is made available to anyone, and is typically included in the certificate document.

HOW IT WORKS

- PKI certificates are documents that act as digital passports, assigned to any entity that wants to participate in a PKI-secured conversation.
- They can include quite a bit of data. One of the most important pieces of information a certificate includes is the entity's public key: the certificate is the mechanism by which that key is shared. But there's also the authentication piece.
- A certificate includes an attestation from a trusted source that the entity is who they claim to be. That trusted source is generally known as a **certificate authority (CA)**.
-

With these concepts under our belt, these are the elements that go into PKI.

- A **certificate authority**, which issues digital certificates, signs them with its own public key, and stores them for reference.
- A **registration authority**, which verifies the identities of those requesting digital certificates. A CA can act as its own registration authority or can use a third party to do so.
- A **certificate database** that stores both the certificates and metadata about them—most importantly, the period of time for which the certificate is valid.
- A **certificate policy** outlining the PKI's procedures, which allows outsiders to judge how trustworthy the PKI is.

X.509

- X.509 is a digital certificate that is built on top of a widely trusted standard known as ITU or International Telecommunication Union X.509 standard, in which the format of PKI certificates is defined.
- X.509 digital certificate is a certificate-based authentication security framework that can be used for providing secure transaction processing and private information. These are primarily used for handling the security and identity in computer networking and internet-based communications.
- The core of the X.509 authentication service is the public key certificate connected to each user. These user certificates are assumed to be produced by some trusted certification authority and positioned in the directory by the user or the certified authority.
- Once an X.509 certificate is provided to a user by the certified authority, that certificate is attached to it like an identity card. The chances of someone stealing it or losing it are less, unlike other unsecured passwords. With the help of this analogy, it is easier to imagine how this authentication works: the certificate is basically presented like an identity at the resource that requires authentication.

Generally, the certificate includes the elements given below:

- **Version number:** It defines the X.509 version that concerns the certificate.
- **Serial number:** It is the unique number that the certified authority issues.
- **Signature Algorithm Identifier:** This is the algorithm that is used for signing the certificate.
- **Issuer name:** Tells about the X.500 name of the certified authority which signed and created the certificate.
- **Period of Validity:** It defines the period for which the certificate is valid.
- **Subject Name:** Tells about the name of the user to whom this certificate has been issued.
- **Subject's public key information:** It defines the subject's public key along with an identifier of the algorithm for which this key is supposed to be used.
- **Extension block:** This field contains additional standard information.
- **Signature:** This field contains the hash code of all other fields which is encrypted by the certified authority private key.

www.ssl.com

Issued by: SSL.com EV SSL Intermediate CA RSA R3

Expires: Saturday, April 17, 2021 at 5:15:06 PM Central Daylight Time

This certificate is valid

▼ **Details**

Subject Name	
Country or Region	US
State/Province	Texas
Locality	Houston
Organization	SSL Corp
Serial Number	NV20081614243
Common Name	www.ssl.com
Postal Code	77098
Business Category	Private Organization
Street Address	3100 Richmond Ave
Inc. State/Province	Nevada
Inc. Country/Region	US