Basics of Information and Network Security

- In daily life we use information for various purposes and use network for communication and exchange information between different parties.
- In many cases the information is sensitive so we need to take care that only authorized party can get that information.
- For maintaining such privacy, we require some mechanism or physical device which ensures that it is safe. Such mechanism or physical devices are known as **security system**.
- **Computer Security:** The protection afforded to an automated information system in order to attain the applicable objectives of preserving the **integrity**, **availability**, and **confidentiality** of information system resources.
- This definition of computer security introduces three key objectives that are at the heart of computer security:
  1. **Confidentiality:** It covers two concepts
     Data Confidentiality: Assures that private or confidential information is not made available or disclosed to unauthorized individuals.
     Privacy: Assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.
  2. **Integrity:** It covers two concepts
     Data Integrity: Assures that information and programs are changed only in a specified and authorize manner.
     System Integrity: Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.
  3. **Availability:** Assures that systems work promptly and service is not denied to authorize user.
- **Unconditionally secure algorithm:** An algorithm or an encryption scheme is unconditionally secure if the attacker cannot obtain the corresponding plaintext from ciphertext no matter how much ciphertext is available.
- **Computationally secure algorithm:** An encryption scheme is said to be computationally secure if either of the following criteria is met:
  - The cost of breaking the cipher exceeds the value of the encrypted information.
  - The time required to break the cipher exceeds the useful lifetime of the information.
- **Threat:** A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger that might exploit vulnerability.

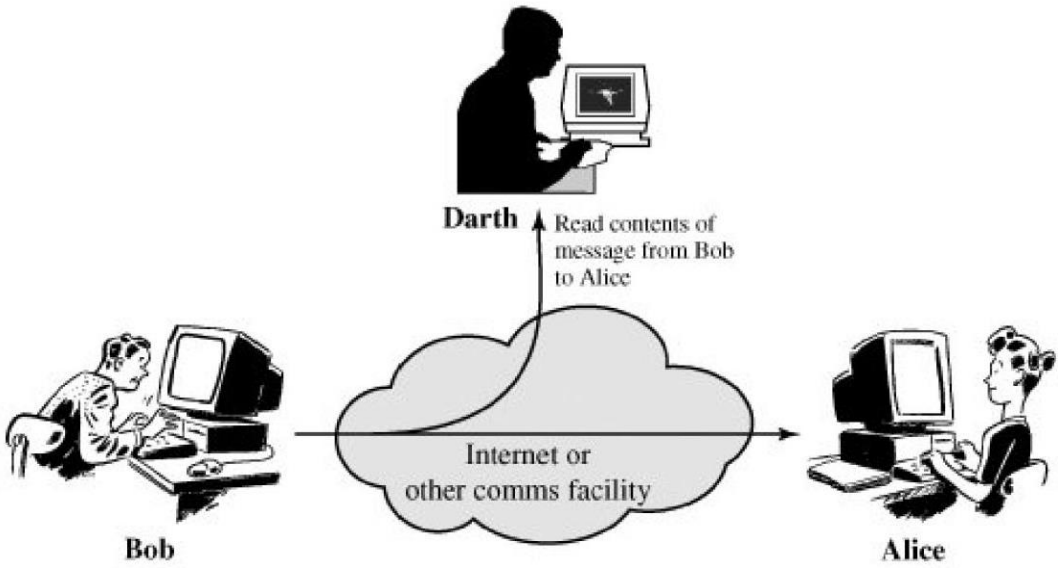
Security Attacks

- **Security Attacks:** An attack is an action that comprises the information or network security.
- There are two types of attacks:
  1. Passive Attack
  2. Active Attack

Passive Attack

- **Passive Attack:** The attacker only monitors the traffic attacking the confidentiality of the data. It contains **release of message contents** and **traffic analysis** (in case of encrypted data).
  1. **Release of message contents:**
     - The release of message contents is easily understood.

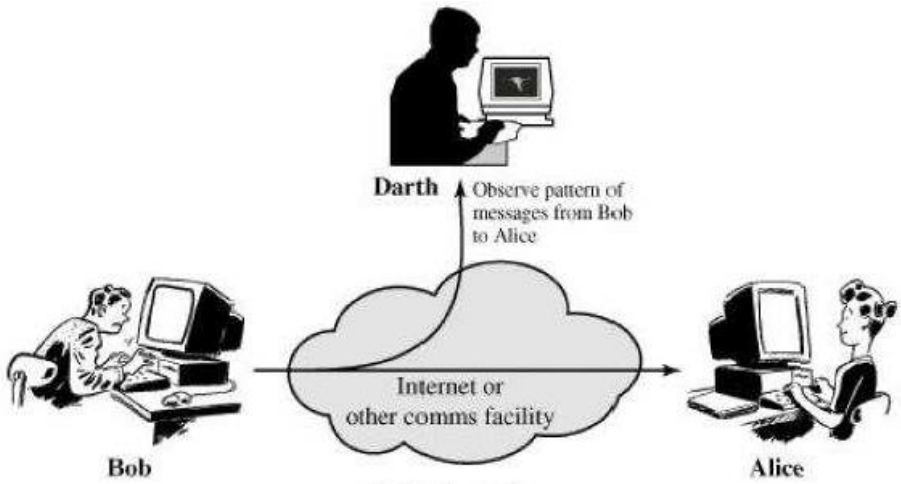
- A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information.
- We would like to prevent an opponent from learning the contents of these transmissions.



(a) Release of message contents

2. Traffic analysis:

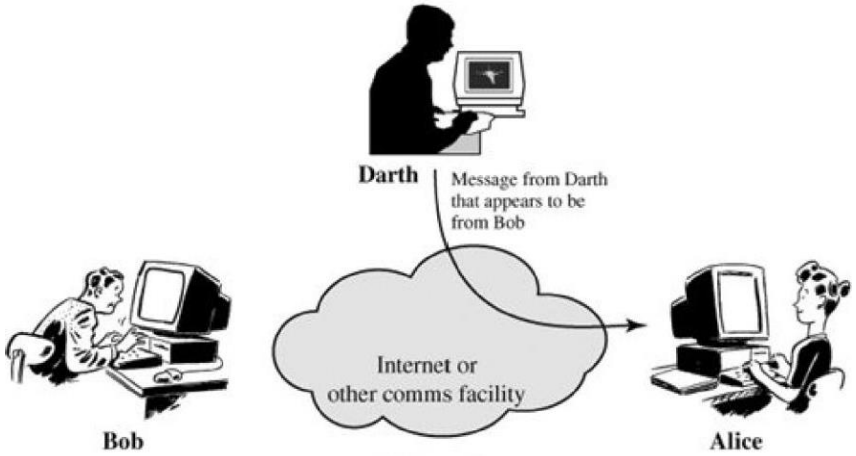
- A second type of passive attack, traffic analysis.
- Suppose that we had a way of masking the contents of messages or other information.
- Even if they captured the message, could not extract the information from the message.
- The common technique for masking contents is encryption.
- If we had encryption protection in place, an opponent might still be able to observe the pattern of these messages.
- The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged.
- This information might be useful in guessing the nature of the communication that was taking place.
- Passive attacks are very difficult to detect because they do not involve any alteration of the data.
- Typically, the message traffic is sent and received in an apparently normal fashion and the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern.



(b) Traffic analysis

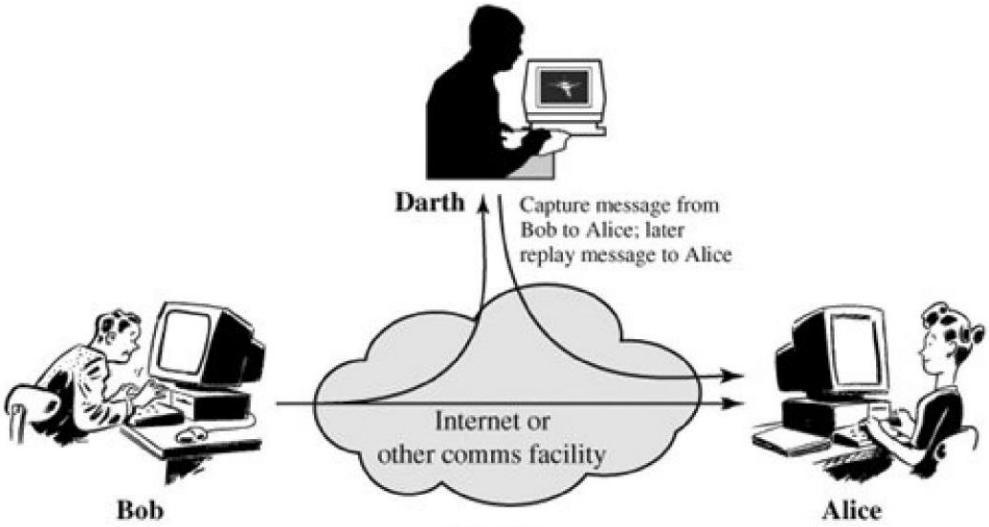
Active attack

- **Active attack:** Attacker tries to alter transmitted data. It includes masquerade, modification, replay and denial of service.
  1. **Masquerade:** A masquerade takes place when one entity pretends to be a different entity (Figure a). A masquerade attack usually includes one of the other forms of active attack.



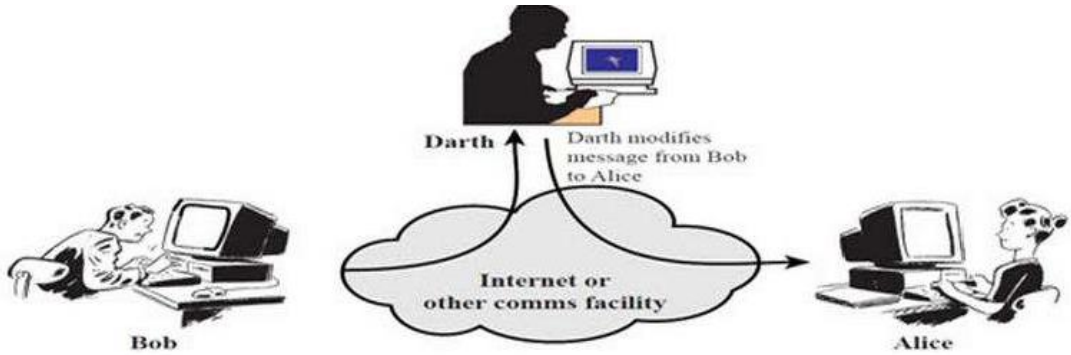
(a) Masquerade

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



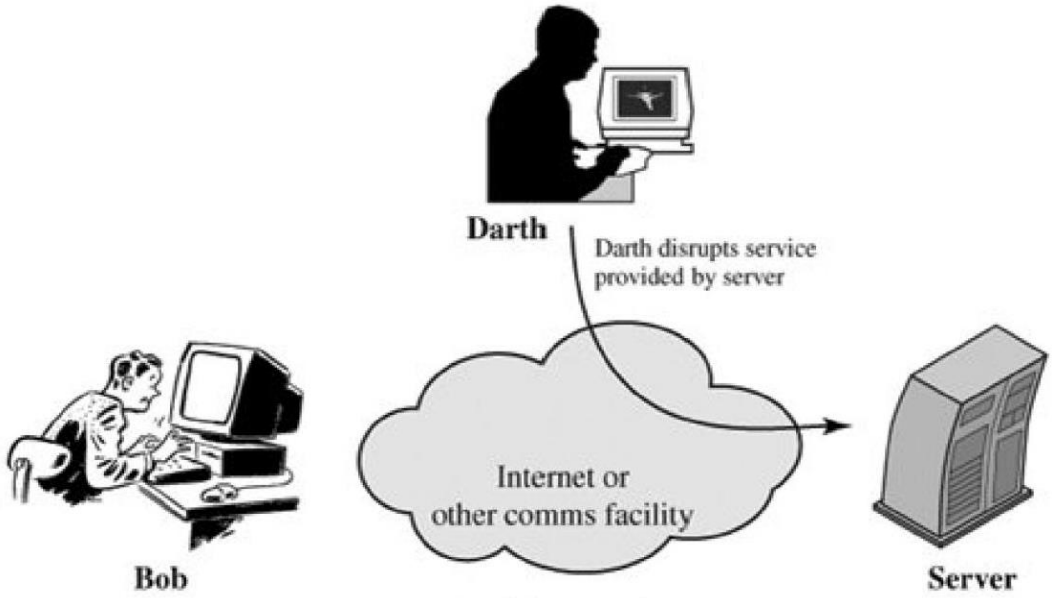
(b) Replay

  3. **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

4. **Denial of service:**
   - The denial of service prevents or inhibits the normal use or management of communications facilities.
   - 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.

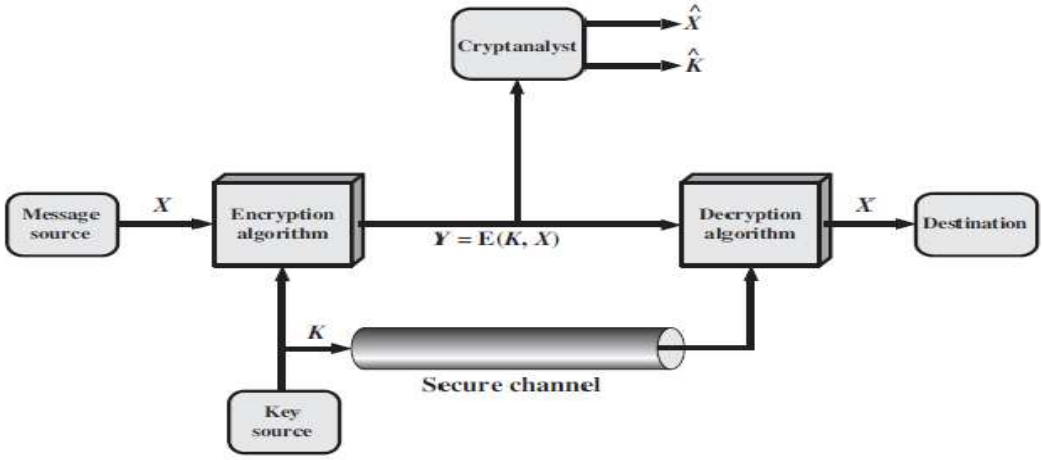


(d) Denial of service

Security services

- A security service is a processing or communicating service that can prevent or detect the above-mentioned attacks. Various security services are:
  - **Authentication:** the recipient should be able to identify the sender, and verify that the sender, who claims to be the sender, actually did send the message.
  - **Data Confidentiality:** An attacker should not be able to read the transmitted data or extract data in case of encrypted data. In short, confidentiality is the protection of transmitted data from passive attacks.
  - **Data Integrity:** Make sure that the message received was exactly the message the sender sent.
  - **Nonrepudiation:** The sender should be able to send the message. The receiver should be able to receive the message.

Symmetric Cipher Model



- A symmetric cipher model broadly contains five parts.
- **Plaintext:** This is the original intelligible message.
- **Encryption algorithm:** The encryption algorithm performs various substitutions and transformations on the plaintext. It takes in plaintext and key and gives the ciphertext.
- **Secret key:** The key is a value independent of the plaintext and of the algorithm. Different keys will yield different outputs.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the secret key.
- **Decryption algorithm:** Runs on the ciphertext and the key to produce the plaintext. This is essentially the encryption algorithm run in reverse.
- Two basic requirements of encryption are:
  1. Encryption algorithm should be strong. An attacker knowing the algorithm and having any number of ciphertext should not be able to decrypt the ciphertext or guess the key.
  2. The key shared by the sender and the receiver should be secret.
- Let the plaintext be $X = [X1, X2,\ldots, X_M]$, key be $K = [K1, K2,\ldots, K_J]$ and the ciphertext produced be $Y = [Y1, Y2,\ldots, Y_N]$. Then, we can write

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

- Here E represents the encryption algorithm and is a function of plaintext X and key K.
- The receiver at the other ends decrypts the ciphertext using the key.

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

- Here D represents the decryption algorithm and it inverts the transformations of encryption algorithm.
- An opponent not having access to X or K may attempt to recover K or X or both.
- It is assumed that the opponent knows the encryption (E) and decryption (D) algorithms.
- If the opponent is interested in only this particular message, then the focus of the effort is to recover by generating a plaintext estimate $\hat{X}$.
- If the opponent is interested in being able to read future messages as well then, he will attempt to recover the key by making an estimate $\hat{K}$.

Cryptography

- **Cryptography:** The area of study containing the principles and methods of transforming an intelligible message into one that is unintelligible, and then retransforming that message back to its original form.
- Cryptographic systems are characterized along three independent dimensions.
  1. **The types of operations used for transforming plaintext to ciphertext.** All encryption algorithms are based on two general principles substitution, and transposition. Basic requirement is that no information be lost. Most systems referred to as product system, involves multiple stages of substitutions and transpositions.
  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 process a block at a time and produce an output block for each input block. A stream cipher process the input element continuously, producing output one element at a time, as it goes along.

Cryptanalysis and Brute-Force Attack

- **Cryptanalysis:** The process of trying to break any cipher text message to obtain the original plain text message.
- **Brute-force attack:** The attacker tries every possible key on a piece of ciphertext until plaintext is obtained. On average, half of all possible keys must be tried to achieve success.
- Based on the amount of information known to the cryptanalyst cryptanalytic attacks can be categorized as:
  - **Cipher text Only Attack:** The attacker knows only cipher text only. It is easiest to defend.
  - **Known plaintext Attack:** In this type of attack, the opponent has some plaintext-cipher text pairs. Or the analyst may know that certain plaintext patterns will appear in a message. For example, there may be a standardized header or banner to an electronic funds transfer message and the attacker can use that for generating plaintext-cipher text pairs.
  - **Chosen plaintext:** If the analyst is able somehow to get the source system to insert into the system a message chosen by the analyst, then a *chosen-plaintext* attack is possible. In such a case, the analyst will pick patterns that can be expected to reveal the structure of the key.
  - **Chosen Cipher text:** In this attack, the analyst has cipher text and some plaintext-cipher text pairs where cipher text has been chosen by the analyst.
  - **Chosen Text:** Here, the attacker has got cipher text, chosen plaintext-cipher text pairs and chosen cipher text-plaintext pairs.
- Chosen cipher text and chosen text attacks are rarely used.
- It is assumed that the attacker knows the encryption and decryption algorithms.
- Generally, an encryption algorithm is designed to withstand a known-plaintext attack.

Brute-force attack

- This type of attack becomes impractical as the key size increases as trying all the possible alternative keys for a very large key may take a huge amount of time.
- For example, for a binary key of 128 bits, 2^{128} keys are possible which would require around 5 $X10^{24}$ years at the rate of 1 decryption per microsecond (current machine's speed).

- The Data Encryption Standard (DES) algorithm uses a 56-bit key a 128-bit key is used in AES.
- With massively parallel systems, even DES is also not secure against Brute Force attack.
- AES with its 128-bit key is secure since the time required to break it makes it impractical to try Brute-Force attack

Substitution Techniques

- Various conventional encryption schemes or substitution techniques are as under:

Caesar cipher

- The encryption rule is simple; replace each letter of the alphabet with the letter standing 3 places further down the alphabet.
- The alphabet is wrapped around so that Z follows A.
- Example:
  Plaintext: MEET ME AFTER THE PARTY
  Ciphertext: PHHW PH DIWHU WKH SDUWB
- Here, the key is 3. If different key is used, different substitution will be obtained.
- Mathematically, starting from a=0, b=1 and so on, Caesar cipher can be written as:

$$E(p) = (p + k) \bmod (26)$$
$$D(C) = (C - k) \bmod (26)$$

- This cipher can be broken
  - If we know one plaintext-cipher text pair since the difference will be same.
  - By applying Brute Force attack as there are only 26 possible keys.

Monoalphabetic Substitution Cipher

- Instead of shifting alphabets by fixed amount as in Caesar cipher, any random permutation is assigned to the alphabets. This type of encryption is called monoalphabetic substitution cipher.
- For example, A is replaced by Q, B by D, C by T etc. then it will be comparatively stronger than Caesar cipher.
- The number of alternative keys possible now becomes 26!
- Thus, Brute Force attack is impractical in this case.
- However, another attack is possible. Human languages are redundant i.e., certain characters are usedmore frequently than others. This fact can be exploited.
- These are easy to break because they reflect the frequency data of the original alphabet.
- Moreover, diagrams like ‘th’ and trigrams like ‘the’ are also more frequent.
- Tables of frequency of these letters exist. These can be used to guess the plaintext if the plaintext is in uncompressed English language.

Playfair Cipher

- In this technique multiple (2) letters are encrypted at a time.
- This technique uses a 5 X 5 matrix which is also called key matrix.

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

- The plaintext is encrypted **two letters at a time**:
  - Break the plaintext into pairs of two consecutive letters.
  - If a pair is a repeated letter, insert a filler like 'X'in the plaintext, e.g., "Balloon" is treated as "ba lx loon".
  - If both letters fall in the same row of the key matrix, replace each with the letter to its right (wrapping back to start from end), e.g., "AR" encrypts as "RM".
  - If both letters fall in the same column, replace each with the letter below it (again wrapping to top from bottom), e.g., "MU" encrypts to "CM".
  - Otherwise, each letter is replaced by the one in its row in the column of the other letter of the pair, e.g., "HS" encrypts to "BP", and "EA" to "IM" or "JM" (as desired)
- Security is much improved over monoalphabetic as here two letters are encrypted at a time and hence there are 26 X 26 =676 diagrams and hence it needs a 676-entry frequency table.
- However, it can be broken even if a few hundred letters are known as much of plaintext structure is retained in cipher text.

Hill Cipher

- This cipher is based on linear algebra.
- Each letter is represented by numbers from 0 to 25 and calculations are done modulo 26.
- This encryption algorithm takes m successive plaintext letters and substitutes them with m cipher text letters.
- The substitution is determined by m linear equations. For *m* = 3, the system can be described as:

$$c_1 = (k_{11}p_1 + k_{12}p_2 + k_{13}p_3) \bmod 26$$
$$c_2 = (k_{21}p_1 + k_{22}p_2 + k_{23}p_3) \bmod 26$$
$$c_3 = (k_{31}p_1 + k_{32}p_2 + k_{33}p_3) \bmod 26$$

- This can also be expressed in terms of row vectors and matrices.

$$(c_1\ c_2\ c_3) = (p_1\ p_2\ p_3) \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_3 & k_{33} \end{pmatrix} \bmod 26$$

  Where **C** and **P** are row vectors of length 3 representing the plaintext and cipher text, and **K** is a 3 X 3 matrix representing the encryption key

- Key is an invertible matrix K modulo 26, of size m. For example:

$$K = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \qquad K^{-1} = \begin{pmatrix} 4 & 19 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

- Encryption and decryption can be given by the following formulae:
  Encryption: $C = PK \bmod 26$, Decryption: $P = CK^{-1} \bmod 26$

- The strength of the Hill cipher is that it completely hides single-letter frequencies.
- Although the Hill cipher is strong against a cipher text-only attack, it is easily broken with a known plaintext attack.
  - Collect m pair of plaintext-cipher text, where m is the size of the key.
  - Write the m plaintexts as the rows of a square matrix P of size m.
  - Write the m cipher texts as the rows of a square matrix C of size m.
  - We have that C=PK mod 26.
  - If P is invertible, then $K=P^{-1}C$ mod 26,
  - If P is not invertible, then collect more plaintext-cipher text pairs until an invertible P is obtained.

The Vigenère cipher

- This is a type of polyalphabetic substitution cipher (includes multiple substitutions depending on the key). In this type of cipher, the key determines which particular substitution is to be used.
- To encrypt a message, a key is needed that is as long as the message. Usually, the key is a repeating keyword.
- For example, if the keyword is *deceptive*, the message "we are discovered save yourself" is encrypted as follows:
  Key: *deceptivedecept*
  Plaintext: wearediscovered
  Ciphertext: ZICVTWQNGRZGVTW
- Encryption can be done by looking in the Vigenere Table where ciphertext is the letter key's row and plaintext's column or by the following formula:

$$C_i = (P_i + K_{i\ mod\ m})\ mod\ 26$$

- Decryption is equally simple. The key letter again identifies the row. The position of the cipher text letter in that row determines the column, and the plaintext letter is at the top of that column.
- The strength of this cipher is that there are multiple ciphertext letters for each plaintext letter, one for each unique letter of the keyword.
- Thus, the letter frequency information is obscured however, not all knowledge of the plaintext structure is lost.

Vernam Cipher

- This system works on binary data (bits) rather than letters.
- The technique can be expressed as follows:

$$C_i = P_i \oplus K_i$$

  Where
  P_i = i^{th} binary digit of plaintext.
  K_i = i^{th} binary digit of key.
  C_i = i^{th} binary digit of ciphertext.
  $\oplus$ = exclusive-or (XOR) operation
- Thus, the ciphertext is generated by performing the bitwise XOR of the plaintext and the key.
- Decryption simply involves the same bitwise operation:

$$P_i = C_i \oplus K_i$$

- The essence of this technique is the means of construction of the key.
- It was produced by the use of a running loop of tape that eventually repeated the key, so that in fact the system worked with a very long but repeating keyword.

- Although such a scheme has cryptanalytic difficulties, but it can be broken with a very long ciphertext or known plaintext as the key is repeated.

One-Time Pad

- In this scheme, a random key that is as long as the message is used.
- The key is used to encrypt and decrypt a single message, and then is discarded. Each new message requires a new key of the same length as the new message.
- This scheme is unbreakable.
- It produces random output that bears no statistical relationship to the plaintext.
- Because the ciphertext contains no information whatsoever about the plaintext, there is simply no way to break the code.
- For any plaintext of equal length to the ciphertext, there is a key that produces that plaintext.
- Therefore, if you did an exhaustive search of all possible keys, you would end up with many legible plaintexts, with no way of knowing which the intended plaintext was.
- Therefore, the code is unbreakable.
- The security of the one-time pad is entirely due to the randomness of the key.
- The one-time pad offers complete security but, in practice, has two fundamental difficulties:
  - There is the practical problem of making large quantities of random keys. Any heavily used system might require millions of random characters on a regular basis. Supplying truly random characters in this volume is a significant task.
  - Another problem is that of key distribution and protection. For every message to be sent, a key of equal length is needed by both sender and receiver.
- Because of these difficulties, the one-time pad is used where very high security is required.
- The one-time pad is the only cryptosystem that exhibits **perfect secrecy**.

Transposition Techniques

- A transposition cipher rearranges the characters in the plaintext to form the cipher text. The letters are not changed.
- The simplest such cipher is the **rail fence** technique.

Rail Fence Technique

- Encryption involves writing plaintext letters diagonally over a number of rows, then read off cipher row by row.
- For example, the text “meet me after the party” can be written (in 2 rows) as:

- Ciphertext is read from the above row-by-row:
  MEMATRHPRY ETEFETEAT
- This scheme is very easy to cryptanalyze as no key is involved.
- Transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is a more complex permutation that is not easily reconstructed.

Difference between Symmetric and Asymmetric key cryptography

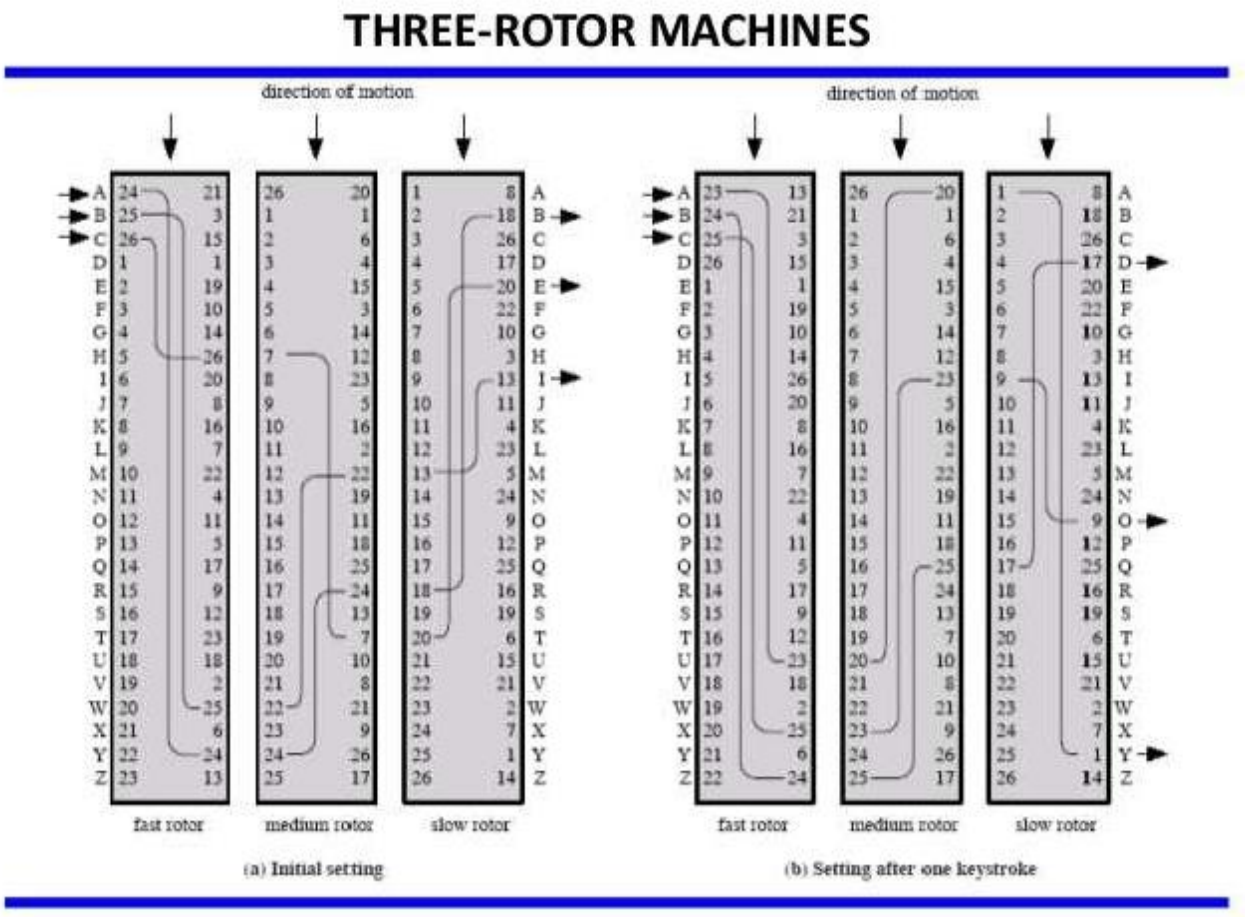
\]
;

Symmetric key Cryptography	Asymmetric Key Cryptography
Symmetric key cryptography uses the same secret (private) key to encrypt and decrypt its data	Asymmetric key cryptography uses a public and a private key to encrypt and decrypt its data
The secret key must be known by both parties.	The public key is known to anyone with which they can encrypt the data but it can only be decoded by the person having the private key
In key distribution process, key information may have to be shared which decreases the security.	Here, the need for sharing key with key distribution center is eliminated.
Symmetric key encryption is faster than asymmetric key.	It is Slower than symmetric key encryption.
Basic operations used in encryption/ decryption are transposition and substitution.	It uses mathematical functions.

Steganography

- Plaintext message may be hidden in one of two ways.
  - Conceal the existence of the message-Steganography.
  - Render the message unintelligible to outsiders by various transformations of the text-Cryptography
- A simple but time consuming form of steganography is the one in which an arrangement of words or letters within an apparently normal text spells out the real message.
- For example, the sequence of first letters of each word of the overall message spells out the hidden message.
- Some other techniques that have been used historically are listed below:
  - **Character marking:** Selected letters of printed or typewritten text are overwritten in pencil. The marks are ordinarily not visible unless the paper is held at an angle to bright light.
  - **Invisible ink:** A number of substances can be used for writing but leave no visible trace until heat or some chemical is applied to the paper.
  - **Pin punctures:** Small pin punctures on selected letters are ordinarily not visible unless the paper is held up in front of a light.
  - **Typewriter correction ribbon:** Used between lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.
- Although these techniques may seem ancient, they have modern equivalents.
- For example, suppose an image has a resolution of 2048 X 3072 pixels where each pixel is denoted by 24 bits (Kodak CD photo format).
- The least significant bit of each 24-bit pixel can be changed without greatly affecting the quality of the image.
- The result is that you can hide a 2.3-megabyte message in a single digital snapshot.
- There are now a number of software packages available that take this type of approach to steganography.
- Steganography has a number of drawbacks when compared to encryption.
  - It requires a lot of overhead to hide a relatively few bits of information.
  - Once the system is discovered, it becomes virtually worthless.
- The advantage of steganography is that it can be employed by parties who have something to lose if the fact of their secret communication is discovered.

Rotor Machines



- The basic principle of the rotor machine is illustrated in figure. The machine consists of a set of independently rotating cylinders through which electrical pulse can flow.
- Each cylinder has 26 input and 26 output pins, with internal wiring that connect each input pin to unique output pin.
- If we associate each input and output pin with a letter of the alphabet, then a single cylinder defines a monoalphabetic substitution.
- If we use multiple cylinders then we will obtain polyalphabetic substitution.

Block Cipher Principles

Stream Cipher and Block Cipher

- A **stream cipher** is one that encrypts a data stream one bit or one byte at a time. Example of stream cipher are the autokey, Vigenère cipher and vernam cipher.
- A **Block Cipher** is one in which a block of plaintext is treated as a whole and used to produce a cipher text block of equal length. Example of block cipher is DES.

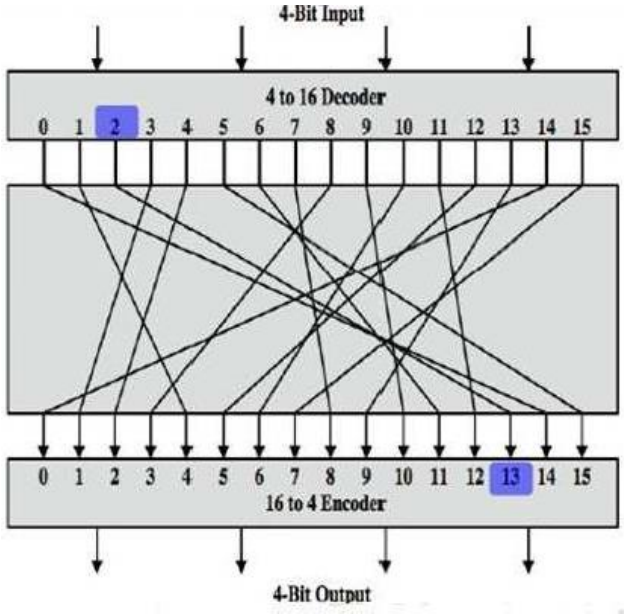
Motivation for the Feistel Cipher Structure

- 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 plain text blocks and for the encryption to be reversible each must produce unique ciphertext block.
- Reversible encryption is also called as **singular encryption**. For example, singular and non-singular transformation 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

- If we limit ourselves to reversible mapping the number of different transformation is $2^n!$.
- Figure below illustrates the logic of a general substitution cipher for n=4



Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

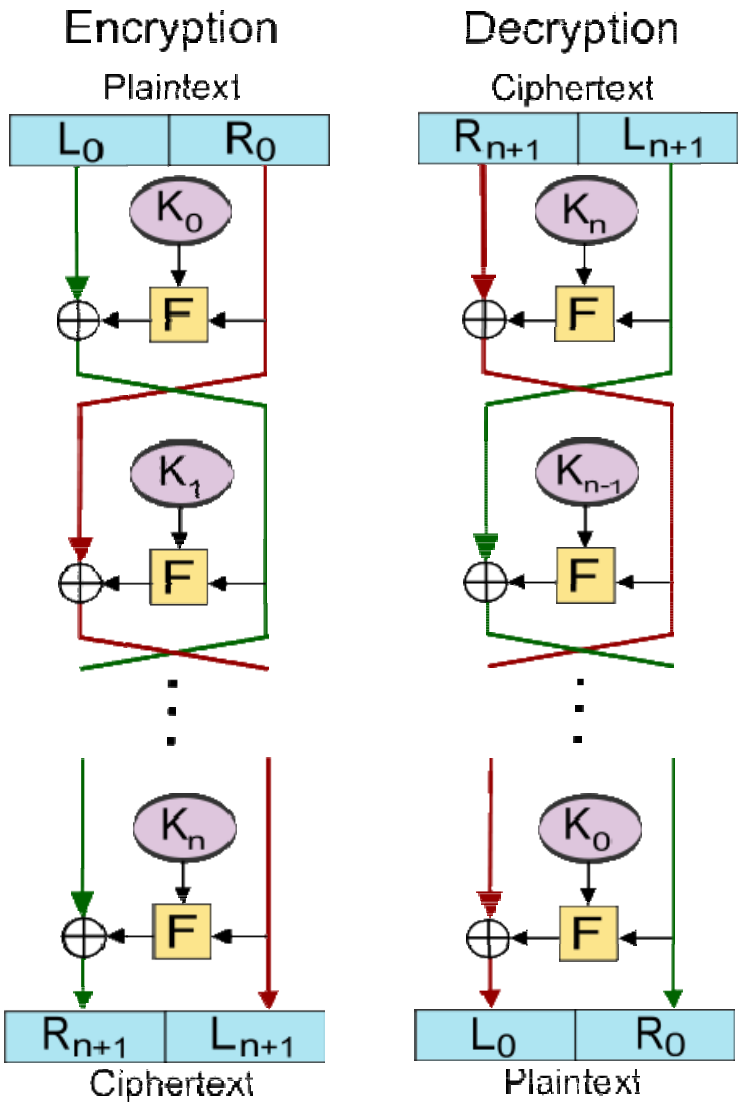
Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

- A 4-bit input produce one of 16 possible input states, which is mapped by substitution cipher into one of unique 16 possible output states, each of which is represented by 4-bit ciphertext.
- The encryption and decryption mapping can be defined by tabulation, as shown in table. This is the most general form of block cipher and can be used to define any reversible mapping between plaintext and ciphertext.
- Feistel refers to this as the ideal block cipher, because it allows for the maximum number of possible encryption mappings from the plaintext block.

- But there is practical problem with ideal block cipher is if we use small block size such as n=4 then it is vulnerable to statistical analysis of the plain text.
- If n is sufficiently large and an arbitrary reversible substitution between plaintext and ciphertext is allowed then statistical analysis is infeasible.
- Ideal block cipher is not practical for large block size according implementation and performance point of view.
- For such transformation mapping itself is a key and we require $nX2^n$ bits for n bit ideal block cipher which is not practical.
- In considering these difficulties, Feistel points out that what is needed is an approximation to the ideal cipher system for large n, built up out of components that are easily realizable.

The Feistel Cipher

- Feistel cipher is based on the idea that instead of using Ideal block cipher which degrades performance, a "substitution-permutation network" can be used.



Feistel Cipher Encryption

- The inputs to the encryption algorithm are a plaintext block of length b bits and a key K.

- The plaintext block is divided into two halves.
- The two halves of the data pass through rounds of processing and then combine to produce the ciphertext block.
- Each round has as inputs and derived from the previous round, as well as a subkey derived from the overall K.
- Any number of rounds could be implemented and 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.
- The Round Function F: F takes right-half block of previous round and a subkey as input.
- The output of the function is XORed with the left half of the data.
- Left and right halves are then swapped.

Feistel Cipher Decryption

- The process of decryption with a Feistel cipher is same as the encryption process.
- The ciphertext is input to the algorithm and the subkeys are used in reverse order. That is, subkey of the last round in encryption is used in the first round in decryption, second last in the second round, and so on.

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

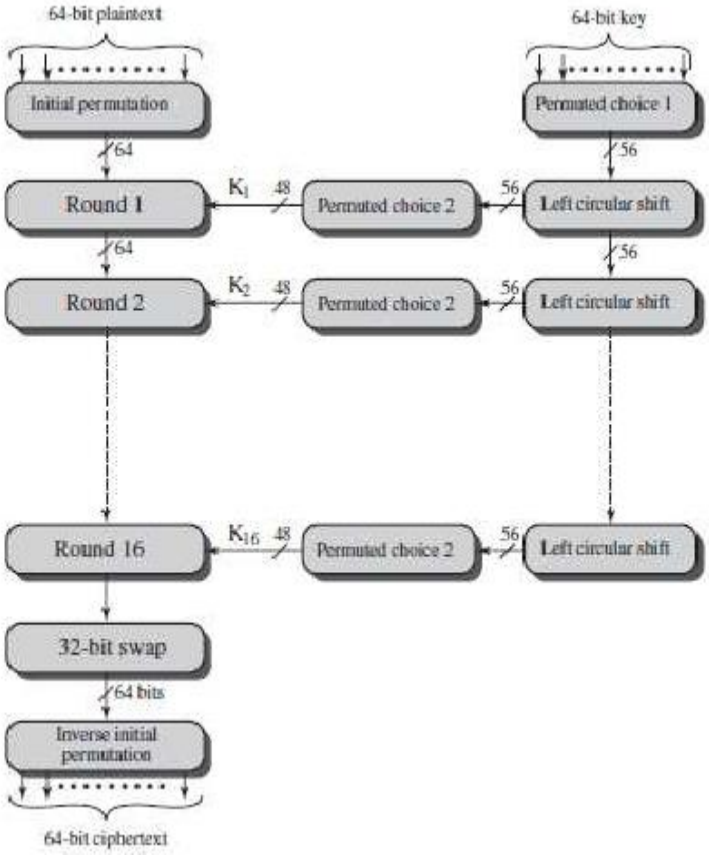
- **Block size**: Larger block sizes mean greater security but reduced encryption/decryption speed for a given algorithm. Traditionally, a block size of 64 bits is used which gives enough security without greatly affecting the speed.
- **Key size**: Larger key size means greater security but may decrease 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.
- **Sub key generation algorithm**: Greater complexity in this algorithm leads to greater difficulty of cryptanalysis.
- **Round function F**: Again, greater complexity generally means greater resistance to cryptanalysis.
- There are two other considerations in the design of a Feistel cipher:
- **Fast software encryption/decryption**: In many cases, encryption is embedded in applications implementation (as software). 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 crypt analyze, there is great benefit in making the algorithm easy to analyze. That is, if The algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a high level of assurance as to its strength.

The Data Encryption Standard

- SDES encrypts 64-bit blocks using a 56-bit key and produces a 64-bit ciphertext.
- Same steps, with the same key, are used to reverse the encryption with the order of the keys reversed.
- The DES is widely used.

DES Encryption

- The DES encryption is shown in the figure below:



- Encryption function has two inputs: the plaintext to be encrypted and the key.
- The processing of the plaintext proceeds in three phases.
  - The 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input.
  - The permuted output is then passed through sixteen rounds of the same function, which involves both permutation and substitution functions. The left and right halves from the last round are swapped to produce preoutput.
  - The preoutput is passed through a permutation that is the inverse of the initial permutation function, to produce the 64-bit cipher text.
- The right-hand portion of the figure shows the way in which the 56-bit key is used.
  - Initially, the key is passed through a permutation function.
  - Then, a sub key (k_i) is produces for each of the sixteen rounds by the combination of a left circular shift and a permutation.
  - The permutation function is the same for each round, but a different sub key is produced because of the repeated shifts of the key bits.

Initial Permutation (IP) and Inverse Initial Permutation (IP^{-1})

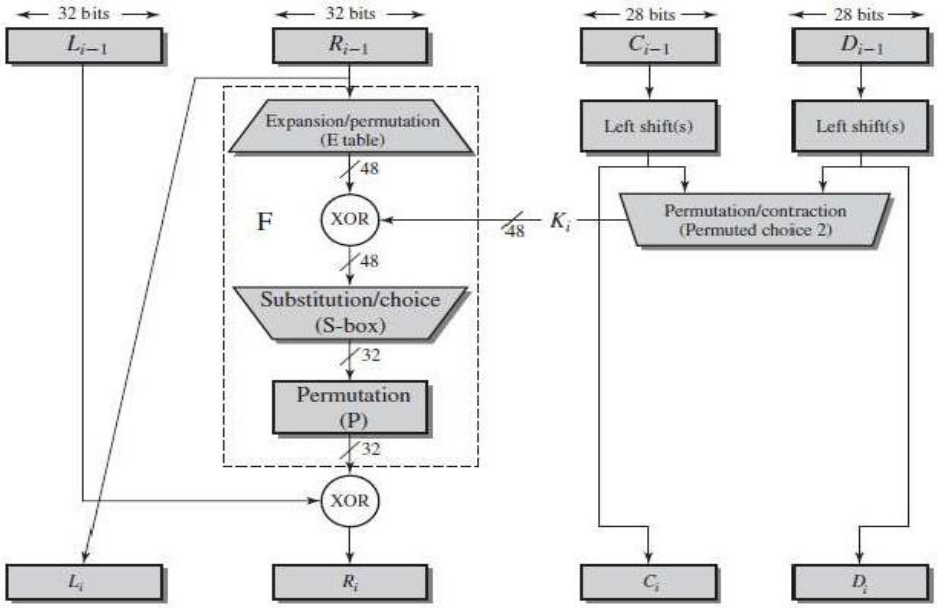
IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

IP^{-1}							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

- The initial permutation and its inverse are defined by tables.
- The tables are to be interpreted as follows.
  - The input to a table consists of 64 bits numbered from 1 to 64.
  - The 64 entries in the permutation table contain a permutation of the numbers from 1 to 64.
  - Each entry in the permutation table indicates the position of an input bit in the output.
- Inverse permutation table nullifies the effect of initial permutation.

Details of Single Round

- Figure shows the internal structure of a single round.



- The left and right halves are treated as separate 32-bit quantities, labeled L (left) and R (right).
- The overall processing at each round can be summarized as:

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

Expansion (E)

- The 32-bit input is first expanded to 48 bits.
  - Bits of input are split into groups of 4 bits.

- Each group is written as groups of 6 bits by taking the outer bits from the wo adjacent groups. For example

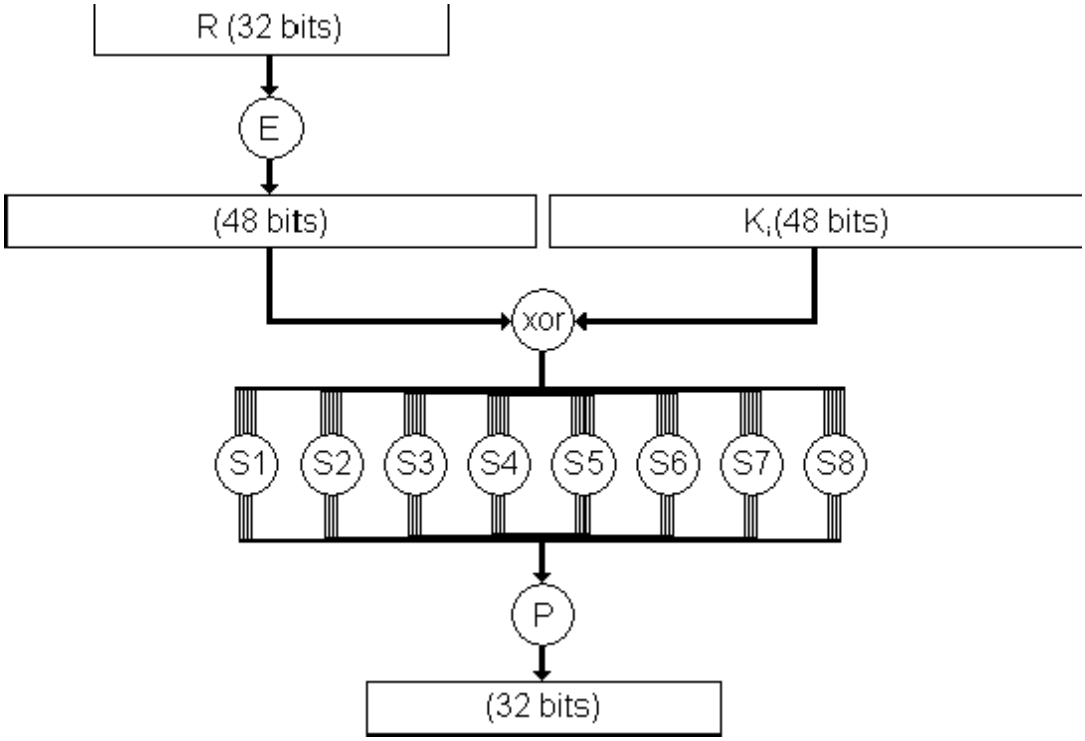
  ... efgh ijkl mnop ... is expanded to
  ... defghi hijklm lnnopq ...

32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

- The resulting 48 bits are XORed with K_i.

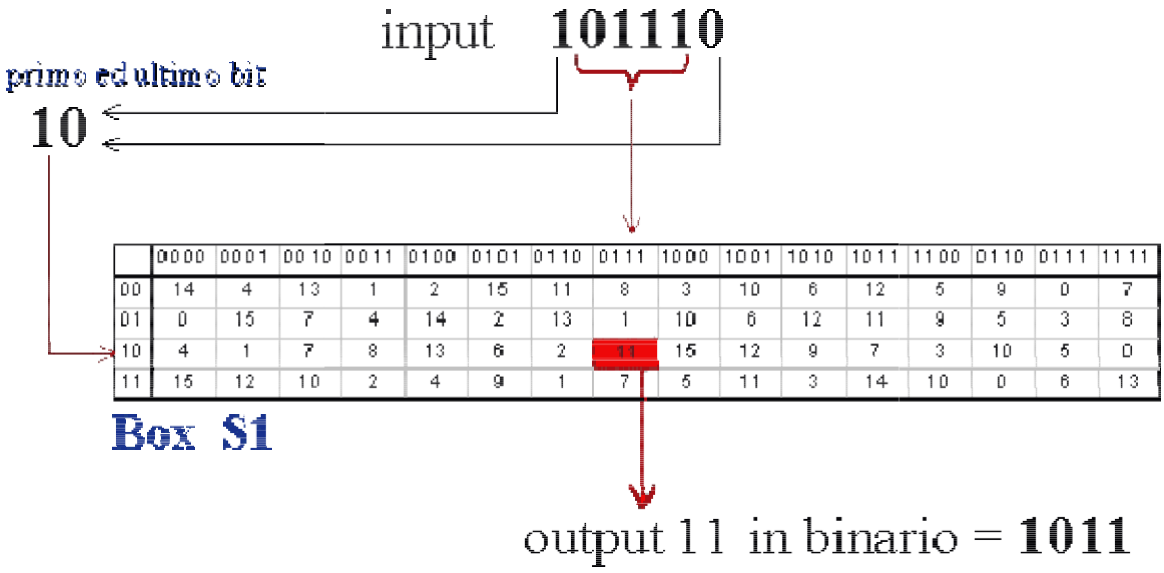
Substitution (S-Box)

- This 48-bit result is input to S-Boxes that perform a substitution on input and produces a 32-bit output.
- It is easy to understand S-Box by following figure.



- DES consists of a set of eight S-boxes.
- Each S-Box takes 6 bits as input and produces 4 bits as output.
- The first and last bits of the input to box form a 2-bit binary number which gives the binary value of row number.
- The middle four bits select one of the sixteen columns.

- The decimal value in the cell selected by the row and column is then converted to its 4-bit binary number to produce the output.
- For example, in S1, for input 101110, the row is 10 (row 2) and the column is 0111 (column 7).The value in row 2, column 7 is 11, so the output is 1011.



	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	0110	0111	1111
00	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
01	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
10	4	1	7	8	13	6	2	11	15	12	9	7	3	10	5	0
11	15	12	10	2	4	9	1	7	5	11	3	14	10	0	6	13

Permutation (P)

- The result is again permuted using a permutation table.

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Key Generation

- A 64-bit key is used as input to the algorithm while only 56 bits are actually used. Every eighth bit is ignored. Sub-keys at each round are generated as given below:
  - The key is first permuted using a table named Permuted Choice One.
  - The resulting 56-bit key is divided into two 28-bit quantities, C_0 and D_0. At each round, C_{i-1} and D_{i-1} are separately subjected to a circular left shift of 1 or 2 bits, as governed by a table.
  - These shifted values are forwarded to the next round. They are also input to a permutation table-Permuted Choice Two.
  - The table produces a 48-bit output that serves as the round key k_i.

(a) Input Key

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(b) Permuted Choice One (PC-1)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

(c) Permuted Choice Two (PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

(d) Schedule of Left Shifts

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES Decryption

- Decryption in DES is same as encryption, except that the sub keys are used in reverse order.

Strength of DES

The Use of 56-Bit Keys

- DES has been developed from LUCIFER which used 128-bit keys.
- As a result, DES with only 56-bit key-length is considered insecure and devices have been proposed time and again showing that DES is no longer secure.

The Nature of the DES

- The only non-linear part of DES is the S-Boxes, design of which was not made public.
- If someone is able to find weakness in S-Box, then attack on DES is possible.
- Characteristics of the algorithm can be exploited as the algorithm is based on linear functions.

Algorithm Timing Attacks

- In this type of attack, the attacker exploits the fact that any algorithm takes different amount of time for different data.

A DES Example

- Let see example of DES and consider some of its implications. Although you are not expected to duplicate the example by hand, you will find it informative to study the hex patterns that occur from one step to the next.

Plaintext:	02468aceeca86420
Key:	0f1571c947d9e859
Ciphertext:	Da02ce3a89ecac3b

- **Result:** Above table shows plain text, key and cipher text when we apply all the steps of DES, we will get cipher text as shown.

- **The Avalanche Effect:** A desirable property of any encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in cipher text.
- In particular, a change in one bit of plaintext or one bit of the key should produce a change in many bits of the ciphertext. This is referred to as the avalanche effect.
- In DES 1-bit change in input will affect nearly 32 bit of output after all rounds.

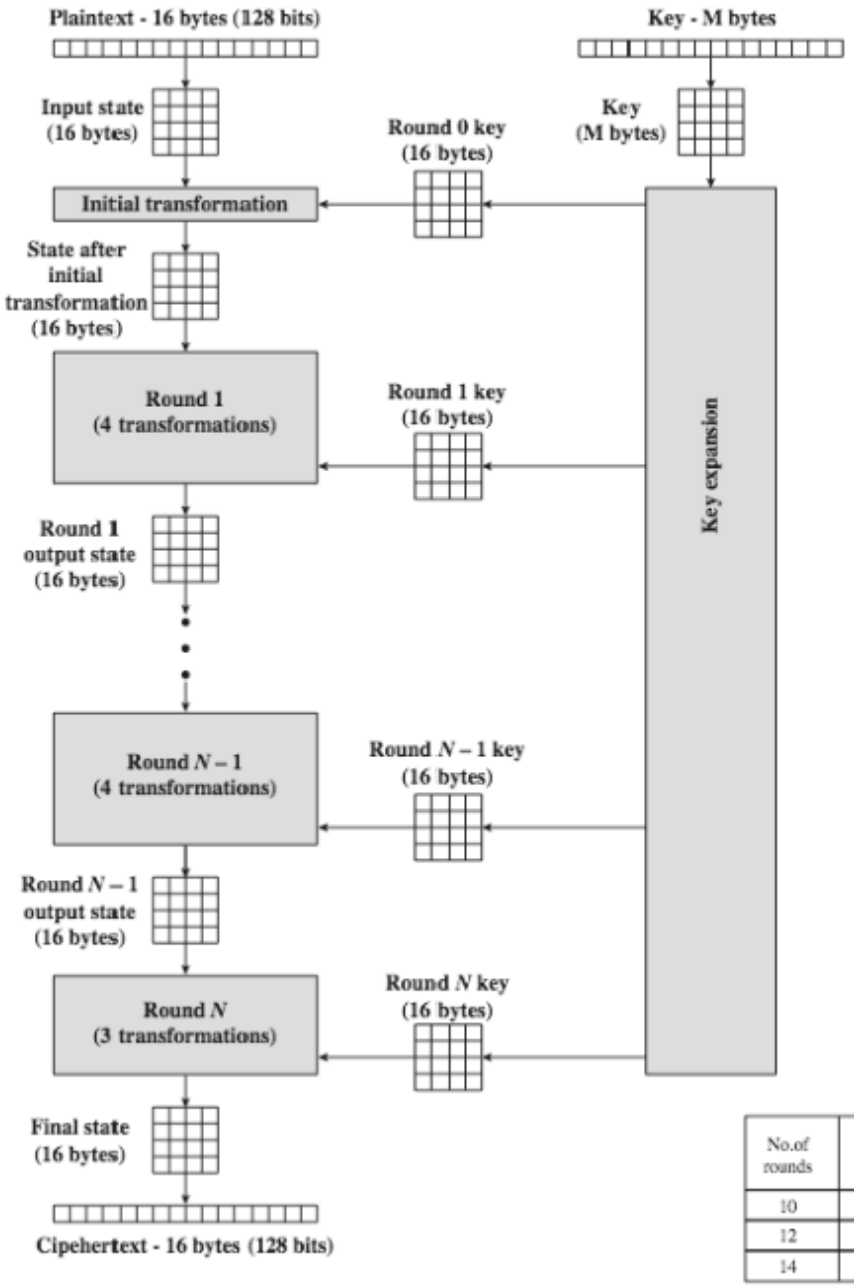
Block Cipher Design Principles

- The followed criteria need to be taken into account when designing a block cipher:
  - **Number of Rounds:** The greater the number of rounds, the more difficult it is to perform cryptanalysis, even for a weak function. The number of rounds is chosen so that efforts required to crypt analyze it becomes greater than a simple brute-force attack.
  - **Design of Function F**: F should be nonlinear and should satisfy strict avalanche criterion (SAC) and bit independence criterion (BIC).
  - **S-Box Design:** S-Box obviously should be non-linear and should satisfy SAC, BIC and Guaranteed Avalanche criteria. One more obvious characteristic of the S-box is its size. Larger S-Boxes provide good diffusion but also result in greater look-up tables. Hence, general size is 8 to 10.
  - **Key Generation Algorithm:** With any Feistel block cipher, the key is used to generate one sub key for each round. In general, sub keys should be selected such that it should be deduce sub keys from one another or main key from the sub key.

Advanced Encryption Standard (AES)

- The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES.
- A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow.
- The features of AES are as follows
  - Symmetric key symmetric block cipher
  - 128-bit data, 128/192/256-bit keys
  - Stronger and faster than Triple-DES
  - Provide full specification and design details
  - Software implementable in C and Java

AES Structure

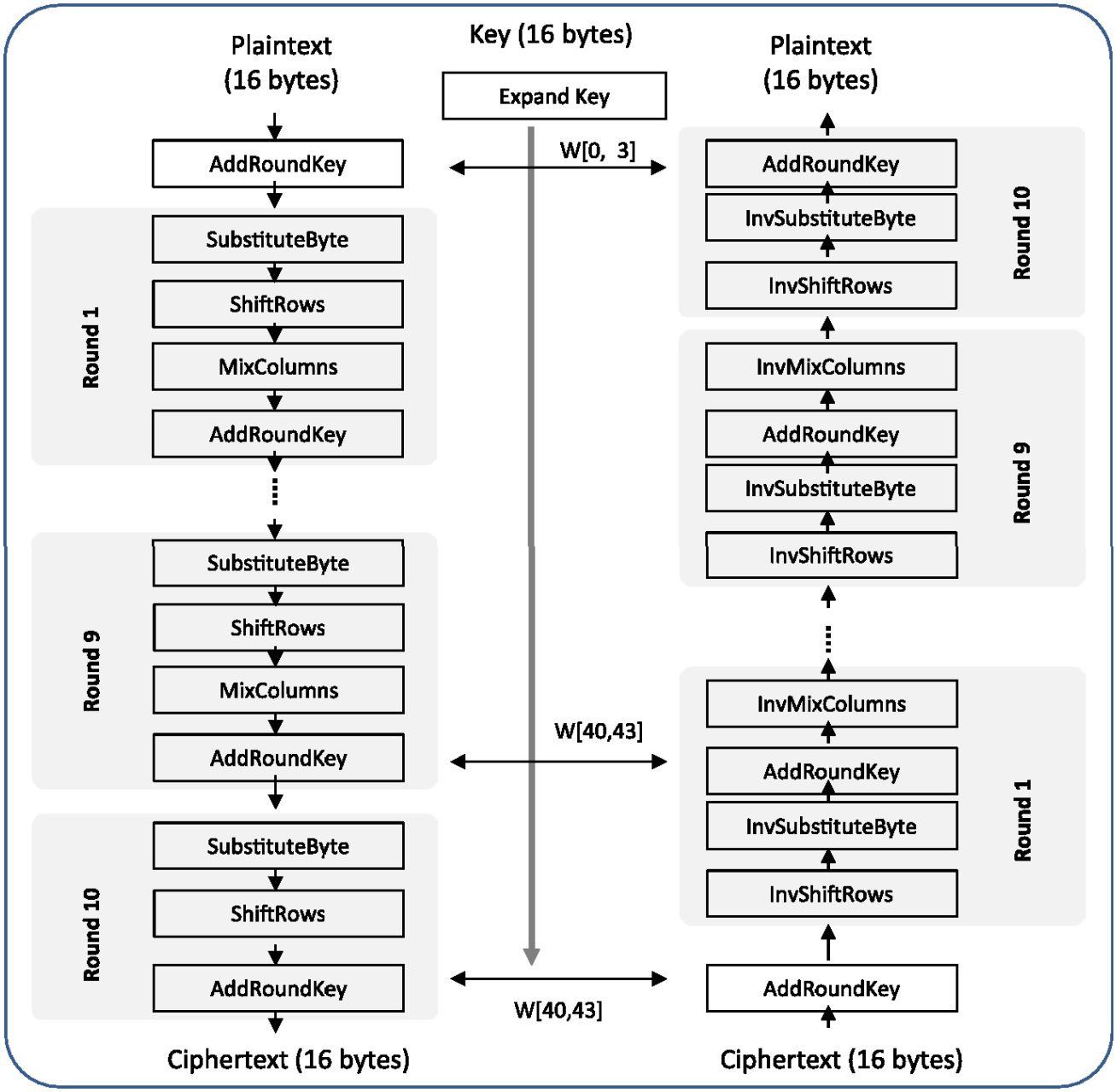


No.of rounds	Key Length (bytes)
10	16
12	24
14	32

- Figure shows the overall structure of the AES encryption process.
- Plain text block size is 128 bits (16 bytes).
- Key size depends on number of round 128, 192, or 256 bit as shown in table.
- Based on key size AES is named as AES-128, AES-192, or AES-256.
- The input 128-bit block, this block is arranged in the form of 4 X 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.
- There is an initial single transformation (AddRoundKey) before the first round which can be considered Round 0.
- The first N-1 rounds consist of four distinct transformation function: SubBytes, ShiftRows, MixColumns, and AddRoundKey, which are described subsequently.
- The final round contains only first three transformations of above round.

- Each transformation takes one or more 4 X 4 matrices as input and produces a 4 X 4 matrix as output.
- The key expansion function generates N+1 round key each of which is distinct 4 X 4 matrices. Each round key serves as one of the inputs to the AddRoundKey transformation in each round.

Detail Structure



- Figure shows detail encryption Decryption process of AES.
- Let's discuss Several comments about AES structure:
  1. It is not a Feistel structure. As we know in Feistel structure half of the data block is used to modify the other half of the data block and then the halves are swapped. While in AES we use full data block as a single matrix during each round.
  2. The key is expanded into an array of forty-four 32-bit words. And such four word (128-bit) serves as round key for each round.
  3. Four different stages are used one of permutation and three of substitution:
     - **SubBytes**: 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 bytes.
- **AddRoundKey**: A simple bitwise XOR of the current block with a portion of the expanded key.

4. The structure is quite simple for both encryption and decryption it begins with AddRoundKey, followed by nine rounds of all four stages, followed by tenth round of three stages.
5. Only AddRoundKey stage use key for this reason, the cipher begins and ends with an AddRoundKey stage. Any other stage, applied at the beginning or end, is reversible without knowledge of the key and so would add no security.
6. The AddRoundKey stage is in effect, a form of Vernam cipher and by itself would not be formidable. The other three stages together provide confusion, diffusion, and nonlinearity, but by themselves would provide no security because they do not use the key.
7. Each stage is easily reversible.
8. In AES, decryption algorithm is not identical to encryption algorithm.
9. Once it is established that all four stages are reversible, it is easy to verify that decryption does recover the plain text.
10. For making AES reversible the final round of both encryption and decryption consist of only three stages.

AES Transformation Function

Substitute bytes Transformation (Forward & Inverse)

- Substitute bytes transformation is simple table lookup. There is separate table for forward and inverse operation.
- 16 X 16 matrix of byte value called s-box that contains the permutation of all 256 8-bit values. Each individual byte of state is mapped into a new byte in the following way.
- The left most 4-bit of the byte are used as row number and right most 4-bit are used as column number. Now row and column number serve as index into the s-box to select unique 8-bit output value.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	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

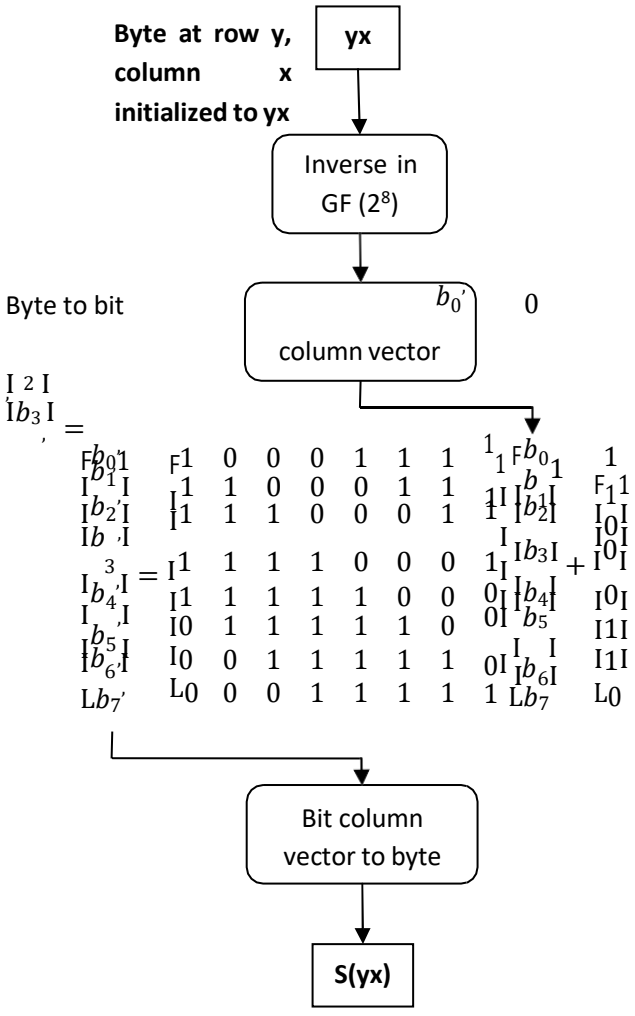
AES S-Box

- For example, hexadecimal value 68 is referred to row 6 and column 8 and value in table at that position is 45 so byte value 68 is replaced with 45.
- For inverse substitute byte procedure is same but S-box is different. Reverse of above example is shown in figure.

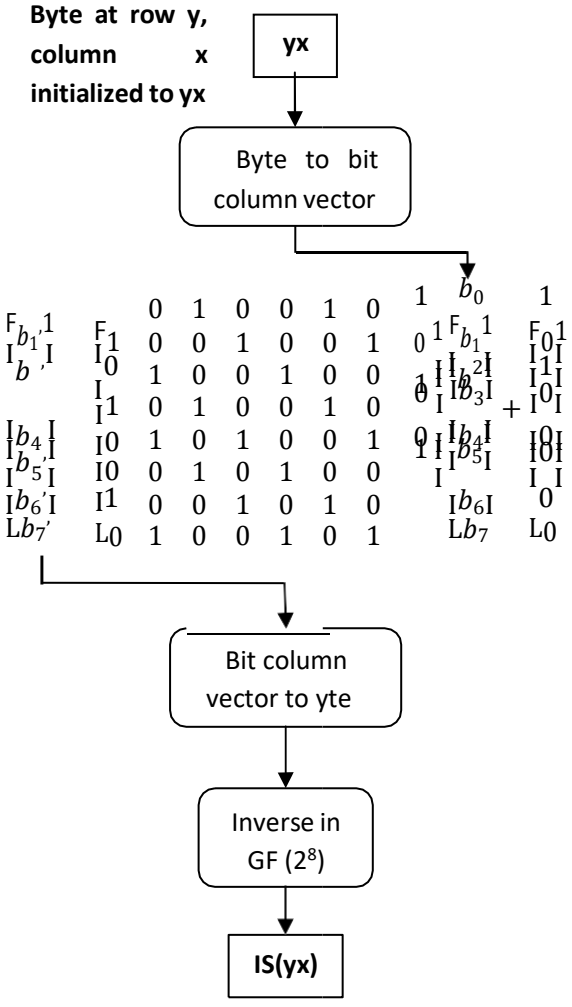
x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

AES Inverse S-Box

- S-box is constructed in the following fashion:



(a) Calculation of byte at row y, column x of S-box



(b) Calculation of byte at row y, column x of IS-

- Construction of S-box:
  1. Initialize the S-box with the byte values in ascending sequence row by row.
  2. Map each byte in the S-box to its multiplicative inverse in the finite field GF (2^8). The value {00} is mapped to itself.
  3. Consider that each byte in the S-box consist of 8 bits labeled (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0). Apply the transformation using matrix multiplication as shown in figure.
  4. Finally convert that bit column vector to byte.
- Construction of IS-box:
  1. Initialize the IS-box with the byte values in ascending sequence row by row.
  2. Consider that each byte in the IS-box consist of 8 bits labeled (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0). Apply the transformation using matrix multiplication as shown in figure.
  3. Convert that bit column vector to byte.
  4. Map each byte in the IS-box to its multiplicative inverse in the finite field GF (2^8).

ShiftRows Transformation (Forward & Inverse)

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 **forward shift row transformation** is performed as below:
  1. The first row of state is not altered.
  2. In second row we apply 1-byte circular left shift.
  3. In third row we apply 2-byte circular left shift.
  4. In fourth row we apply 3-byte circular left shift.
- In the **inverse shift row transformation,** we apply right circular rotation.
  1. The first row of state is not altered.
  2. In second row we apply 1-byte circular right shift.
  3. In third row we apply 2-byte circular right shift.
  4. In fourth row we apply 3-byte circular right shift.

MixColumns Transformation (Forward & Inverse)

- In the forward MixColumn transformation each byte of a column is mapped into a new value that is a function of all bytes in that column.
- The transformation can be defined by the following matrix multiplication on state:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} = \begin{bmatrix} S_{0,0}' & S_{0,1}' & S_{0,2}' & S_{0,3}' \\ S_{1,0}' & S_{1,1}' & S_{1,2}' & S_{1,3}' \\ S_{2,0}' & S_{2,1}' & S_{2,2}' & S_{2,3}' \\ S_{3,0}' & S_{3,1}' & S_{3,2}' & S_{3,3}' \end{bmatrix}$$

- In this case, the individual additions and multiplications are performed in GF (2^8).
- In the inverse MixColumn transformation procedure is same but matrix is different which is shown below.

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} = \begin{bmatrix} S_{0,0}' & S_{0,1}' & S_{0,2}' & S_{0,3}' \\ S_{1,0}' & S_{1,1}' & S_{1,2}' & S_{1,3}' \\ S_{2,0}' & S_{2,1}' & S_{2,2}' & S_{2,3}' \\ S_{3,0}' & S_{3,1}' & S_{3,2}' & S_{3,3}' \end{bmatrix}$$

AddRoundKey Transformation

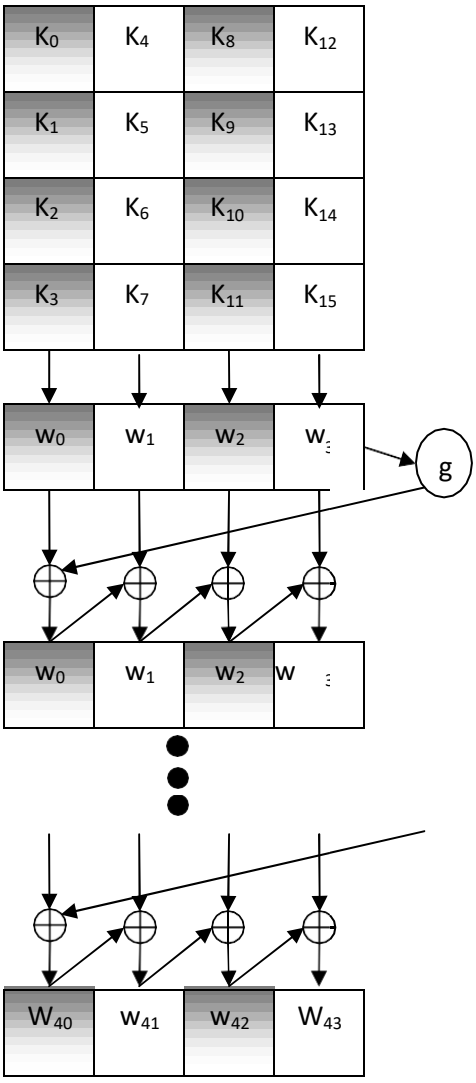
- In this transformation 128 bits state are bitwise XORed with the 128 bits of the round key.
- It is viewed as a byte level operation.
- Example

$$\begin{pmatrix} 47 & 40 & A3 & 4C \\ 37 & D4 & 70 & 9F \\ 94 & E4 & 3A & 42 \\ ED & A5 & A6 & BC \end{pmatrix} \oplus \begin{pmatrix} AC & 19 & 28 & 57 \\ 77 & FA & D1 & 5C \\ 66 & DC & 29 & 00 \\ F3 & 21 & 41 & 6A \end{pmatrix} = \begin{pmatrix} EB & 59 & 8B & 1B \\ 40 & 2E & A1 & C3 \\ F2 & 38 & 13 & 42 \\ 1E & 84 & E7 & D6 \end{pmatrix}$$

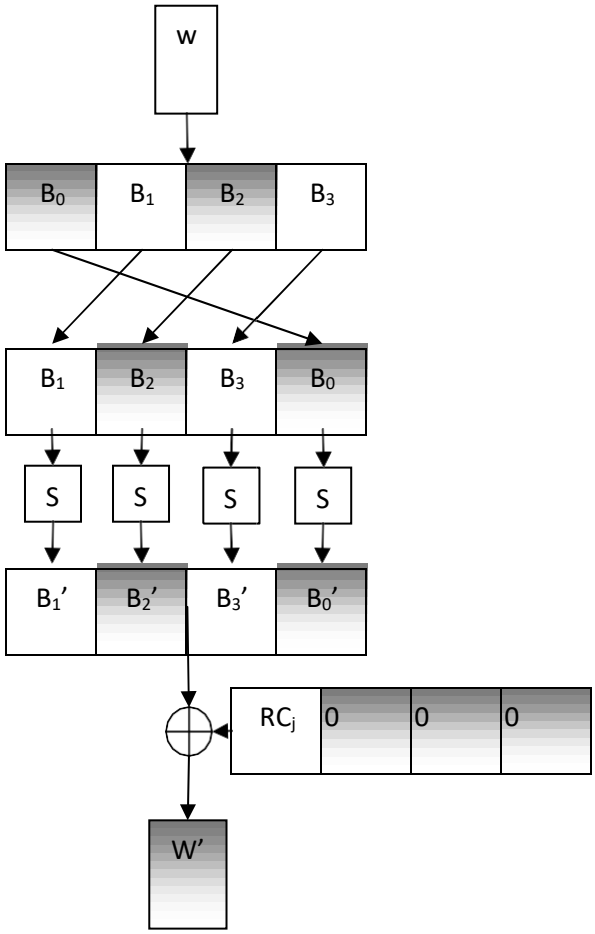
- Inverse of AddRoundKey is same because inverse of XOR is again XOR.

AES Key Expansion

- AES takes 16-byte key as input.
- As shown in figure below key expansion process is straight forward.



(a) Overall Key expansion algorithm.



(b) Function g.

- First of all, key is stored in 4X4 matrix in column major matrix as shown in figure.
- Each column combines to form 32-bit word.
- Then we apply function g to every fourth word that is w3, w7, w11 etc.

- Then X-OR operation is performed as shown in figure to obtain next four word. And this process continues till generation of all words.
- As shown in figure (b) internal structure of function g.
- First, we convert word to 4 byte.
- Then apply circular left shift operation.
- Then apply substitute byte operation using S-box which is same as S-box of AES encryption process.
- Then we apply X-OR operation with round constant which have least significant 3 byte as zero and most significant byte is depend on round number which is shown in table below.

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

- And output of this function is used for X-OR operation as shown in figure (a).

AES Example

- Let see example of AES and consider some of its implications.
- Although you are not expected to duplicate the example by hand, you will find it informative to study the hex patterns that occur from one step to the next.

Plaintext:	0123456789abcdeffedcba9876543210
Key:	0f1571c947d9e8590cb7add6af7f6798
Ciphertext:	Ff0b844a0853bf7c6934ab4364148fb9

- **Result:** Above table shows plain text, key and cipher text when we apply all the steps of AES, we will get cipher text as shown.
- **The Avalanche Effect:** A desirable property of any encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in cipher text.
- In particular, a change in one bit of plaintext or one bit of the key should produce a change in many bits of the ciphertext. This is referred to as the avalanche effect.
- In AES 1 bit change in input will affect nearly all bit of output after all rounds.

AES Implementation

Equivalent Inverse Cipher

- While implementing AES if we interchange the order of operation than it will affect the result or not is discussed here.
- If we interchange inverse shift row and inverse substitute byte operation than it will not affect and we get the same output.
- So, we can obtain two equivalent decryption algorithms for one encryption algorithm.
- As inverse shift row will change position of byte and it will not affect byte value. While inverse substitute byte will change byte value by table lookup and it not concern with byte position. So, we can interchange those two operations.
- If we interchange inverse mix column and add round key operation than it will affect and we do not get the same output.
- Both the operation will affect the value and so it cannot be interchange.

Implementation Aspects

- As in AES out of four three operation are byte level operation and it can be efficiently implemented on 8-bit processors.

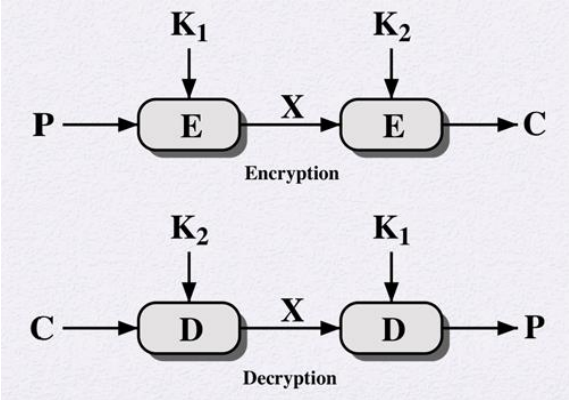
- Only mix column operation is requiring matrix multiplication which requires some storage space and also time consuming on 8-bit processor.
- To overcome it we can use table lookup to reduce time requirement.
- Also, we can implement it on 32-bit processors.
- In 32-bit processor we can use word by word operation and it much faster.

Multiple Encryption and Triple DES

- As we know that DES is venerable to brute-force attack, we are interested to find an alternative.
- One possible solution is to design completely new algorithm like AES.
- Second solution is to use DES multiple times.

Double DES

- The simplest form of multiple encryptions has two encryption stages and two keys and is known as Double DES.



- Given a plaintext P and two encryption keys K1 and K2, cipher text C is generated as: **C = E(K2, E(K1, P))**
- Decryption applies keys in reverse order: **P = D(K1, D(K2, C))**
- This scheme involves a key length of 56 * 2 = 112 bits, making Brute-Force attack impractical.
- However, other types of attacks are possible:

Reduction to a Single Stage

- If it is possible to find a key such that: **E(K2, E(K1, P)) = E(K3, P)** then double encryption, or any number of stages of multiple encryption with DES, would be useless.
- Because the result would be equivalent to a single encryption with a single 56- bit key.
- However, by the principle of reverse mapping, such a key is not possible.

Meet-In-The-Middle Attack

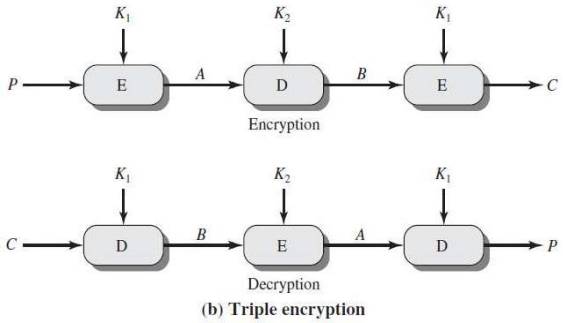
- This attack is based on the on the observation that if:

  $C = E(K_2, E(K_1, P))$, then

  $X = E(K_1, P) = D(K_2, C)$
- Given a known (P, C) pair, the attack proceeds as follows:
  - ✓ First, encrypt P for all 256 possible values of K1.
  - ✓ Store these results in a table and then sort the table by the values of X.
  - ✓ Decrypt C using all 256 possible values of K2.
  - ✓ Check the result against the table for a match after every decryption.
  - ✓ If a match occurs, then test the two resulting keys against a new known plaintext– ciphertext pair.
  - ✓ If the two keys produce the correct ciphertext, accept them as the correct keys.
  - ✓ For any given plaintext, 248 false alarms are possible since there are only 264 ciphertext values whereas 2112 key values.
  - ✓ Thus, the order of attack can be reduced to 248 instead of 2112.

Triple DES with Two Keys

- An alternative to the meet-in-the-middle attack is to use three stages of encryption with three or two different keys.



(b) Triple encryption

- The function follows an encrypt decrypt-encrypt (EDE) sequence.

  $C = E(K1, D(K2, E(K1, P)))$
  $P = D(K1, E(K2, D(K1, C)))$
- 3DES with two keys is a relatively popular alternative to DES.
- Currently, there are no practical cryptanalytic attacks on 3DES.
- Brute-force key search on 3DES is on the order of 2112 and the cost of differential cryptanalysis also has an exponential growth, compared to single DES.
- Several proposed attacks (though impractical) on 3DES are:

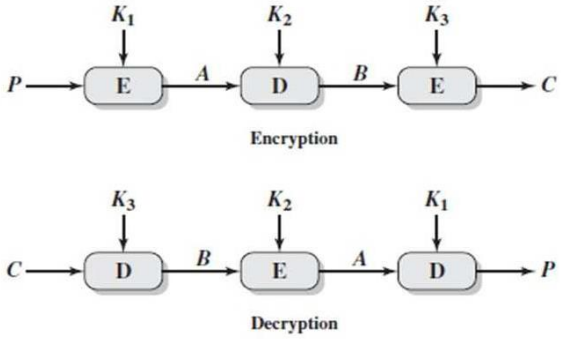
Chosen-Plaintext Attack

- Find plaintext values that gives A = 0.
- Then, use the meet-in-the- middle attack to determine the two keys.
- However, this attack requires 256 chosen plaintext-cipher text pairs which is impractical.

Known-Plaintext Attack

- This method does not require chosen plaintext-cipher text pairs but requires more effort.
- The attack is based on the observation that if an attacker knows A and C, then the problem reduces to that of an attack on double DES.
- The attack is as follows:
  - ✓ The attacker obtains n(P, C) pairs places them in Table 1 sorted on the values of P.
  - ✓ For an arbitrary value a for A, calculate the plaintext value that produces:
    Pi = D(i, a)
  - ✓ For each Pi that matches an entry in Table 1, create an entry in Table 2 that contains value of K1 and b that is obtained by decrypting the corresponding cipher text from Table 2.
    $B = D(i, C)$
  - ✓ Table 2 contains a number of candidate values of Ki. Now, for each of the 256 possible values of K2, calculate the second intermediate value for our chosen value of a:
    Bj = D(j, a)
  - ✓ At each step, look up Bj in Table 2. If there is a match, then the corresponding key i from Table 2 plus this value of j are candidate values for the unknown keys (K1, K2).
  - ✓ Test each candidate pair of keys on a few other plaintext–cipher text pairs. If a pair of keys produces the desired cipher text, the task is complete.
  - ✓ If no pair succeeds, repeat from step 1 with a new value of a.

Triple DES with Three Keys

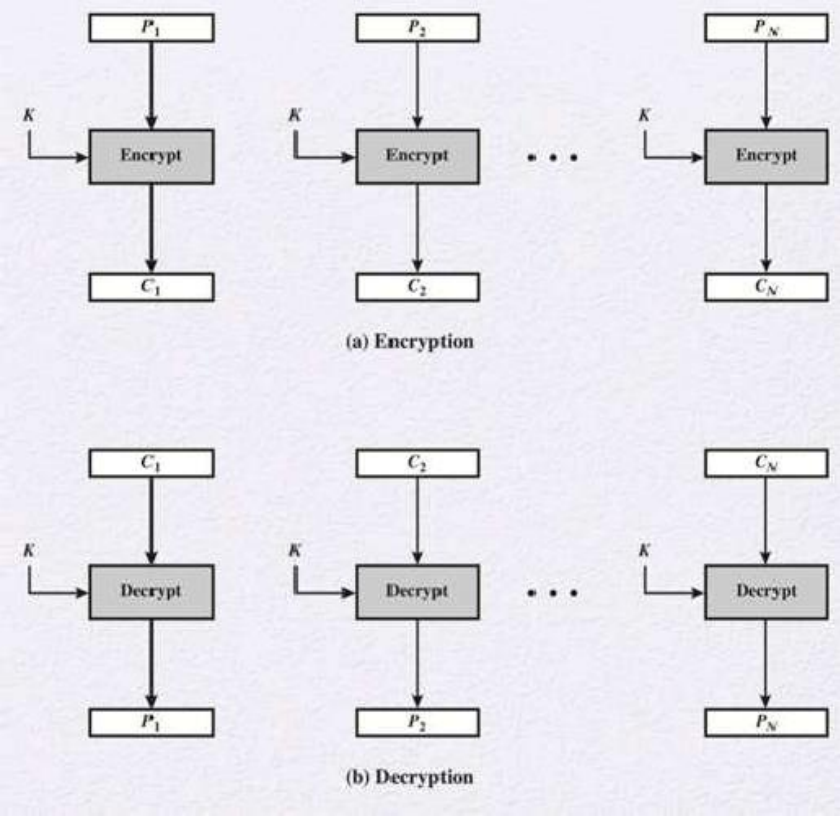


- Although the attacks just described appear impractical, anyone using two-key 3DES may feel some concern.
- In that case, three-key 3DES is the preferred alternative.
- Three-key 3DES has an effective key length of 168 bits and is defined as:
  C = E(K3, D(K2, E(K1, P)))
- Backward compatibility with DES is provided by putting K3 = K1 or K1 = K3.
- A number of Internet-based applications have adopted three-key 3DES, including PGP and S/MIME.

Modes of Operations

There are 5 modes of operation which are listed below.

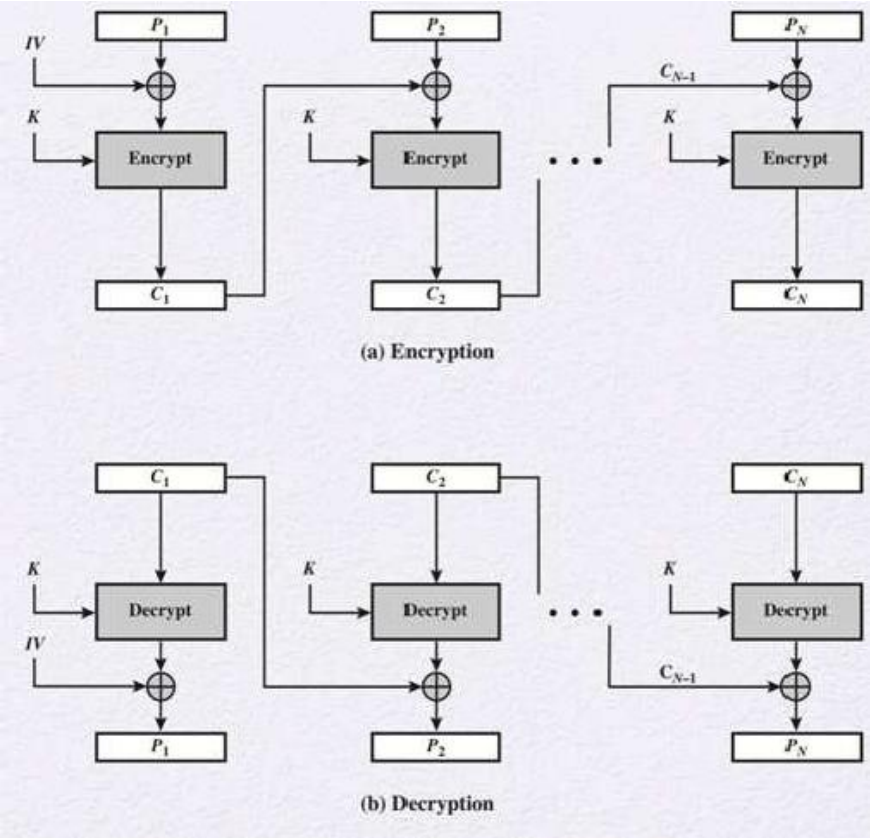
1. Electronic Codebook mode (ECB)



- This is the simplest mode in which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key.

- The term codebook is used because, for a given key, there is a unique ciphertext for every -bit block of plaintext.
- Therefore, we can imagine a huge codebook in which there is an entry for every possible b-bit plaintext showing its corresponding ciphertext.
- For a message longer than b bits, the procedure is simply to break the message into b-bit blocks, padding the last block if necessary.
- Decryption is performed one block at a time, always using the same key.
- For lengthy messages, ECB mode may be not secure. If the message has repetitive elements, then these elements can be identified by the analyst.
- Thus, the ECB method is ideal for a short amount of data, such as an encryption key.

2. Cipher Block Chaining Mode (CBC)



- To overcome the security deficiencies of ECB, a technique is needed in which the same plaintext block, if repeated, produces different cipher text blocks.
- A simple way to satisfy this requirement is the cipher block chaining (CBC) which is shown in the figure.
- In this mode, the input to the encryption algorithm is the X-OR of the current plaintext block and the preceding ciphertext block; the same key is used for each block.
- The input to the encryption function for each plaintext block has no fixed relationship to the plaintext block.
- Therefore, repeating patterns will not produce same ciphertext.
- The last block is padded to a full b bits if it is a partial block.
- For decryption, each cipher block is passed through the decryption algorithm. The result is X-ORed with the preceding ciphertext block to produce the plaintext block.
- The expressions for CBC are:

Encryption:

$C\mathrm{j} = E(K, [C\mathrm{j} - 1 \oplus P\mathrm{j})$

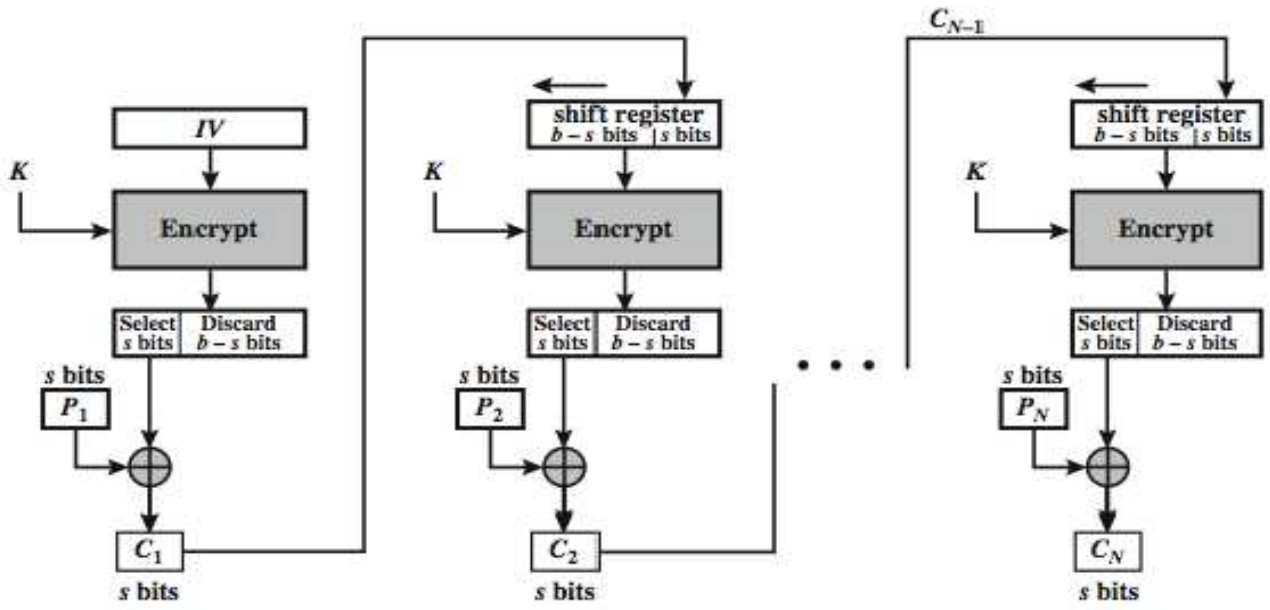
Decryption:

$D(K, C\mathrm{j}) = D(K, E(K, [c\mathrm{j} - 1 \oplus P\mathrm{j}]))$

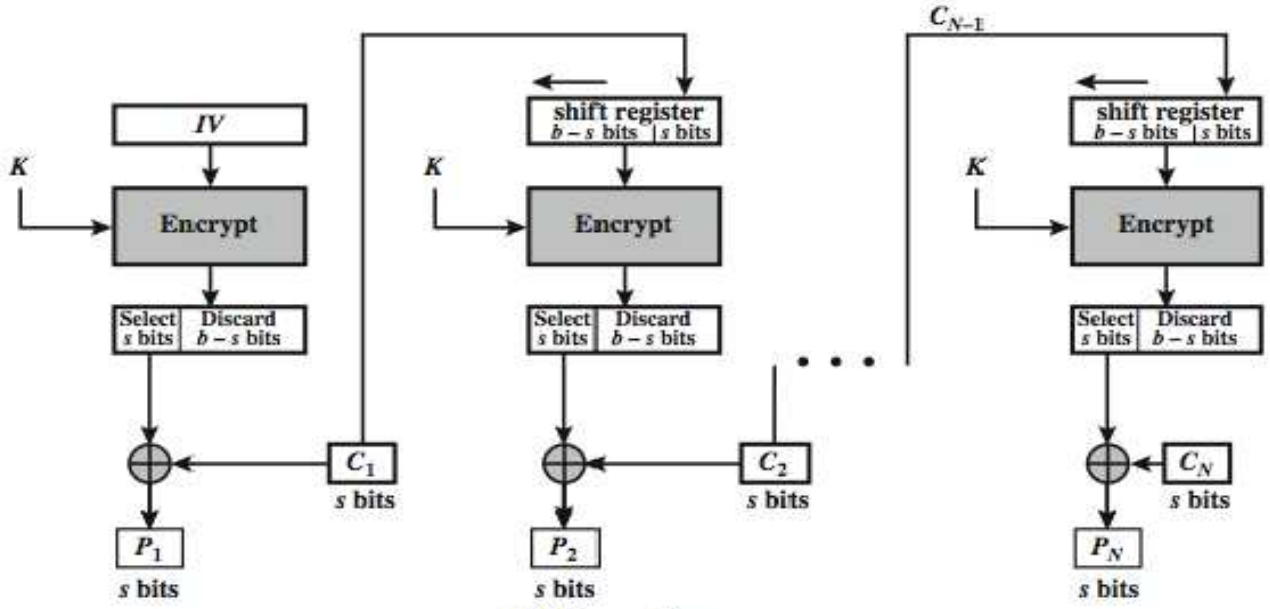
$D(K, C\mathrm{j}) = C\mathrm{j} - 1 \oplus P\mathrm{j}$

$C\mathrm{j} - 1 \oplus D(K, C\mathrm{j}) = C\mathrm{j} - 1 \oplus C\mathrm{j} - 1 \oplus P\mathrm{j} = P\mathrm{j}$

3. Cipher Feedback Mode (CFB)



(a) Encryption

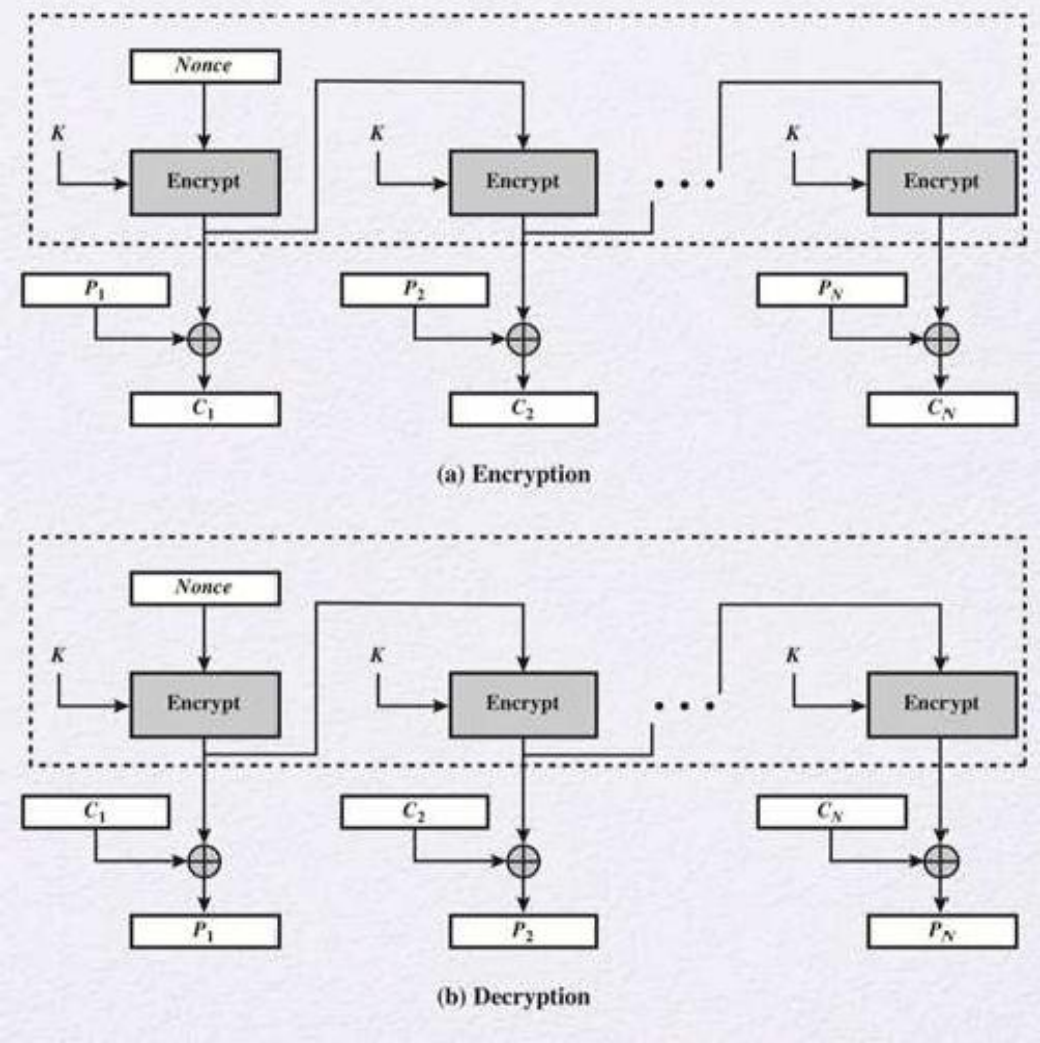


(b) Decryption

- DES is a block cipher, but it may be used as a stream cipher if the Cipher Fee back Mode (CFB) or the Output Feedback Mode (OFB) is used. CFB scheme is depicted below.
- A stream cipher eliminates the need to pad a message to be an integral number of blocks.
- It also can operate in real time.

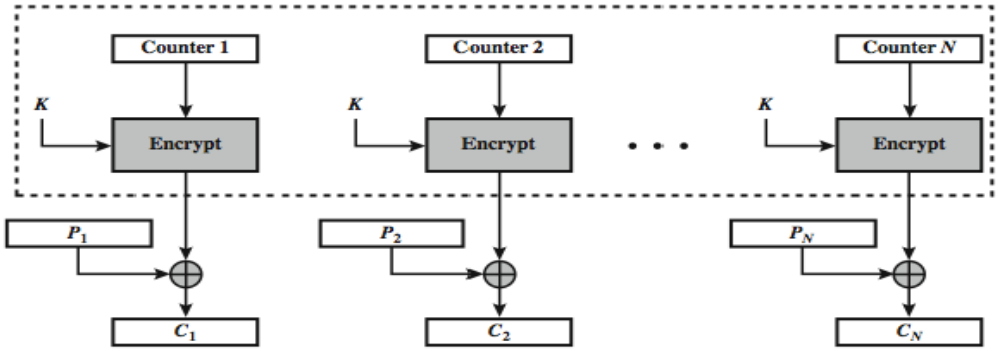
- **'s'** bits is the size usually selected by the user, most of time it is 8 bits.
- In this case, rather than block of 64 bits, the plaintext is divided into segments of s bits.
- **Encryption:** The input to the encryption function is a 64-bit shift register that is initially set to some initialization vector (IV).
- The leftmost (most significant) s bits of the output of the encryption function are X-ORed with the first segment of plaintext P1 to produce the first unit of ciphertext C1, which is then transmitted.
- In addition, the contents of the shift register are shifted left by **s** bits and C1 is placed in the rightmost **s** bits of the shift register.
- This process continues until all plaintext units have been encrypted.
- **Decryption:** The same scheme is used except that the received ciphertext unit is X-ORed with the output of the encryption function to produce the plaintext unit.
- The disadvantage of this scheme is that bit error in one ciphertext propagates to next stage also.

4. Output Feedback Mode (OFB)

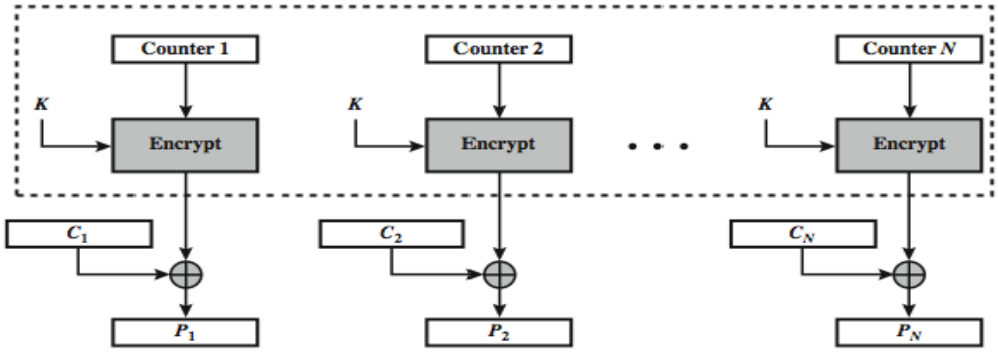


- The Output Feedback Mode (OFB) is similar in structure to that of CFB:
- The difference between CFB and OFB is that in OFB the output of the encryption function is fed back to the shift register in OFB, whereas in CFB the ciphertext is fed to the shift register.
- The other difference is that the OFB mode operates on full blocks of plaintext and ciphertext, not on 's' bit subset.
- One advantage of the OFB method is that bit errors in transmission do not propagate.
- The disadvantage of OFB is that it is more vulnerable to a message stream modification attack than CFB.

5. Counter Mode (CTR)



(a) Encryption



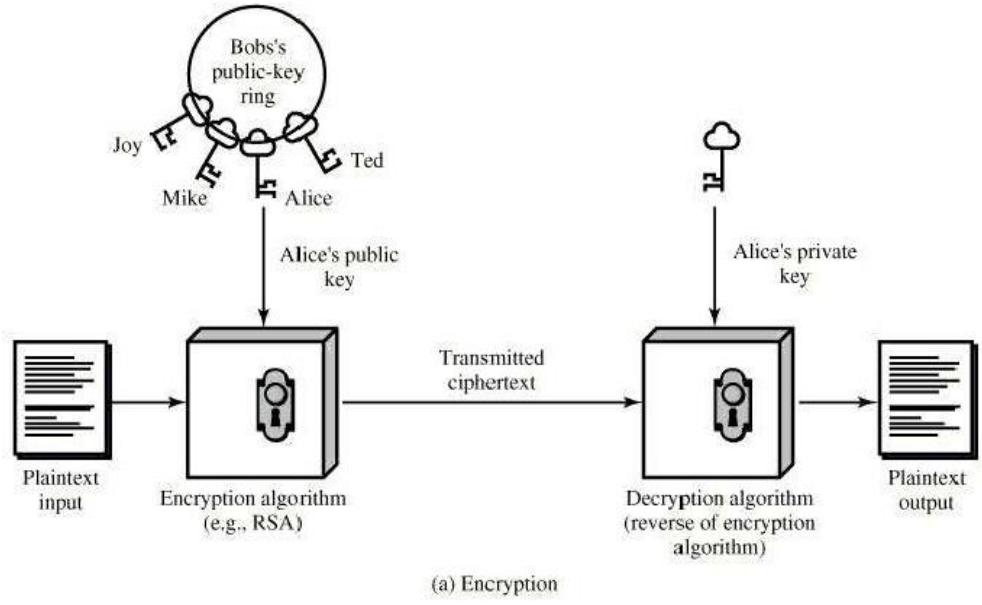
(b) Decryption

- In this mode, a counter equal to the plaintext block size is used.
- The only requirement is that the counter value must be different for each plaintext block that is encrypted.
- Typically, the counter is initialized to some value and then incremented by 1 for each subsequent block (modulo 2^b, where **b** is the block size)
- Counter Mode works as follows:
- **Encryption**: The counter is encrypted and then X-ORed with the plaintext block to produce the cipher text block; there is no chaining.
- **Decryption**: The same sequence of counter values is used. Each encrypted counter is X-ORed with a cipher text block to recover the corresponding plaintext block.
- CTR has following advantages:
  - **Hardware efficiency:** In this mode, encryption (or decryption) can be done in parallel on multiple blocks of plaintext or cipher text. For the chaining modes, the algorithm must complete the computation on one block before beginning on the next block.
  - **Software efficiency**: Similarly, because of the opportunities for parallel execution in CTR mode, processors that support parallel features, such as aggressive pipelining, multiple instruction dispatch per clock cycle, large number of registers can be effectively utilized.
  - **Preprocessing:** The execution of the encryption algorithm does not depend on input of the plaintext or cipher text. Therefore, preprocessing can be used to prepare the output of the encryption boxes which can be fed into the X-OR functions when the plaintext or cipher text input is presented.
  - **Random access:** The **i^{th}** block of plaintext or cipher text can be processed in random-access fashion. With the chaining modes, block cannot be computed until **i– 1** prior block is computed.
  - **Provable security:** It can be shown that CTR is as secure as the other modes.
  - **Simplicity:** CTR mode requires only the implementation of the encryption algorithm and not the decryption algorithm and has a very simple implementation.
- This mode is used in ATM (asynchronous transfer mode) and IPsec (IP security) nowadays.

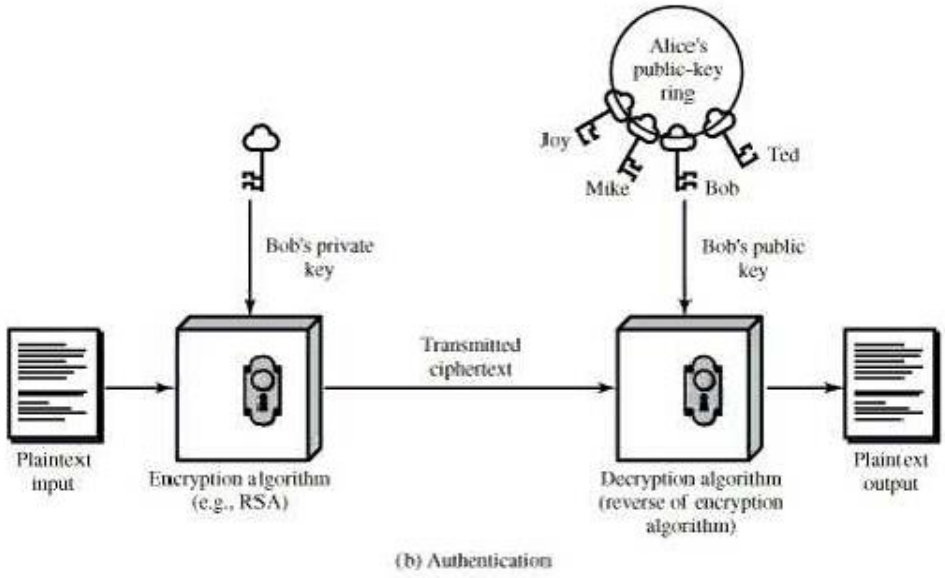
Define Public Key Cryptography

Public-key cryptography is a cryptographic system that uses two separate keys, one of which is secret and the other one is public. The algorithms used for public key cryptography are based on mathematical functions.

Public Key Cryptosystem



(a) Encryption



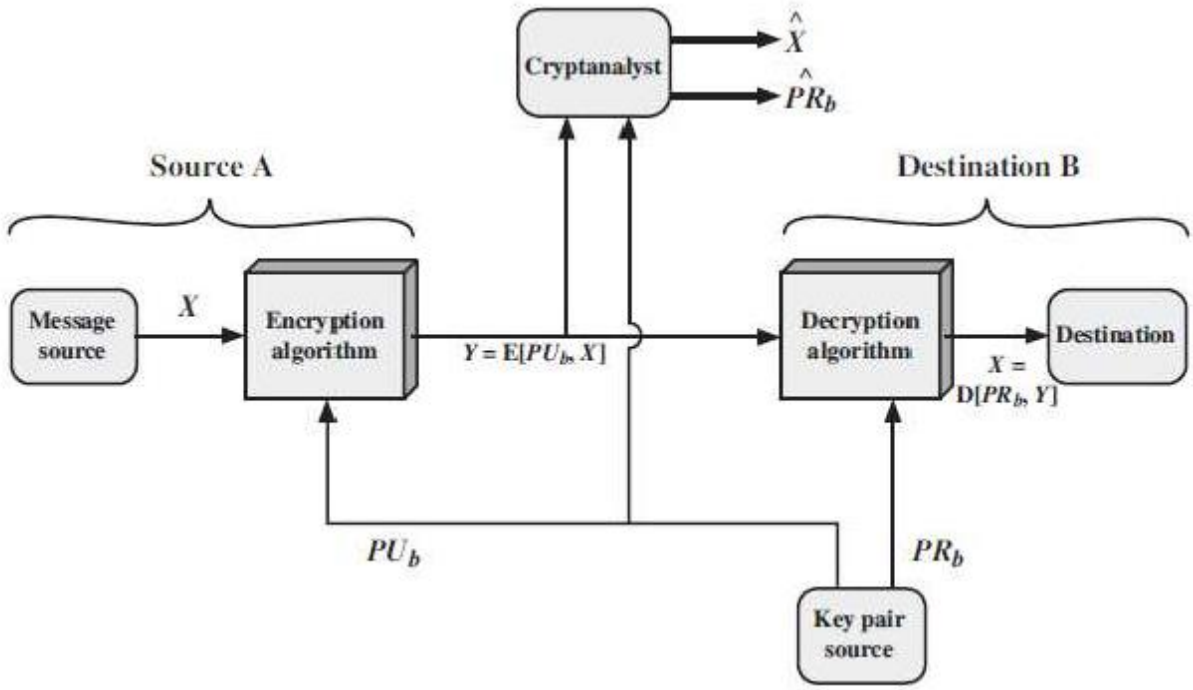
(b) Authentication

- A public-key encryption scheme has six parts.
  - **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.
  - **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key

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

- Any cryptosystem is designed to meet following goal
  1. Secrecy (Encryption)
  2. Authentication
- Now we will discuss how it is maintained in public key cryptosystem

Public Key Cryptosystem: Secrecy



Encryption using public key cryptography

- The essential steps are the following.
  - Each user generates a pair of keys to be used for the encryption and decryption of messages.
  - Each user places one of the two keys in a public register or other accessible file. This is the public key. The other key is kept private.
  - If A wishes to send a confidential message to B, A encrypts the message using B's public key.
  - When B receives the message, it decrypts it using the private key. No other recipient can decrypt the message because only B knows B's private key.
  - As long as a user's private key remains protected and secret, incoming communication is secure.
  - At any time, a system can change its private key and publish the companion public key to replace its old public key.
- Suppose there is some source A that produces a message in plaintext, $X = [X_1, X_2, \ldots, X_M]$ and sends it to B.
- B generates a related pair of keys: a public key, PU_b, and a private key, PR_b. $PU_b$is publicly available and therefore accessible by A.
- With the message X and the encryption key PU_b as input, A forms the ciphertext $Y = [Y_1, Y_2, \ldots, Y_N]$:

$$Y = E(PU_b, X)$$

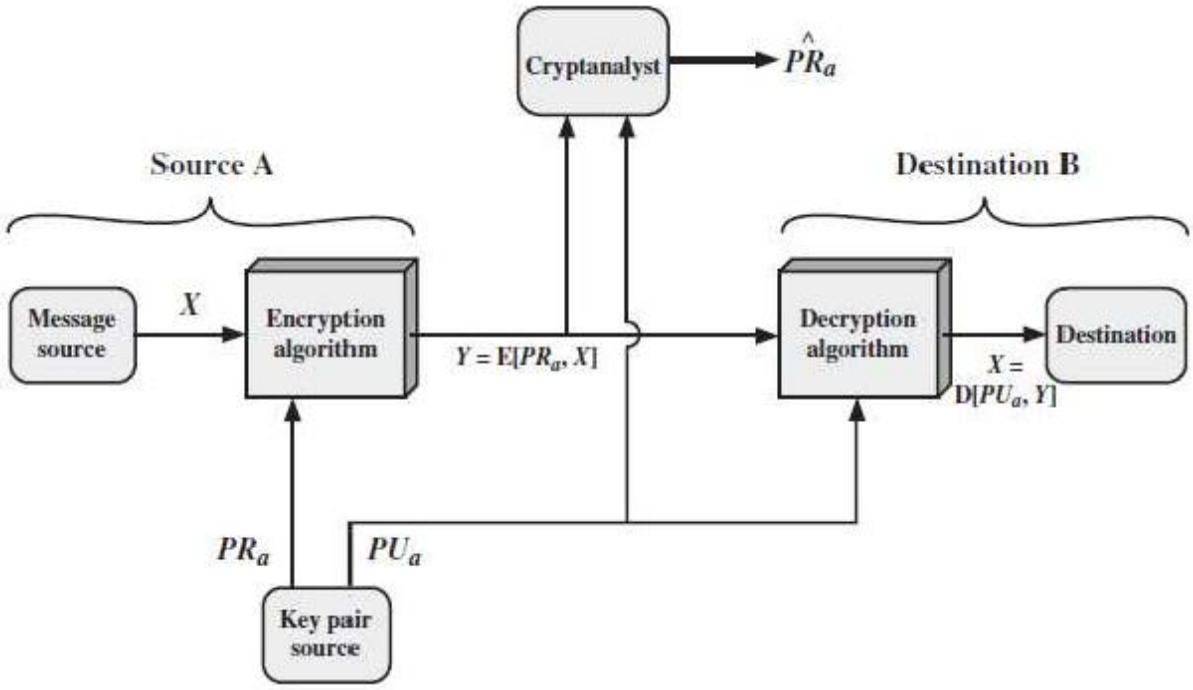
- The intended receiver, having the matching private key, is able to decrypt the message:

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

- An adversary, observing Y and having access to Pu_b only, may attempt to recover X and/or PR_b.

- If the adversary is interested only in this particular message, then the focus of effort is to recover X by generating a plaintext estimate. $\hat{X}$
- Whereas if the adversary is interested in being able to read future messages as well, then he attempts to recover PR_b by generating an estimate $\hat{PR_b}$

Public Key Cryptosystem: Authentication



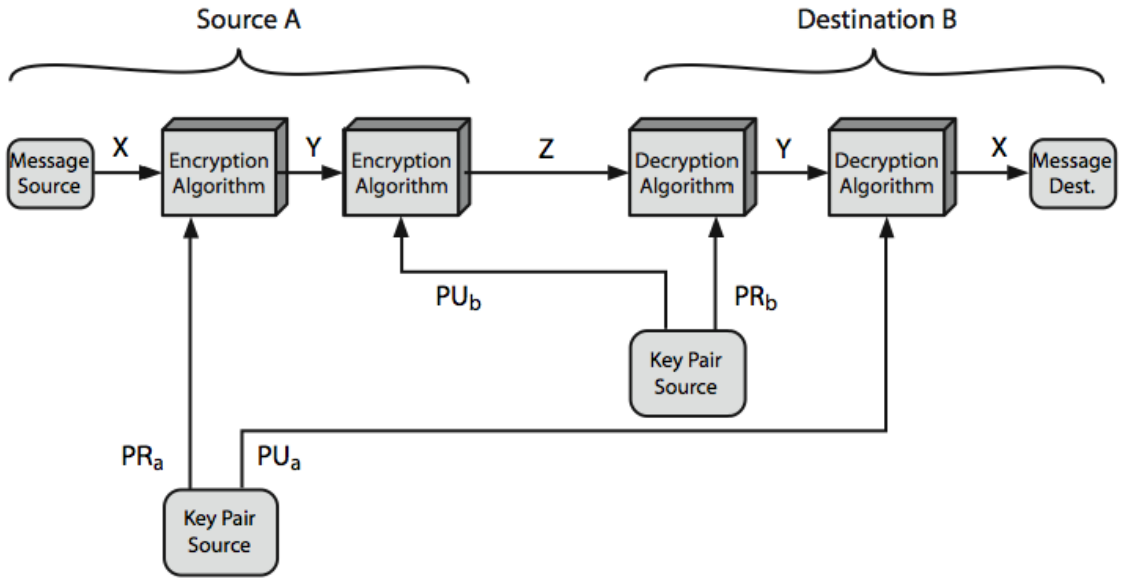
Authentication using public key cryptography

- However, the above scheme does not provide authentication of sender as, anyone having access to the public key can encrypt the message.
- Public-key encryption can be used to provide authentication in the following manner:
  - When A wishes to send a message to B where confidentiality is not needed but authentication is required, A encrypts the message using PR_a.
  - Anyone having access to PU_a can decrypt the message. However, one thing is sure that the message originated from A since no one except A could have encrypted the message using PR_a.
- A prepares a message to B and encrypts it using A's private key before transmitting it.

  Y = E (PRa, X)
- B can decrypt the message using A's public key.

  X = D (PUa, Y)
- 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.
- In addition, 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.
- However, the entire message needs to be stored to bring up in case of dispute.
- A more efficient way of achieving the same results is to encrypt a small block of bits that is a function of the document.
- Such a block, called an authenticator.
- It must have the property that it is infeasible to change the document without changing the authenticator.
- If the authenticator is encrypted with the sender's private key, it serves as a signature.

Public Key Cryptosystem: Authentication and Secrecy



- Authentication and Secrecy both can be achieved by combining above both techniques.
  - First sender A encrypt message X with private key of A.

    $$Y = E(PRa, X)$$
  - Then again A encrypt Y with public key of B.

    $$Z = E(PUb, Y)$$
  - Then send Z.
  - Only B can decrypt Z as it is encrypted with public key of B. So, it gives Secrecy.

    $$Y = D(PRb, Z)$$
  - Now Y can be decrypted with public key of A. So, it gives authentication.

    $$X = D(PUa, Y)$$
- So, by using public key cryptography we can achieve secrecy and authentication.

Applications of Public Key Cryptography

- Applications of public-key cryptosystems can be classified into three categories:
  1. **Encryption /decryption:** The sender encrypts a message with the recipient's public key.
  2. **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.
  3. **Key exchange:** Two sides cooperate to exchange a session key. Several different methods are possible.
- Some algorithm like RSA and Elliptic Curve are suitable for all three applications whereas others can be used for one or two of these applications.

Requirements for Public Key Cryptography

- Requirements that public key algorithms must fulfill are:
  - It is computationally easy for a party B to generate a key pair.

- It is computationally easy for a sender A, knowing the public key and the message **M**, to generate the corresponding ciphertext and for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message.
- It is computationally infeasible for an adversary, knowing the public key, $\mathbf{PU_b}$, to determine the private key, $\mathbf{PR_b}$.
- It is computationally infeasible for an adversary, knowing the public key, $\mathbf{PU_b}$, and a ciphertext, **C**, to recover the original message, **M.**
- The two keys can be applied in either order:

$$\mathbf{M = D[PU_b, E(PR_b,M)] = D[PR_b, E(PU_b,M)]}$$

- These are requirements that only a few algorithms have been able to fulfill. Some of these are RSA, elliptic curve cryptography, Diffie-Hellman, & DSS.

Public Key Cryptanalysis

Brute force attack

- This attack includes trying all the alternate keys until the correct key is found.
- Counter measure to this is use large keys.
- However, public-key systems depend on the use of some sort of invertible mathematical function which is really time-consuming and increases overhead.
- Thus, there is a tradeoff. The key size must be large enough to make brute-force attack impractical but small enough for practical encryption and decryption.
- Secure keys are long enough to make encryption decryption really slow.
- Hence, public-key encryption is currently confined to key management and signature applications

Computation of private key from public key

- In this attack, some characteristics of algorithm are exploited to calculate the private key from public key.
- This attack needs many known or chosen plaintext-ciphertext pairs.
- To date it has not been mathematically proven that this form of attack is infeasible for a particular algorithm. Thus, any given algorithm is suspect.

Probable message attack

- In this attack, the opponent has some idea about the plaintext and he uses this information to find the private key.
- Suppose that a message consists only a 56-bit DES key.
- An adversary could encrypt all possible 56-bit DES keys using the public key and could discover the encrypted key by matching the transmitted ciphertext.
- Thus, no matter how large the key size of the public-key scheme, the attack is reduced to a brute-force attack on a 56-bitkey.
- This attack can be prevented by appending some random bits to such simple messages.

The RSA Algorithm

- RSA algorithm processes plaintext blocks, with each block having a binary value less than some number n.
- The block size must be less than or equal to $\log_2(n) + 1$.
- Steps for RSA:

- Select two large prime numbers **p** and **q.**
- Calculate n = pq.
- Calculate $\phi(n) = (p - 1)(q - 1)$.
- Select e such that e is relatively prime to $\phi(n)$.
- Compute d such that $d*e \equiv 1 \pmod{\phi(n)}$.

- RSA is a public key algorithm with public key PU = {e, n} and private key PR = {d, n}.
- Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C:

$$\mathbf{C = M^e \bmod n}$$
$$\mathbf{M = C^d \bmod n}$$
$$\mathbf{M = (M^e)^d \bmod n}$$

- For the above equation to be true, d must be an inverse of e.
- D can be calculated from e using extended Euclid's algorithm.
- Both sender and receiver must know the value of n.
- The sender knows the value of e, and only the receiver knows the value of d.
- RSA can also be subjected to various attacks like brute-force attack, various mathematical attacks, timing attacks and chosen ciphertext attacks.
- Some of these attacks exploit the mathematical characteristics of RSA.

RSA Example

- Let p = 17 and q = 11.
- n = pq = 17 X 11 = 187
- $\phi(n) = (p-1)(q-1) = 16 \text{ X } 10 = 160$
- Let e be 7.
- $d = e^{-1} \bmod 160 = 23$ (can be calculated by extended Euclid's algorithm).
- Now, PU = { 7, 187 } and PR = { 23, 187 }
- If M = 88, then by RSA

Encryption

$C = 88^7 \bmod 187$
$= [\ 88 \text{ X } 88^2 \text{ X } 88^4\ ] \bmod 187$
$= 11$

Decryption

- Here, C = 11.

$M = 11^{23} \bmod 187$
$= [\ 11 \text{ X } 11^2 \text{ X } 11^4 \text{ X } 11^8 \text{ X } 11^8\ ] \bmod 187$
$= 88$

Computational Aspects of RSA

- **Use of modular arithmetic makes calculation practical:**
  - Both encryption and decryption in RSA involve calculating huge exponents, mod n.
  - If the exponentiation is done over the integers and then reduced modulo n, the intermediate values would be extremely large.
  - However, the following property of modular arithmetic makes the calculation practical:
    [(a mod n) * (b mod n)] mod n = (a * b) mod n]

- **Efficiency of exponentiation:**
  - RSA deals with very large exponents.
  - But this operation can be implemented efficiently.
  - Consider x^{16}. A straightforward approach requires multiplying x 16 times.
  - But, the same can be achieved by only four multiplications - x^2,$(x^2)^2=x^4$, $(x^4)^2=x^8$, $(x^8)^2=x^{16}$.
- **Efficient operation using the public key:**
  - To speed up the operation of the RSA algorithm using the public key, a specific choice of e is usually made.
  - The most common choice is 65537 (2^{16} + 1).

The Security of RSA

Four possible approaches to attacking the RSA algorithm are

Brute force

- This involves trying all possible private keys.
- The defense against this attack is to use a large key.
- However, the key should not be so large that it makes calculation too time consuming and hence impractical.
- Thus, there is a tradeoff between key size and security of RSA.

Mathematical attacks

- There are three approaches to attacking RSA mathematically, all of which are equivalent in effort to the factoring the product of two primes.
  - Factor n into its two prime factors. This enables calculation of $\phi(n) = (p - 1)(q - 1)$, which in turn enables determination of $d = e^{-1} \pmod{\phi(n)}$.
  - Determine $\phi(n)$ directly, without first determining p and q. Again, this enables determination of $d = e^{-1} \pmod{\phi(n)}$. This is equivalent to factoring n.
  - Determine d directly, without first determining $\phi(n)$ which is at least as time-consuming as the factoring problem.
- Size of **n** should be considerably large.
- To avoid values of **n** that may be factored more easily, the
  - **P** and **q** should differ in length by only a few digits.
  - Both (p - 1) and (q - 1) should contain a large prime factor.
  - gcd(p - 1, q - 1) should be small.

Timing attacks

- These depend on the running time of the decryption algorithm.
- It is a ciphertext only attack.
- In RSA, modular exponentiation is done bit by bit. Suppose the system uses a modular multiplication function that is very fast in almost all cases but in a few cases takes much more time than an entire average modular exponentiation.
- The attack proceeds as follows:
  - Suppose that the first j bits are known.
  - For a given ciphertext, the attacker can complete the first j iterations of the for-loop.
  - The operationof the subsequent step depends on the unknown exponent bit.

- For a few values of e and d, the modular multiplication will be extremely slow, and the attacker knows which these are.
- Therefore, if the observed time to execute the decryption algorithm is always slow when this particular iteration is slow with a 1 bit, then this bit is assumed to be 1.
- If a number of observed execution times for the entire algorithm are fast, then this bit is assumed to be 0.

- Generally modular exponentiation implementations do not have such extreme timing variations but there is enough variation to make this attack practical.
- Countermeasures to this attack are:
  - **Constant exponentiation time:** Ensure that all exponentiations take the same amount of time before returning a result. However, this degrades performance.
  - **Random delay:** Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack. But if the defenders don't add enough noise, attackers could still succeed by 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 that is essential to the timing attack. Steps for blinding are:
    - ✓ Generate a secret random number **r** between 0 and n – 1.
    - ✓ **$C' = C(r^e) \bmod n$**, where e is the public exponent.
    - ✓ Compute **$M' = (C')^d \bmod n$**
    - ✓ Compute **$M = M'r^{-1} \bmod n$**, r^{-1} is the multiplicative inverse of r mod n

    Implementing blinding incurs only a 2 to 10% performance penalty.

Chosen ciphertext attacks

- This type of attack exploits properties of the RSA algorithm.
- The adversary could select a plaintext, encrypt it with the target's public key, and then gets the plaintext back by having it decrypted with the private key.
- This provides no new information. Instead, the adversary exploits properties of RSA and selects blocks of data that, when processed using the target's private key, gives information needed for cryptanalysis.
- An example of one such attack is that the attacker exploits the following property of RSA.

$E(PU,M1) \times E(PU,M2) = E(PU, [M1 \times M2])$

  - Compute $X = (C \times 2^e) \bmod n$.
  - Submit X as a chosen ciphertext and receive back $Y = X^d \bmod n$
  - But now note that

    $X = (C \bmod n) * (2^e \bmod n)$

    $= (M^e \bmod n) * (2^e \bmod n)$

    $X = (2M)^e \bmod n$
  - Therefore, $Y = (2M) \bmod n$.
- To overcome this simple attack, randomly pad the plaintext before encryption.
- This randomizes the ciphertext so that the Equation no longer holds.

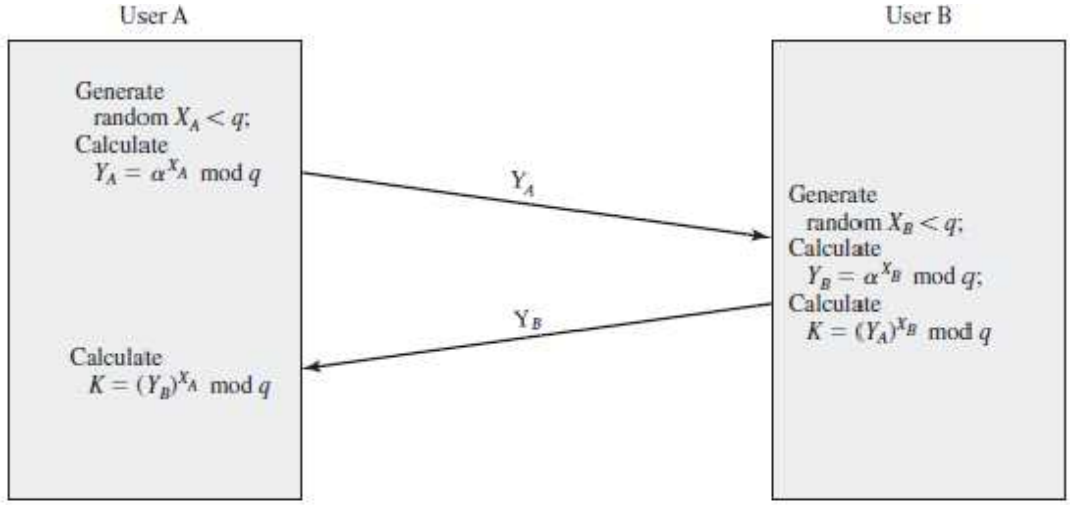
Diffie-Hellman key exchange

- The purpose of the algorithm is to enable two users to securely exchange a key that can be used for encryption of messages.

- The Diffie-Hellman algorithm depends on the difficulty of computing discrete logarithms.
- For this scheme, there are two publicly known numbers: a prime number **q** and an integer **α** that is a primitive root of **q.**
- Suppose the users A and B wish to exchange a key.
- User A selects a random private integer $X_A < q$ and computes public integer $Y_A = \alpha^{X_A} \bmod q$.
- Similarly, user B independently selects a random private integer $X_B < q$ and computes public integer $Y_B = \alpha^{X_B} \bmod q$.
- Each side keeps the **X** value private and makes the **Y** value available publicly to the other side.
- User A computes the key as $K = (Y_B)^{X_A} \bmod q$ and
- User B computes the key as $K = (Y_A)^{X_B} \bmod q$.
- These two calculations produce identical results:

$$
\begin{aligned}
K &= (Y_A)^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q \text{ (by the rules of modular arithmetic)} \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (Y_B)^{X_A} \bmod q
\end{aligned}
$$

- The result is that the two sides have exchanged a secret value.
- Furthermore, because X_A and X_B are private, an adversary only has the following information: q, α, Y_A and Y_B.
- Thus, the adversary is forced to take a discrete logarithm to determine the key.
- For example, to determine the private key of user B, an adversary must compute $X_B = \text{dlog}_{a,q}(Y_B)$
- The adversary can then calculate the key K.
- The security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms.
- For large primes, calculating discrete logarithms is considered infeasible.



Diffie-Hellman Key Exchange

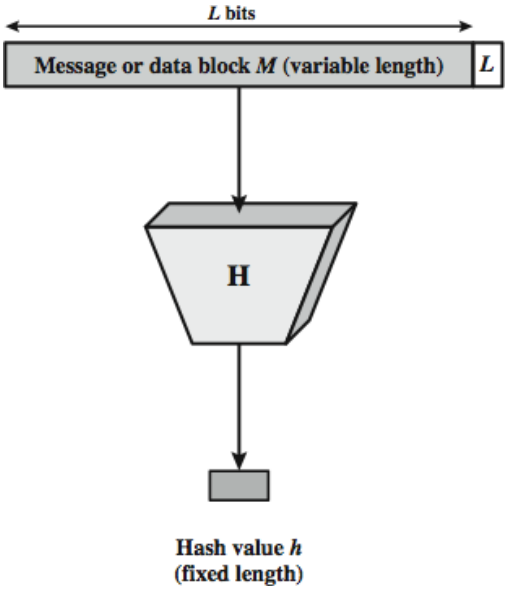
- Because only A and B can determine the key, no other user can read the message (confidentiality).
- Recipient B knows that only user A could have created a message using this key (authentication).
- However, the technique does not protect against replay attacks. One such example is Man-in-the-Middle Attack.

Man-in-the-Middle Attack

- Suppose A and B wish to exchange keys, and E is the attacker.
- The attack proceeds as follows.
  - E generates two random private keys $\mathbf{X_{E1}}$andand$\mathbf{X_{E2}}$then computing the corresponding public keys $\mathbf{Y_{E1}}$and $\mathbf{Y_{E2}}$.
  - A transmits **YA** to B.
  - E intercepts $\mathbf{Y_A}$ and transmits ' $\mathbf{Y_{E1}}$ to B.
  - B receives $\mathbf{Y_{E1}}$ and calculates $\mathbf{K_1 = (Y_{E1})^{XB} mod\ q.}$
  - B transmits $\mathbf{Y_B}$to A.
  - E intercepts $\mathbf{Y_B}$and transmits $\mathbf{Y_{E2}}$ to A.
  - A receives$\mathbf{Y_{E2}}$ and calculates $\mathbf{K_2 = (Y_{E2})^{XA} mod\ q.}$
  - E also calculates $\mathbf{K_1 = (Y_B)^{XE1} mod\ q}$and $\mathbf{K_2 = (Y_A)^{XE2} mod\ q}$
- At this point, B and A think that they share a secret key, but instead Band E share secret key and A and E share secret key.
- All future communication between B and A is compromised in the following way.
  - A sends an encrypted message **M** as $\mathbf{E(K_2, M).}$
  - E intercepts the encrypted message and decrypts it to recover M.
  - E sends $\mathbf{E(K_1, M)}$ or $\mathbf{E(K_1, M')}$ to B , where **M'** is any message.
- The key exchange protocol is vulnerable to such an attack because it does not authenticate the participants.
- This vulnerability can be overcome with the use of digital signatures and public-key certificates.

Hash Function

- A **hash function** H accepts a variable-length block of data as input and produces a fixed-size hash value.
- A "good" hash function has the property that the results of applying the function to a large set of inputs will produce outputs that are evenly distributed and apparently random.
- In general terms, the principal object of a hash function is data integrity.
- A change to any bit or bits in results, with high probability, in a change to the hash code.



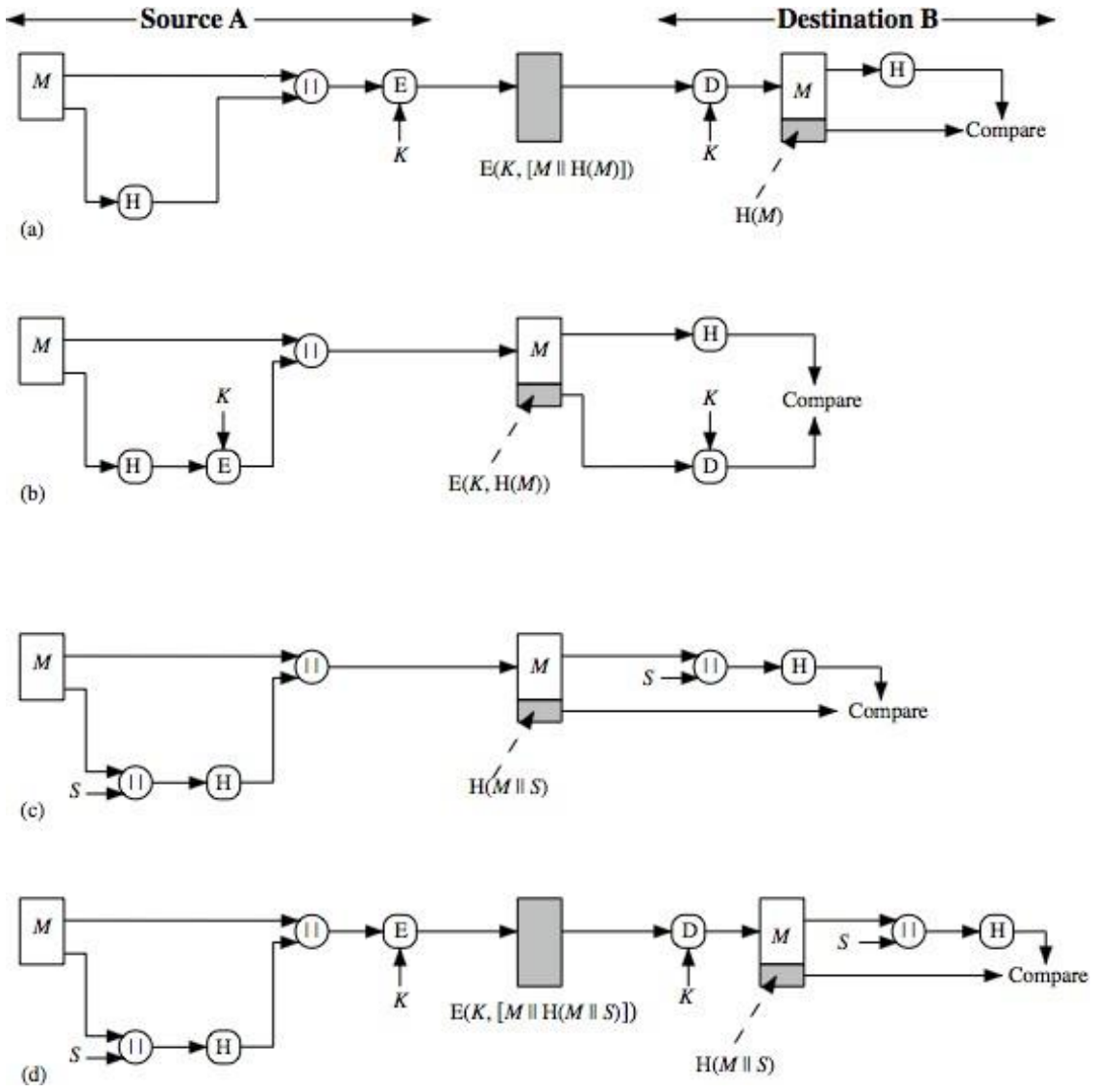
- Hash function used for security applications is referred to as a **cryptographic hash function**.
- A cryptographic hash function is an algorithm for which it is computationally infeasible to find either
  - A data object that maps to a pre-specified hash result (the one-way property) or
  - Two data objects that map to the same hash result (the collision-free property).
- Because of these characteristics, hash functions are often used to determine whether or not data has changed.
- Figure depicts the general operation of a cryptographic hash function.

Applications of Cryptographic Hash Functions

The range of applications in which it is employed.

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.
- Hash function is used to provide message authentication.
- The hash function value is often referred to as a **message digest**.
- Variety of ways in which a hash code can be used to provide message authentication, as follows.



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. 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.

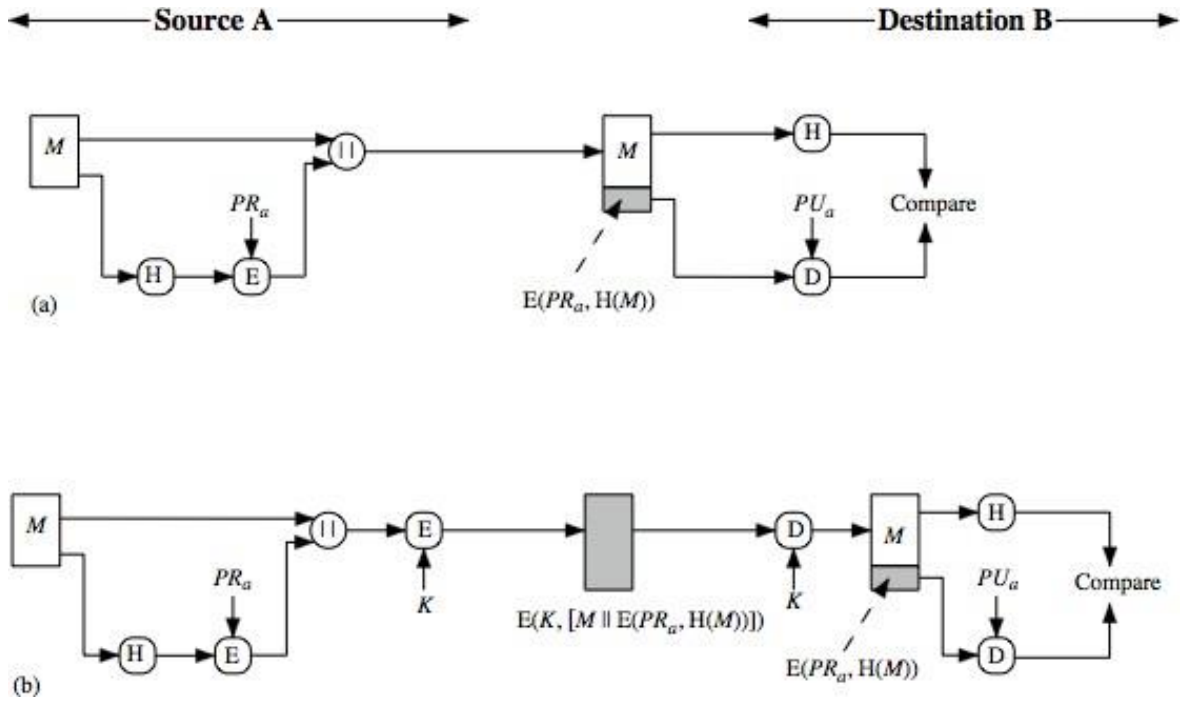
c) It is possible to use a hash function but no encryption for message authentication. Two communicating parties share a common secret value S. A computes the hash value over the concatenation of M and S and appends the resulting hash value to M. Because B possesses, it can recomputed the hash value to verify. Opponent 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.
- In this case, an attacker who wishes to alter the message would need to know the user's private key.



a) The hash code is encrypted, using public-key encryption with the sender's private key. This provides authentication. It also provides a digital signature, because only the sender could have produced the encrypted hash code.

b) If confidentiality as well as a digital signature is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key.

Other Applications

- Hash functions are commonly used to create a **one-way password file**.
- Hash functions can be used for **intrusion detection** and **virus detection**.
- A cryptographic hash function can be used to construct a **pseudorandom function (PRF)** or a **pseudorandom number generator (PRNG)**.

Simple Hash Functions

- Two simple, insecure hash functions are shown here.
- All hash functions operate using the following general principles.
  - The input (message, file, etc.) is viewed as a sequence of ***n***-bit blocks.
  - The input is processed one block at a time in an iterative fashion to produce an -bit hash function**.**

1. First Function

- One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block.
- This can be expressed as

  $C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$

  Where

C_i = ith bit of the hash code, $1 \leq i \leq n$
m = number of n − bit blocks in the input
b_{ij} = ith bit in jth block
$\oplus$ = XOR operation

- This operation produces a simple parity for each bit position and is known as a longitudinal redundancy check.
- It is reasonably effective for random data as a data integrity check.
- Each n-bit hash value is equally likely.
- Thus, the probability that a data error will result in an unchanged hash value is 2^{-n}.

2. Second Function

- A simple way to improve matters is to perform a one-bit circular shift, or rotation, on the hash value after each block is processed.
- The procedure can be summarized as follows.
  1) Initially set the n-bit hash value to zero.
  2) Process each successive n-bit block of data as follows:
     a. Rotate the current hash value to the left by one bit.
     b. XOR the block into the hash value.
- This has the effect of "randomizing" the input more completely and overcoming any regularities that appear in the input.
- Although the second procedure provides a good measure of data integrity, it is virtually useless for data security.
- When an encrypted hash code is used with a plaintext message, it is an easy matter to produce a new message that yields that hash code.
- Simply prepare the desired alternate message and then append an n-bit block that forces the new message plus block to yield the desired hash code.

Security Requirements for Cryptographic Hash Functions

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.

Security attack on Cryptographic Hash Function

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.
- We look first at brute-force attacks.

Preimage and Second Preimage Attacks

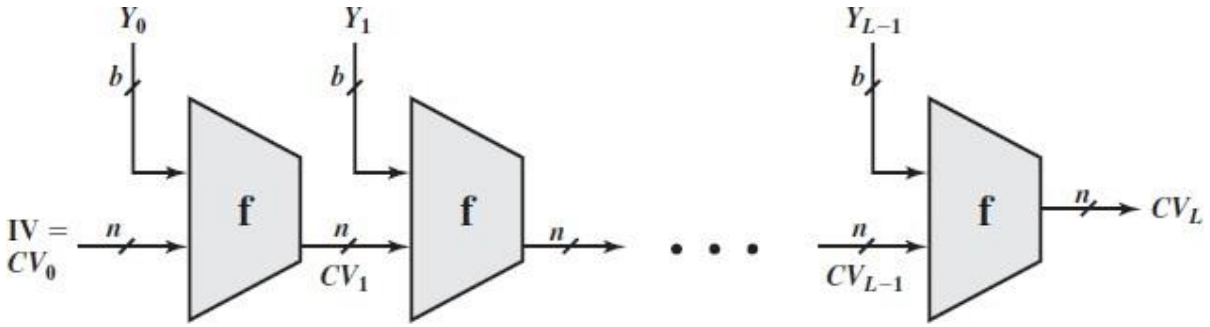
- Adversary wishes to find a value such that H(y) is equal to a given hash value h.
- The brute-force method is to pick values of y at random and try each value until a collision occurs.
- For an m-bit hash value, the level of effort is proportional to 2^m.
- Specifically, the adversary would have to try, on average, 2^{m-1} values of y to find one that generates a given hash value h.

Collision Resistant Attacks

- Adversary wishes to find two messages or data blocks, x and y, that yield the same hash function: H(x)=H(y).
- This turns out to require considerably less effort than a preimage or second preimage attack.
- The effort required is explained by a mathematical result referred to as the birthday paradox.
- Thus, for an m-bit hash value, if we pick data blocks at random, we can expect to find two data blocks with the same hash value within $\sqrt{2^m} = 2^{m/2}$ attempts.
- Yuval proposed the following strategy to exploit the **birthday paradox** in a collision resistant attack
  1. The source, A, is prepared to sign a legitimate message x by appending the appropriate m-bit hash code and encrypting that hash code with A's private key.
  2. The opponent generates $2^{m/2}$ variations x' of x, all of which convey essentially the same meaning, and stores the messages and their hash values.
  3. The opponent prepares a fraudulent message y for which A's signature is desired.
  4. The opponent generates minor variations y' of y, all of which convey essentially the same meaning. For each y', the opponent computes H(y'), checks for matches with any of the H(x') values, and continues until a match is found.
  5. The opponent offers the valid variation to A for signature. This signature can then be attached to the fraudulent variation for transmission to the intended recipient.

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.
- In recent years, there has been considerable effort, and some successes, in developing cryptanalytic attacks on hash functions.
- To understand these, we need to look at the overall structure of a typical secure hash function, indicated in Figure.



IV = Initial value
CV_i = Chaining variable
Y_i = ith input block
f = Compression algorithm
L = Number of input blocks
n = Length of hash code
b = Length of input block

- Cryptanalysis of hash functions focuses on the internal structure of f and is based on attempts to find efficient techniques for producing collisions for a single execution of f.
- Once that is done, the attack must take into account the fixed value of IV.
- The attack on f depends on exploiting its internal structure.
- Typically, as with symmetric block ciphers, f consists of a series of rounds of processing, so that the attack involves analysis of the pattern of bit changes from round to round.

Hash Functions Based on Cipher Block Chaining

- A number of proposals have been made for hash functions based on using a cipher block chaining technique, but without using the secret key. One of the first such proposals was that of Rabin.
- Divide a message M into fixed-size blocks M1, M2, … , M_N and use a symmetric encryption system such as DES to compute the hash code G as

$H_0 = initial\ value$
$H_i = E(M_i, H_{i-1})$
$G = H_N$

- This is similar to the CBC technique, but in this case, there is no secret key.
- As with any hash code, this scheme is subject to the birthday attack, and if the encryption algorithm is DES and only a 64-bit hash code is produced, then the system is vulnerable.
- Furthermore, another version of the birthday attack can be used even if the opponent has access to only one message and its valid signature and cannot obtain multiple signings.
- Here is the scenario: We assume that the opponent intercepts a message with a signature in the form of an encrypted hash code and that the unencrypted hash code is bits long.
  1. Use the algorithm defined at the beginning of this subsection to calculate the unencrypted hash code G.
  2. Construct any desired message in the form $Q_1, Q_2, \dots, Q_{N-2}$.
  3. Compute $H_i = E(Q_i, H_{i-1})$ for $1 \leq i \leq (N-2)$.
  4. Generate $2^{m/2}$ random blocks; for each block X, compute $E(X, H_{N-2})$. Generate an additional $2^{m/2}$ random blocks; for each block Y, compute D (Y, G), where D is the decryption function corresponding to E.
  5. Based on the birthday paradox, with high probability there will be an X and Y such that $E(X, H_{N-2}) = D(Y, G)$.
  6. Form the message $Q_1, Q_2, \dots, Q_{N-2}$, X, Y, This message has the hash code G and therefore can be used with the intercepted encrypted signature.
- This form of attack is known as a **meet-in-the-middle-attack**.
- A number of researchers have proposed refinements intended to strengthen the basic block chaining approach. For example, Davies and Price describe the variation:

$H_i = E(M_i, H_{i-1}) \oplus H_{i-1}$

- Another variation, proposed is

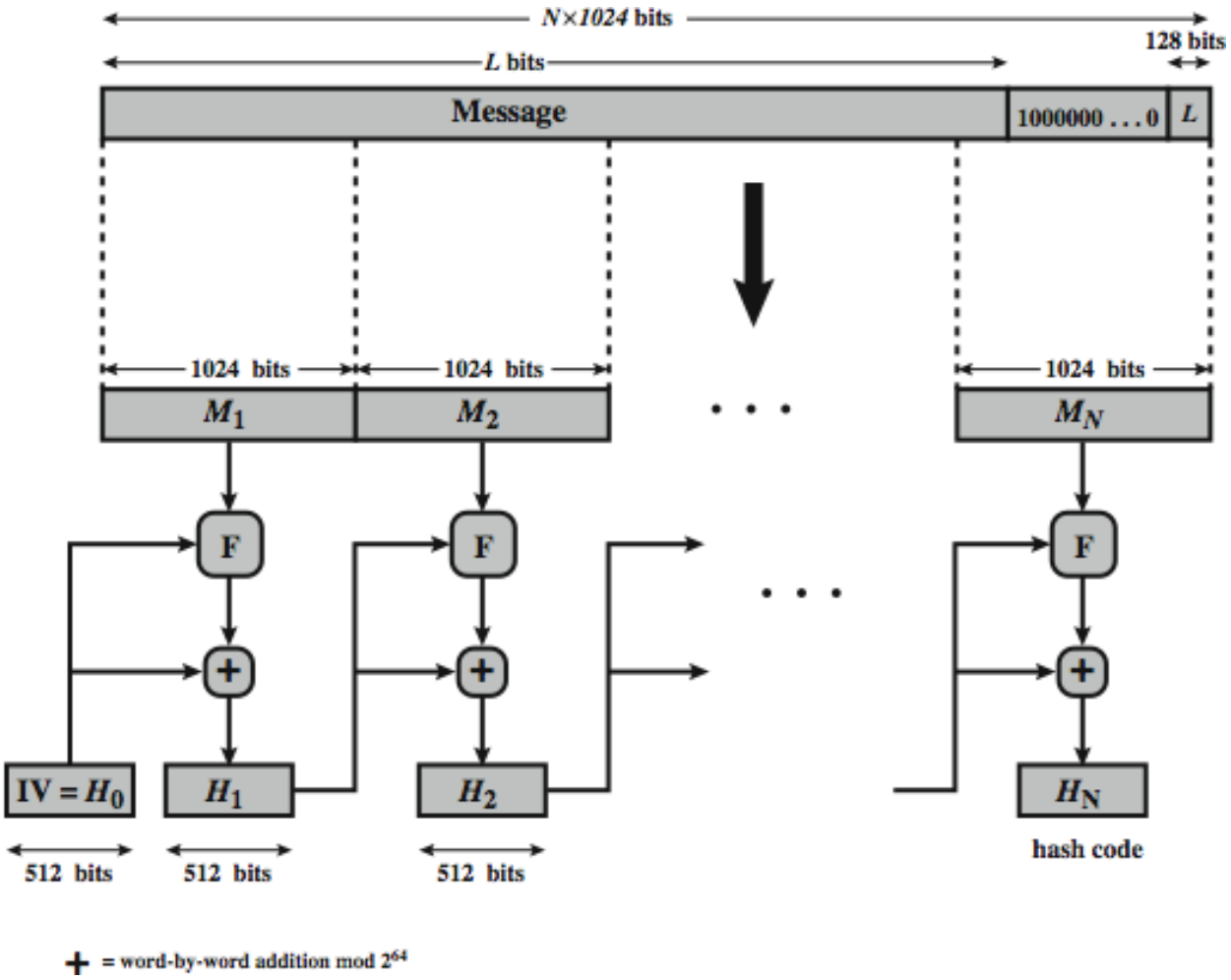
$H_i = E(H_{i-1}, M_i) \oplus M_i$

- However, both of these schemes have been shown to be vulnerable to a variety of attacks.

Secure Hash Algorithm (SHA)

- SHA is based on the hash function MD4.
- The algorithm takes as input a message of maximum length of less than 2128bitsand produces a 512-bit message digest.
- The input is processed in 1024-bit blocks.

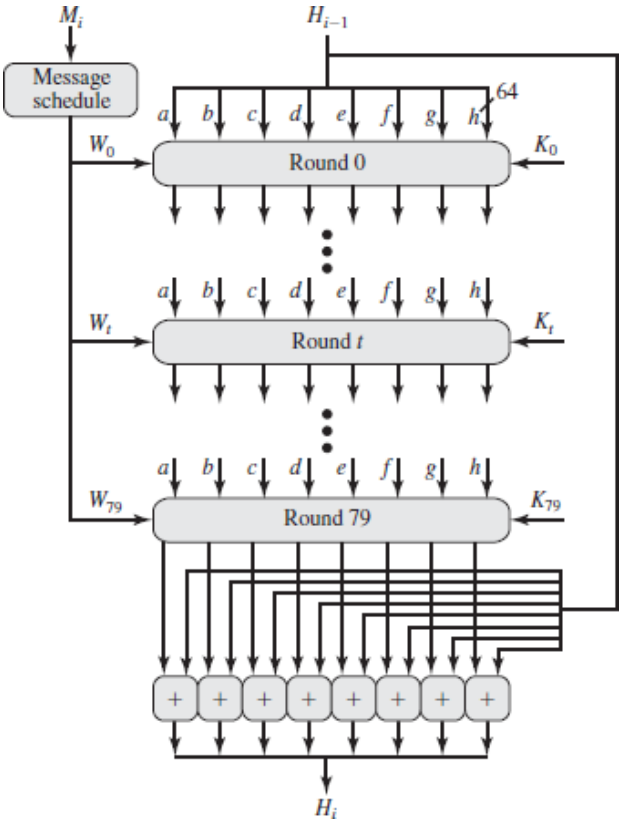
- The processing consists of the following steps:
  1. **Append padding bits.**
     - ✓ The message is padded so that its length is congruent to 896 modulo 1024.
     - ✓ The padding consists of a single 1-bit followed by the necessary number of 0-bits.
  2. **Append length.**
     - ✓ A block of 128 bits is appended to the message. This block contains the length of the original message (before the padding).
     - ✓ The message is now an integer multiple of 1024 bits in length.
- In the figure below, expanded message is represented as the sequence of 1024-bit blocks M1, M2,..., MN and the total length of the expanded message is N x 1024 bits.



  3. **Initialize hash buffer.**
     - ✓ A 512-bit buffer is used to hold intermediate and final results of the hash function.
     - ✓ The buffercan be represented as eight 64-bit registers (a, b, c, d, e, f, g, h).
     - ✓ These registers are initialized to the 64-bit integers(hexadecimal values) obtained by taking the first sixty-four bits of the fractional parts of the square roots of the first eightprime numbers.
  4. **Process message in 1024-bit (128-word) blocks.**
     - ✓ The heart of the algorithm is a module **F**that consists of 80 rounds.
- SHA has 80 rounds.
- Each round takes as input:
  - 512-bit buffer value ($\mathbf{H}_{i-1}$)
  - 64-bit words $\mathbf{W}_t$obtained from the current data block by message schedule.

- Additive constant $\mathbf{K_t}$ which represent the first sixty-four bits of the fractional parts of the cube roots of the first eighty prime numbers.

- The contents of the buffer are updated after every round.



SHA algorithm

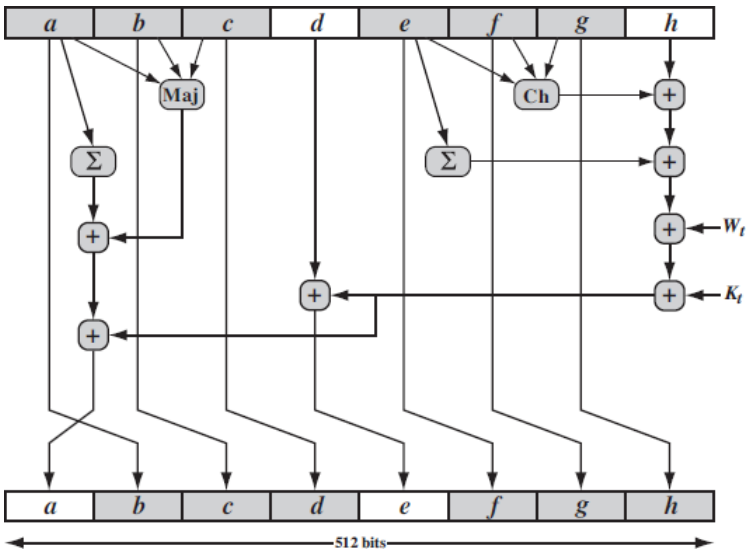
- The output of the eightieth round is added modulo 264to the input to the first round (Hi-1) to produce Hi.

5. Output.

✓ After all *N* 1024-bit blocks have been processed, the output from the N^{th} stage is the 512-bit message digest.

SHA-512 Round Function

- Each round updates the buffer in the following way:



$$T_1 = h + \text{Ch}(e, f, g) + \left(\sum_1^{512} e\right) + W_t + K_t$$
$$T_2 = \left(\sum_0^{512} a\right) + \text{Maj}(a, b, c)$$
$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

Where

t = step number; $0 \le t \le 79$

$\text{Ch}(e, f, g) = (e \text{ AND } f) \oplus (\text{NOT } e \text{ AND } g)$
the conditional function: If e then f else g

$\text{Maj}(a, b, c) = (a \text{ AND } b) \oplus (a \text{ AND } c) \oplus (b \text{ AND } c)$
the function is true only of the majority (two or three) of the arguments are true

$\left(\sum_0^{512} a\right) = \text{ROTR}^{28}(a) \oplus \text{ROTR}^{34}(a) \oplus \text{ROTR}^{39}(a)$

$\left(\sum_1^{512} e\right) = \text{ROTR}^{14}(e) \oplus \text{ROTR}^{18}(e) \oplus \text{ROTR}^{41}(e)$

$\text{ROTR}^n(x)$ = circular right shift (rotation) of the 64-bit argument x by n bits

W_t = a 64-bit word derived from the current 512-bit input block

K_t = a 64-bit additive constant

$+$ = addition modulo 2^{64}

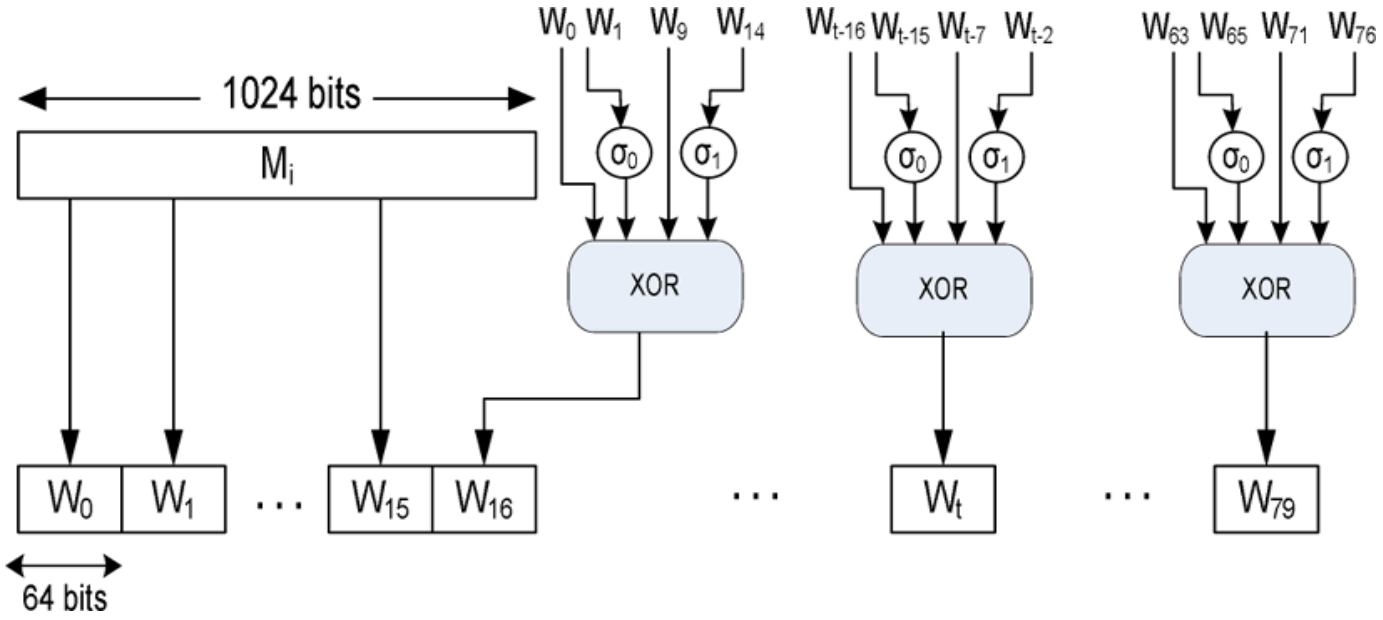
Message Schedule

- The 64-bit word values Wtare derived from the 1024-bit message.
- The first16 values ofWt are taken directly from the 16 words of the current block. The remaining values are defined as follows:

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

where

$$\sigma_0^{512}(x) = \text{ROTR}^1(x) \oplus \text{ROTR}^8(x) \oplus \text{SHR}^7(x)$$
$$\sigma_1^{512}(x) = \text{ROTR}^{19}(x) \oplus \text{ROTR}^{61}(x) \oplus \text{SHR}^6(x)$$
$$\text{ROTR}^n(x) = \text{circular right shift (rotation) of the 64-bit argument } x \text{ by } n \text{ bits}$$
$$\text{SHR}^n(x) = \text{left shift of the 64-bit argument } x \text{ by } n \text{ bits with padding by zeros on the right}$$
$$+ = \text{addition modulo } 2^{64}$$



Message schedule

- The message schedule introduces a great deal of redundancy and interdependence into the message blocksthat are compressed, which complicates the task of finding a different message block that maps to the same compression function output.

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 by (i.e., contain no modification, insertion, deletion, or replay) and that the purported identity of the sender is valid.

Need for Message Authentication

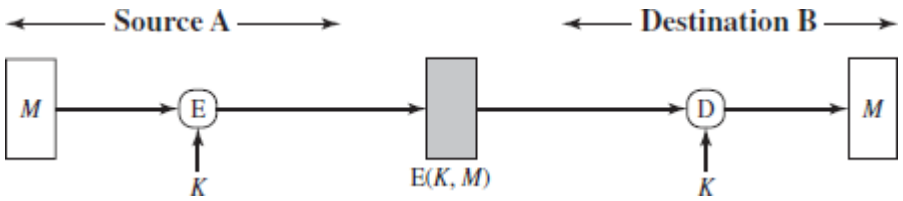
- Following attacks are possible which are the reason why authentication is needed:
  1. **Disclosure:** Release of message contents to any person not knowing the secret key.
  2. **Traffic analysis:** Discovery of the pattern of traffic between parties. Traffic analysis reveals information like the frequency and length of messages between parties and the communicating parties could be determined.
  3. **Masquerade:** Impersonating other person and sending messages.
  4. **Content modification:** Changes are made to the contents of a message. Changes may include insertion, deletion, transposition, and modification.
  5. **Sequence modification:** Sequence of messages between parties is modified. This attack may include insertion, deletion, and reordering.
  6. **Timing modification:** Delay or replay of messages.
  7. **Source repudiation:** Denial of transmission of message by source.
  8. **Destination repudiation:** Denial of receipt of message by destination.
- Message authentication verifies that received messages come from the alleged source and have not been altered.
- Message authentication may also verify sequencing and timeliness.

Authentication Techniques

- Following techniques are used for authentication:
  - **Hash function:** Hash function maps a message of any length into a fixed-lengthhash value, which serves as the authenticator.
  - **Message encryption:** The ciphertext of the entire message serves as its authenticator.
  - **Message authentication code (MAC):** A MAC is a function of the message and a secretkey that produces a fixed-length value that serves as the authenticator.
- Authentication using message encryption is explained below:

Message Encryption

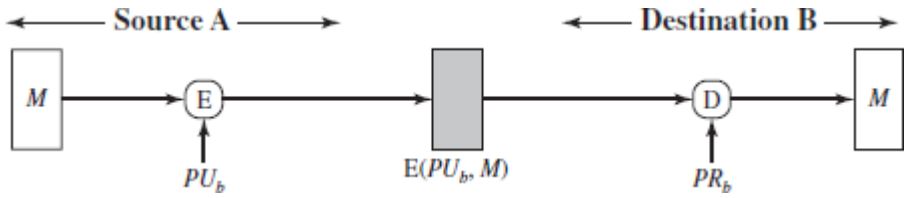
- **Symmetric Encryption:** A message **M** transmitted from source A to destination B is encrypted using a secret key K shared by A and B.



Confidentiality and authentication with symmetric encryption

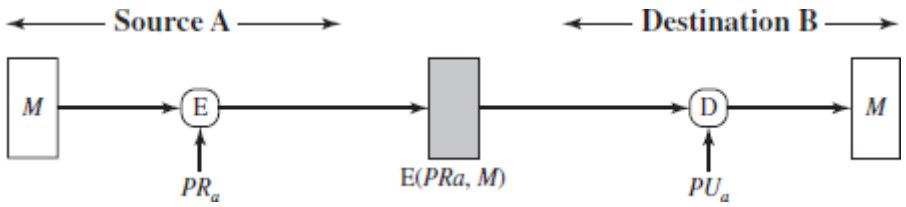
- No other party knows the key, and hence **confidentiality** is provided as no other party can recover the plaintext of the message without the knowledge of key.
- The message must have come from A because A is the only other party that possesses *K* and therefore the only other party which can construct cipher text that can be decrypted with *K*. Thus, authentication is provided.
- Furthermore, if *M* is recovered, B knows that none of the bits of *M* have been altered, because an opponent that does not know *K* would not know how to alter bits in the cipher text to produce desired changes in the plaintext. Thus, **data integrity** is also provided.

- If the message contains regular language, then the legitimacy of the message can be determined.
- But if the message contains arbitrary data like binary object file, digitized X-ray, then alteration in the message cannot be determined by simply looking at the messages.
- In that case, plaintext must have some structure like some message based function (one example is checksum) or add TCP header if TCP/IP is being used.
- **Public-Key Encryption:** The source (A) uses the public key $\mathbf{PU_b}$ of the destination (B) to encrypt *M*. Because only B has the corresponding private key $\mathbf{PR_b}$, only B can decrypt the message. But this scheme provides **confidentiality** but not authentication because any opponent could also use B's public key to encrypt a message, claiming to be A.



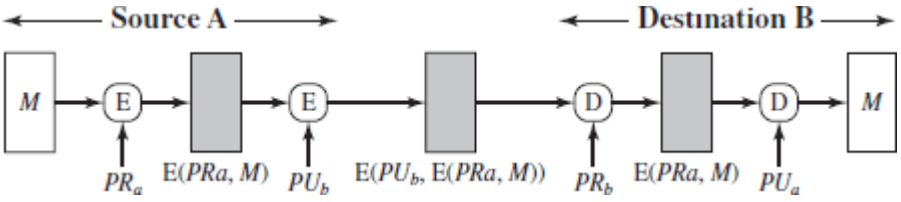
Confidentiality using public key encryption

- To provide authentication, A uses its private key to encrypt the message, and B uses A's public key to decrypt it. The message must have come from A because A is the only party that possesses $\mathbf{PR_a}$. Anyone with $\mathbf{PU_a}$ can decrypt the message. This scheme also provides digital signature because only A could have constructed the cipher text by encrypting it with $\mathbf{PR_a}$.



Authentication using public key encryption

- If both authentication and confidentiality is needed, then message is encrypted using both $\mathbf{PU_a}$ and **PRa.by** using its private key to encrypt. Note that this scheme does not provide confidentiality.



Confidentiality and Authentication using public key encryption

- This scheme also requires some structure in plaintext if it contains arbitrary data.

Message Authentication Code /Cryptographic Checksum

- Cryptographic checksum or MAC is a function of the message and a secret key that produces a fixed-length value that serves as the authenticator.

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

where

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

- A MAC function is similar to encryption. One difference is that the MAC algorithm need not be reversible, as in the case of decryption.
- A MAC function is generally a many-to-one function.

Application of MAC

- Three situations in which a message authentication code is used are:
  1. **Many applications need to broadcast message to a number of destinations.**
     - ✓ Examples are notification to users that the network is now unavailable or an alarm signal in a military control center.
     - ✓ Instead of decrypting message at every node it is cheaper and more reliable to have only one destination responsible for monitoring authenticity.
     - ✓ The message is broadcasted in plaintext with an associated message authentication code. The responsible system has the secret key and performs authentication.
     - ✓ If a violation occurs, the other destination systems are alerted by a general alarm.
  2. **One side in the communication has a heavy load and cannot afford the time to decrypt all incoming messages.**
     - ✓ Authentication is carried out on a selective basis. Messages are chosen at random for checking.
  3. **Authentication of a computer program in plaintext.**
     - ✓ The computer program can be executed without having to decrypt it every time.
     - ✓ However, if a message authentication code were attached to the program, it could be checked whenever assurance is required about the integrity of the program.

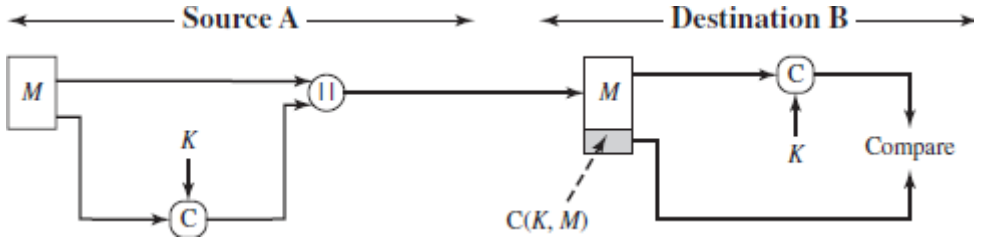
Basic Uses of MAC

- A MAC is an 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. The MAC is then appended to the message.
- Here, sender and receiver share a secret key.
- When A has to send a message to B, it calculates the MAC as a function of the message and the key:

MAC = MAC (*K*, *M*)

Where M is plaintext
C is the MAC function
K is the secret key and
MAC is the 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.



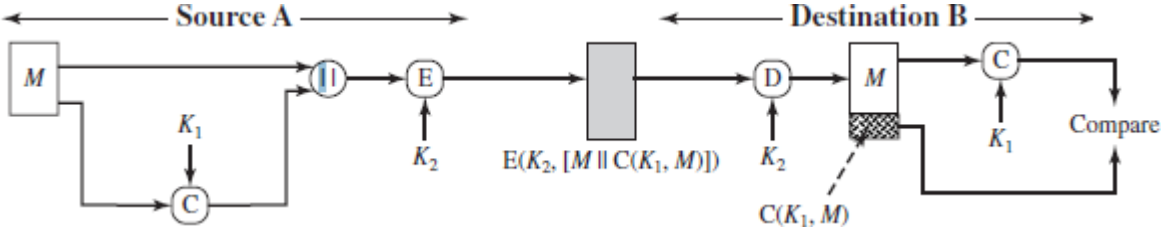
Authentication using MAC, no confidentiality

- Since only the receiver and the sender know the secret key, and if the received MAC matches the calculated MAC, then

○ The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC.

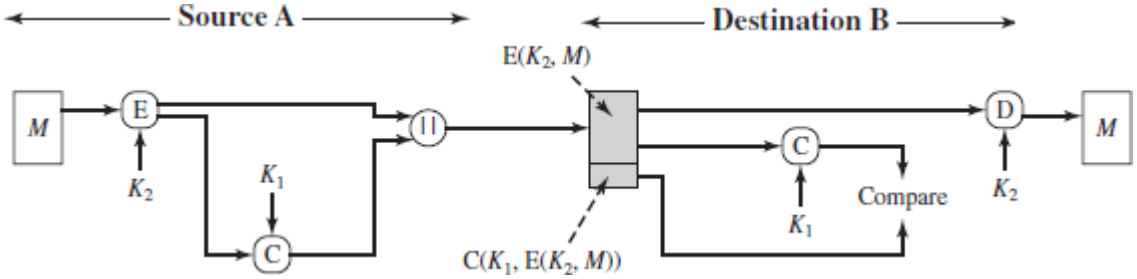
○ The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key.

- Confidentiality can be provided by performing message encryption either after or before the MAC algorithm.
- In both these cases, two separate keys are needed, each of which is shared by the sender and the receiver.
- MAC can be calculated with the message as input and then concatenated to the message. The entire block is then encrypted.



Authentication and confidentiality using MAC

- It is preferable to tie the authentication directly to the plaintext, hence the above method is typically preferred.
- Alternately, the message is encrypted first. Then the MAC is calculated using the resulting cipher text and is concatenated to the cipher text.



Authentication and confidentiality using MAC

Requirements for Message Authentication Codes

- A MAC, also known as a cryptographic checksum, is generated by a function C of the form

$T = MAC(K, M)$

Where

M is a variable-length message,

K is a secret key shared only by sender and receiver, and

MAC (K, M) is the fixed-length authenticator also called a **tag**.

- The tag 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 tag.
- When an entire message is encrypted for confidentiality, using either symmetric or asymmetric encryption, the security of the scheme generally depends on the bit length of the key.
- Barring some weakness in the algorithm, the opponent must resort to a brute-force attack using all possible keys.
- On average, such an attack will require 2^{k-1} attempts for a k-bit key. In particular, for a cipher text only attack, the opponent, given cipher text C, performs $P_i = D(K_i, C)$ for all possible key values K_i until a P_i is produced that matches the form of acceptable plaintext.
- Then the MAC function should satisfy the following requirements.
  1. If an opponent observes M and $MAC(K, M)$, it should be computationally infeasible for the opponent to construct a message M' such that $MAC(K, M) = MAC(K, M')$.
  2. $MAC(K, M)$ should be uniformly distributed in the sense that for randomly chosen messages, M and M', the probability that $MAC(K, M) = MAC(K, M')$ is 2^{-n}, where n is the number of bits in the tag.
  3. Let M' be equal to some known transformation on M. That is, $M' = f(M)$. For example, f may involve inverting one or more specific bits. In that case, $\Pr[MAC(K, M) = MAC(K, M')] = 2^{-n}$

Security of MACs/ Attacks on MACs

We group attacks on MACs into two categories: brute-force attacks and cryptanalysis.

Brute-Force Attacks

- A brute-force attack on a MAC requires more known message-MAC pairs than a brute-force attack on a hash function.
- There are two types of possible attack:
  - attack the key space
  - attack the MAC value

1. Attacking the key space

- ✓ If an attacker can determine the MAC key, then it is possible to generate a valid MAC value for any input.
- ✓ Suppose the key size is **k** bits and that the attacker has one known text–tag (MAC) pair.
- ✓ The attacker can then compute the **n-bit** tag on the known text for all possible keys.
- ✓ At least one key will produce the correct MAC value for the message. Till now, the level of effort is $\mathbf{2^k}$.
- ✓ However, the MAC is a many-to-one mapping, so there may be other keys that produce the correct value.
- ✓ Thus, if more than one key is found to produce the correct value, additional text–tag pairs must be tested.
- ✓ The level of effort becomes less with each additional text–MAC pair and after 2 or 3 levels, a single key is obtained.

2. Attacking the MAC value

- ✓ The attacker will try to generate a valid MAC for a given message or to find a message that matches a given MAC value.
- ✓ Here the level of effort is that of $\mathbf{2^n}$.
- ✓ This attack cannot be conducted off line without further input; the attacker will require chosen text–tag pairs or knowledge of the key.

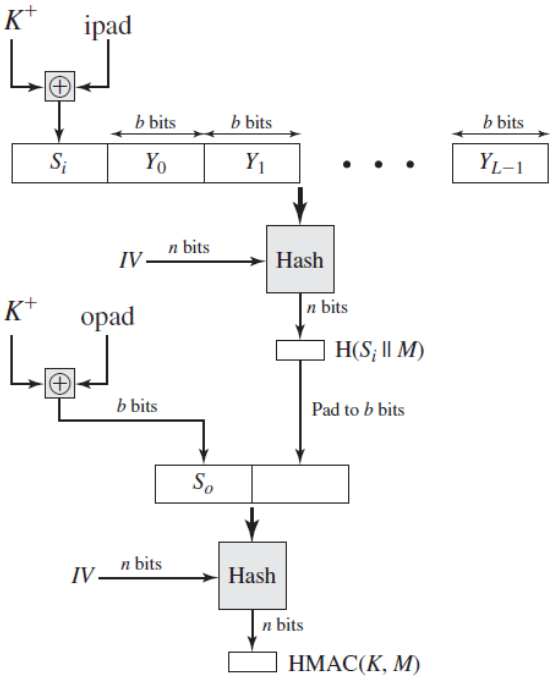
Cryptanalysis

- Cryptanalytic attacks on MAC algorithms try to exploit some property of the algorithm to perform some attack other than an exhaustive search.

- The way to measure the resistance of a MAC algorithm to cryptanalysis is to compare its strength to the effort required for a brute-force attack.
- An ideal MAC algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort.

MACs Based On Hash Functions: HMAC

- AMAC derived from hash function is called HMAC.
- The reason for developing HMAC were that hash functions incur less overhead than encryption and the code of hash functions is easily and freely available.
- The overall operation of HMAC is shown below:



1. Append zeros to the left end of *K* to create a **b-bit** string **K⁺.**
2. XOR **K⁺** with **ipad** to produce the *b*-bit block ***S*i**. Value of ipad is 36 in hexadecimal.
3. Append *M* to ***S*i**.
4. Apply H to the stream generated in the above step.
5. XOR **K⁺** with **opad** to produce the b-bit block ***S*o**. Value of opad is 5C in hexadecimal.
6. Append the hash result H from step 4 to***S*o**.
7. Apply H to the stream generated in the above step and output the result.

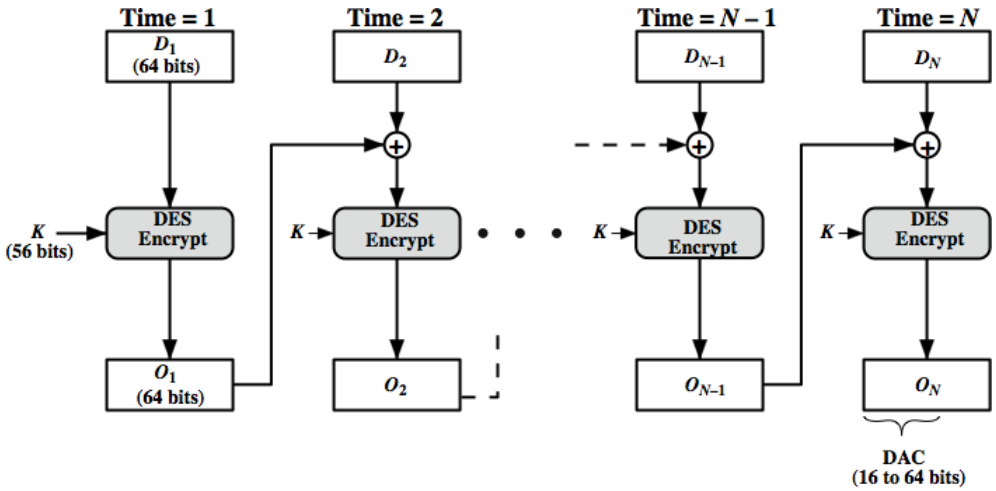
- XOR with ipad results in flipping one-half of the bits of *K.*
- Similarly, the XOR with opad results in flipping one-half of the bits of*K*, but a different set of bits.

Security of HMAC

- The security of any HMAC function is based on the cryptographic strength of the underlying hash function.
- The security of a MAC function is expressed in terms of the probability of successful forgery with a given amount of time spent by the forger and a given number of message-MAC pairs created with the same key.
- The probability of successful attack on HMAC is equivalent to one of the following attacks on the embedded hash function:
  - The attacker is able to compute an output of the compression function even with an IV that is random, secret, and unknown to the attacker.
  - The attacker finds collisions in the hash function even when the IV is random and secret.

MACS Based on Block Ciphers: Data Authentication Algorithm (DAA)

- The **Data Authentication Algorithm** (DAA), based on DES, has been one of the most widely used MACs for a number of years.
- Security weaknesses in this algorithm have been discovered, and it is being replaced by newer and stronger algorithms.
- The algorithm can be defined as using the cipher block chaining (CBC) mode of operation of DES with an initialization vector of zero.
- The data to be authenticated are grouped into contiguous 64-bit blocks: D_1, D_2, …, D_N.
- If necessary, the final block is padded on the right with zeroes to form a full 64-bit block.
- Using the DES encryption algorithm E and a secret key K, a data authentication code (DAC) is calculated as follows:



$O_1 = E(K, D)$
$O_2 = E(K, [D_2 \oplus O_1)$
$O_3 = E(K, [D_3 \oplus O_2)$
.
.
.
$O_N = E(K, [D_N \oplus O_{N-1})$

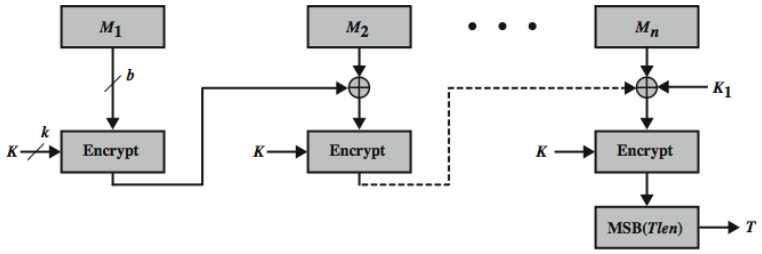
- The DAC consists of either the entire block O_N or the leftmost M bits of the block, with $16 \le M \le 64$.

MACS Based on Block Ciphers: Cipher-Based Message Authentication Code (CMAC)

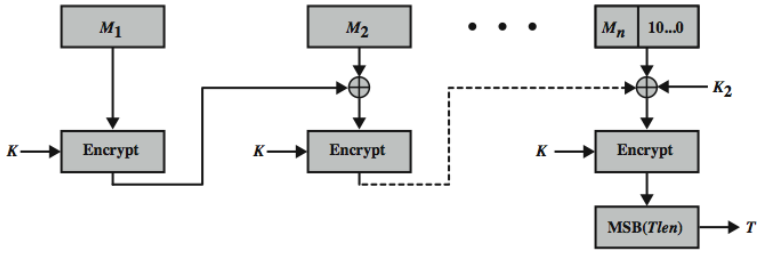
- As was mentioned, DAA has been widely adopted in government and industry.
- MAC is secure under a reasonable set of security criteria, with the following restriction.
  - Only messages of one fixed length of **mn** bits are processed, where **n** is the cipher block size and **m** is a fixed positive integer.
- Black and Rogaway demonstrated that this limitation could be overcome using three keys:
- one key of length **K** to be used at each step of the cipher block chaining and two keys of length **n**, where **k** is the key length and **n** is the cipher block length.
- This proposed construction was refined by Iwata and Kurosawa so that the two n-bit keys could be derived from the encryption key.
- This refinement, adopted by NIST, is the **Cipher-based Message Authentication Code** (CMAC) mode of operation for use with AES and triple DES.

operation of CMAC

- The message is divided into **n** blocks (M1, M2, …, Mn).
- The algorithm makes use of a k-bit encryption key **K** and an n-bit constant, $\mathbf{K_1}$.
- For AES, the key size is 128, 192, or 256 bits;
- for triple DES, the key size is 112 or 168 bits.
- CMAC is calculated as follows:



(a) Message length is integer multiple of block size



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

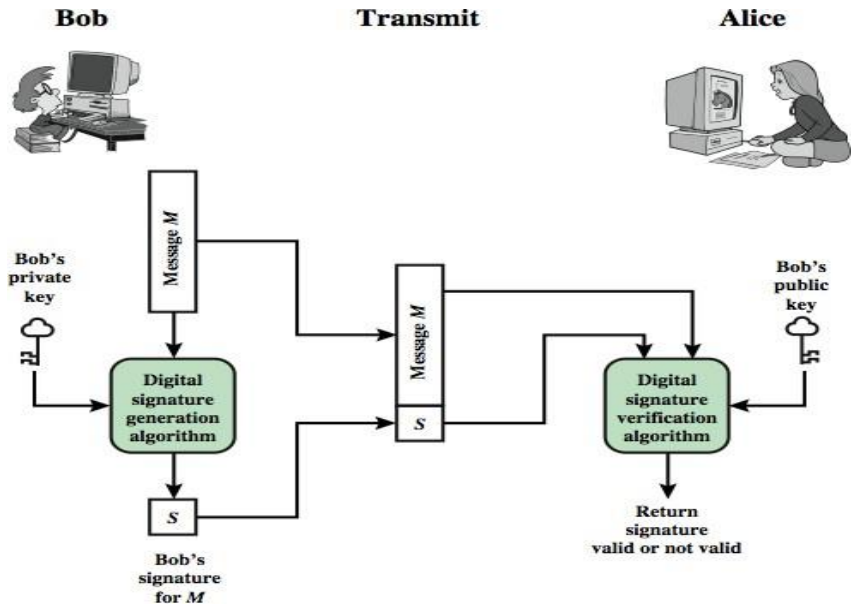
$C_1 = E(K, M_1)$
$C_2 = E(K, [M_2 \oplus C_1)$
$C_3 = E(K, [M_3 \oplus C_2)$
.
.
.
$C_n = E(K, [M_n \oplus C_{n-1} \oplus K_1)$
$T = MSB_{Tlen}(C_n)$
Where
$T =$ message authetication code, also referred to as the tag
$Tlen =$ bit length of T
$MSB_s(X) =$ the s leftmost bits of the bit string$X$

- If the message is not an integer multiple of the cipher block length, then the final block is padded to the right with a 1 and as many 0s as necessary.
- The CMAC operation then proceeds as before, except that a different n-bit key K_2 is used instead of K_1.
- The two n-bit keys are derived from the k-bit encryption key as follows.
  $L = E(K, 0^n)$
  $K_1 = L \cdot x$
  $K_2 = L \cdot x^2 = (L \cdot x) \cdot x$

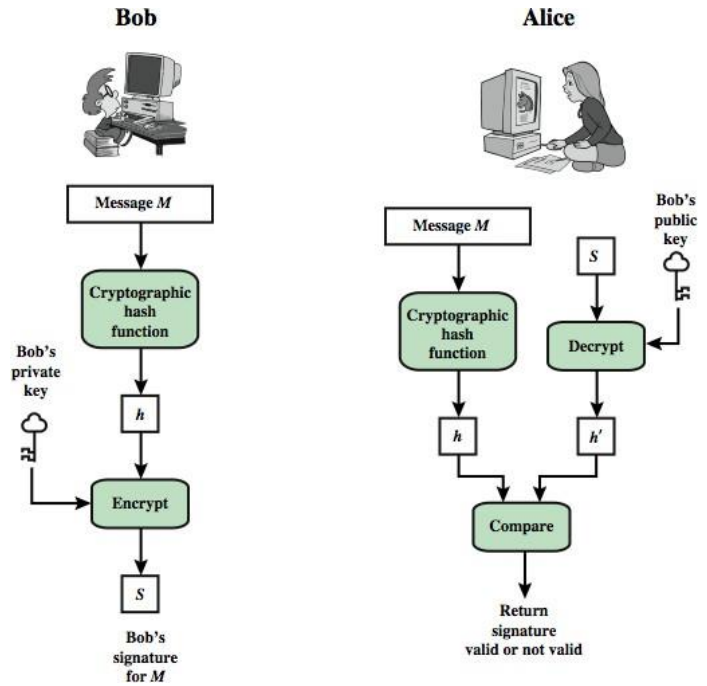
where multiplication (●) is done in the finite field GF(2^n) and **x** and $\mathbf{x^2}$ are first- and second-order polynomials that are elements of GF(2^n).

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.

Generic Model of the Digital Signature



- Figure above is a generic model of the process of making and using digital signatures.
- Bob can sign a message using a digital signature generation algorithm.
- The inputs to the algorithm are the message and Bob's private key.
- Any other user, say Alice, can verify the signature using a verification algorithm, whose inputs are the message, the signature, and Bob's public key.
- In simplified terms, the essence of the digital signature mechanism is shown in Figure below.



Needs of Digital Signature

- 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.
  1. Mary may forge a different message and claim that it came from John. Mary would simply have to create a message and append an authentication code using the key that John and Mary share.
  2. John can deny sending the message. Because it is possible for Mary to forge a message, there is no way to prove that John did in fact send the message.
- Both scenarios are of legitimate concern.
- The sender pretends that the message was never sent.
- In situations where there is not complete trust between sender and receiver, something more than authentication is needed.
- The most attractive solution to this problem is the digital signature.

Properties of Digital Signature

- The digital signature must have the following properties:
  1. It must verify the author and the date and time of the signature.
  2. It must authenticate the contents at the time of the signature.
  3. It must be verifiable by third parties, to resolve disputes.
- Thus, the digital signature function includes the authentication function.

Security of Digital Signature

- Following types of attacks can be possible on digital signature.
- Here A denotes the user whose signature method is being attacked, and C denotes the attacker.
  1. **Key-only attack:** C only knows A's public key.
  2. **Known message attack:** C is given access to a set of messages and their signatures.
  3. **Generic chosen message attack:** C chooses a list of messages before attempting to breaks A's signature scheme, independent of A's public key. C then obtains from A valid signatures for the chosen messages. The attack is generic, because it does not depend on A's public key; the same attack is used against everyone.
  4. **Directed chosen message attack:** Similar to the generic attack, except that the list of messages to be signed is chosen after C knows A's public key but before any signatures are seen.
  5. **Adaptive chosen message attack:** C is allowed to use A as an "oracle." This means the A may request signatures of messages that depend on previously obtained message–signature pairs.
- If anyone success at breaking a signature scheme can do any of the following with a non-negligible probability:
  1) **Total break:** C determines A's private key.
  2) **Universal forgery:** C finds an efficient signing algorithm that provides an equivalent way of constructing signatures on arbitrary messages.
  3) **Selective forgery:** C forges a signature for a particular message chosen by C.
  4) **Existential forgery:** C forges a signature for at least one message. C has no control over the message. Consequently, this forgery may only be a minor nuisance to A.

Digital Signature Requirements

- On the basis of the properties and attacks just discussed, we can formulate the following requirements for a digital signature.
  1) The signature must be a bit pattern that depends on the message being signed.
  2) The signature must use some information unique to the sender to prevent both forgery and denial.
  3) It must be relatively easy to produce the digital signature.
  4) It must be relatively easy to recognize and verify the digital signature.
  5) It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
  6) It must be practical to retain a copy of the digital signature in storage.

Direct Digital Signature

- The term **direct digital signature** refers to a digital signature scheme that involves only the communicating parties (source, destination).
- It is assumed that the destination knows the public key of the source.
- Confidentiality can be provided by encrypting the entire message plus signature with a shared secret key (symmetric encryption).
- Note that it is important to perform the signature function first and then an outer confidentiality function.
- In case of dispute, some third party must view the message and its signature.
- If the signature is calculated on an encrypted message, then the third party also needs access to the decryption key to read the original message.
- However, if the signature is the inner operation, then the recipient can store the plaintext message and its signature for later use in dispute resolution.
- The validity of the scheme just described depends on the security of the sender's private key.
- If a sender later wishes to deny sending a particular message, the sender can claim that the private key was lost or stolen and that someone else forged his or her signature.
- Another threat is that some private key might actually be stolen from X at time T.
- The opponent can then send a message signed with X's signature and stamped with a time before or equal to T.
- The universally accepted technique for dealing with these threats is the use of a digital certificate and certificate authorities.

Elgamal Digital Signature Scheme

- The ElGamal signature scheme involves the use of the private key for encryption and the public key for decryption.
- As with ElGamal encryption, the global elements of **ElGamal digital signature** 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 $Y_A = \alpha^{X_A} mod\ q$.
  3. A's private key is X_A; A's pubic key is $\{q, \alpha, Y_A\}$.
- To sign a message **M**, user A first computes the hash $m = H(M)$, such that **m** is an integer in the range $0 < m < q - 1$.

- A then forms a digital signature as follows.
  1) Choose a random integer K such that $1 < K < q - 1$ and$\gcd(K, q - 1) = 1$. That is, K is relatively prime to q-1.
  2) Compute $S_1 = \alpha^K mod\ q$. Note that this is the same as the computation of C_1 for ElGamal encryption.
  3) Compute $K^{-1} mod\ (q - 1)$.That is, compute the inverse of K modulo q-1.
  4) Compute $S_2 = K^{-1}(m - X_A S_1) mod (q - 1)$.
5) The signature consists of the pair (S_1, S_2).
- Any user B can verify the signature as follows.
  1. Compute $V_1 = \alpha^m mod\ q$.
  2. Compute $V_2 = (Y_A)^{S1}(S_1)^{S2}\ mod\ q$.
- The signature is valid if $V_1 = V_2$.

Example

- let us start with the q = 19 and α=10
- Alice generates a key pair as follows:
  1. Alice chooses $X_A = 16$.
  2. Then $Y_A = \alpha^{XA} mod\ q = 10^{16} mod\ 19 = 4$.
  3. Alice's private key is 16; Alice's pubic key is $\{q, \alpha, Y_A\} = \{19, 10, 4\}$.
- Suppose Alice wants to sign a message with hash value $m = 14$.
  1) Alice chooses $K = 5$, which is relatively prime to $q - 1 = 18$.
  2) $S_1 = \alpha^K mod\ q = 10^5 mod\ 19 = 3$.
  3) $K^{-1} mod\ (q - 1) = 5^{-1} mod\ (19 - 1) = 11$.
  4) $S_2 = K^{-1}(m - X_A S_1) mod (q - 1) = 11(14 - (16)(3)) mod (19 - 1) = -347\ mod\ 18 = 4$.
- Bob can verify the signature as follows.
  1. $V_1 = \alpha^m mod\ q = 10^{14} mod\ 19 = 16$.
  2. $V_2 = (Y_A)^{S1}(S_1)^{S2}\ mod\ q = (4)^3(3)^4\ mod\ 19 = 5184\ mod\ 19 = 16$.
- Thus, the signature is valid.

Schnorr Digital Signature Scheme

- Schnorr signature scheme is based on discrete logarithms.
- The Schnorr scheme minimizes the message-dependent amount of computation required to generate a signature.
- The main work for signature generation does not depend on the message and can be done during the idle time of the processor.
- The message-dependent part of the signature generation requires multiplying a 2n-bit integer with an n-bit integer.
- The scheme is based on using a prime modulus **p**, with **p-1** having a prime Factor **q** of appropriate size.
- The first part of this scheme is the generation of a private/public key pair, which consists of the following steps.
  1. Choose primes **p** and **q**, such that **q** is a prime factor of **p-1**.
  2. Choose an integer **α**, such that $\alpha^q = 1\ mod\ p$.The values **α**, **p**, and **q** comprise a global public key that can be common to a group of users.
  3. Choose a random integer **s** with $0 < s < q$.This is the user's private key.
  4. Calculate $v = \alpha^{-s} mod\ p$.This is the user's public key.
- A user with private key **s** and public key **v** generates a signature as follows.

1) Choose a random integer **r** with $0 < r < q$ and compute $x = \alpha^r mod\ p$. This computation is a preprocessing stage independent of the message **M** to be signed.
2) Concatenate the message with **x** and hash the result to compute the value **e**: $e = H(M||x)$.
3) Compute$y = (r + se) mod q$. The signature consists of the pair **(e, y)**.

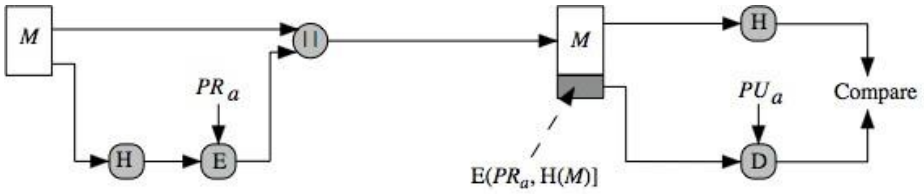
- Any other user can verify the signature as follows.
  1. Compute $x' = \alpha^y v^e mod\ p$.
  2. Verify that $e = H(M||x')$.

Digital Signature Standard

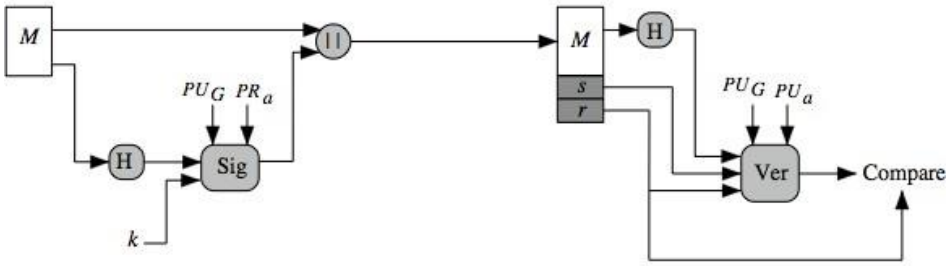
- The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard FIPS 186, known as the Digital Signature Standard (DSS).
- The DSS makes use of the SHA and presents a new digital signature technique, the **Digital Signature Algorithm (DSA)**.
- Latest version also incorporates digital signature algorithms based on RSA and on elliptic curve cryptography.

The DSS Approach

- The DSS uses an algorithm that is designed to provide only the digital signature function.
- Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique.



(a) RSA Approach



(b) DSS Approach

- Figure contrasts the DSS approach for generating digital signatures to that used with RSA.

RSA approach

- 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.

DSS approach

- The DSS approach also makes use of a hash function.
- The hash code is provided as input to a signature function along with a random number k, generated for this particular signature.
- The signature function also depends on the sender's private key (PR_a), and a set of parameters known to a group of communicating principals.
- We can consider this set to constitute a global public key (PU_G).
- The result is a signature consisting of two components, labeled *s* and *r*.
- At the receiving end, the hash code of the incoming message is generated.
- This plus the signature is input to a verification function.
- The verification function also depends on the global public key as well as the sender's public key (PU_a), which is paired with the sender's private key.
- The output of the verification function is a value that is equal to the signature component r, if the signature is valid.
- The signature function is such that only the sender, with knowledge of the private key, could have produced the valid signature.

The Digital Signature Algorithm

- The DSA is based on the difficulty of computing discrete logarithms and is based on schemes originally presented by ElGamal and Schnorr.

Global Public-Key Components

p prime number where $2^{L-1} < p < 2^L$ for $512 \le L \le 1024$ and L a multiple of 64; i.e., bit length of between 512 and 1024 bits in increments of 64 bits

q prime divisor of $(p - 1)$, where $2^{159} < q < 2^{160}$; i.e., bit length of 160 bits

g $= h^{(p-1)/q} \bmod p$, where h is any integer with $1 < h < (p - 1)$ such that $h^{(p-1)/q} \bmod p > 1$

User's Private Key

x random or pseudorandom integer with $0 < x < q$

User's Public Key

$y = g^x \bmod p$

User's Per-Message Secret Number

k = random or pseudorandom integer with $0 < k < q$

Signing

$r = (g^k \bmod p) \bmod q$

$s = [k^{-1} (\mathrm{H(M)} + xr)] \bmod q$

Signature = (r, s)

Verifying

$w = (s')^{-1} \bmod q$

$\mathrm{u_1} = [\mathrm{H(M')}w] \bmod q$

$\mathrm{u_2} = (r')w \bmod q$

$v = [(g^{u1}\, y^{u2}) \bmod p] \bmod q$

TEST: $v = r'$

M = message to be signed

$\mathrm{H}(M)$ = hash of M using SHA-1

M', r', s' = received versions of M, r, s

- Figure summarizes the algorithm.

Global Public-Key Components

- There are three parameters that are public and can be common to a group of users.
- A 160-bit prime number **q** is chosen.
- Next, a prime number **p** is selected with a length between 512 and 1024 bits such that **q** divides **(*p* - 1)**.
- Finally, g is chosen to be of the form $h^{(p-1)/q} mod\ p$, where h is an integer between 1 and **(*p* - 1)** with the restriction that **g** must be greater than 1.

User's Private Key

- With these numbers in hand, each user selects a private key and generates a public key.
- The private key **x** must be a number from 1 to **(*q* - 1)** and should be chosen randomly.

User's Public Key

- The public key is calculated from the private key as shown in figure.
- It should be computationally infeasible to determine **x**, from **y.**

User's Per-Message Secrete Number

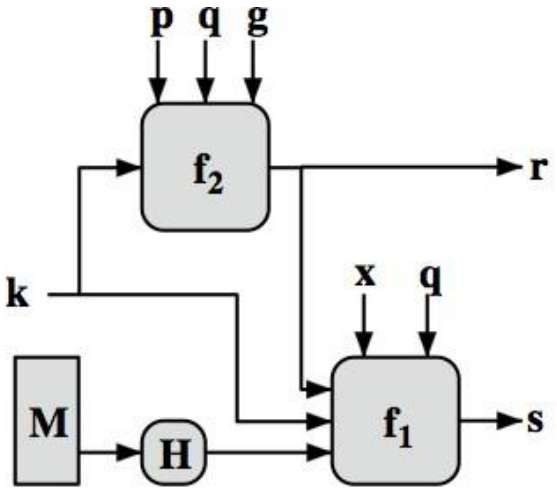
- Random number k is integer with 0 < k <q.

Creating Signature

- To create a signature, a user calculates two quantities, **r** and **s**, that are functions of the public key components **(p, q, g)**, the user's private key **x**, the hash code of the message H(M), and an additional integer **k** that should be generated randomly and be unique for each signing.

Verification

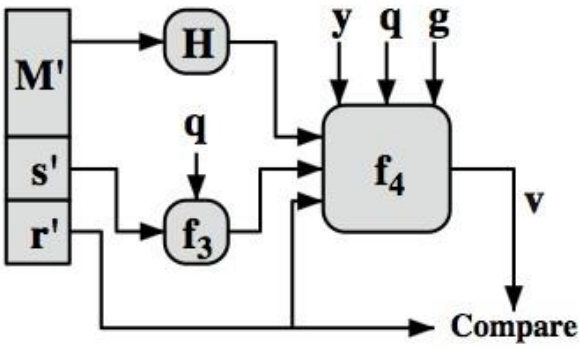
- At the receiving end, verification is performed using the formulas shown in Figure.
- The receiver generates a quantity that is a function of the public key components, the sender's public key, and the hash code of the incoming message.
- If this quantity matches the component of the signature, then the signature is validated.
- Figure below depicts the functions of signing and verifying.



$s = f_1(H(M), k, x, r, q) = (k^{-1} (H(M) + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

(a) Signing



$w = f_3(s', q) = (s')^{-1} \bmod q$

$v = f_4(y, q, g, H(M'), w, r')$

$= ((g^{(H(M')w) \bmod q} y^{r'w \bmod q}) \bmod p) \bmod q$

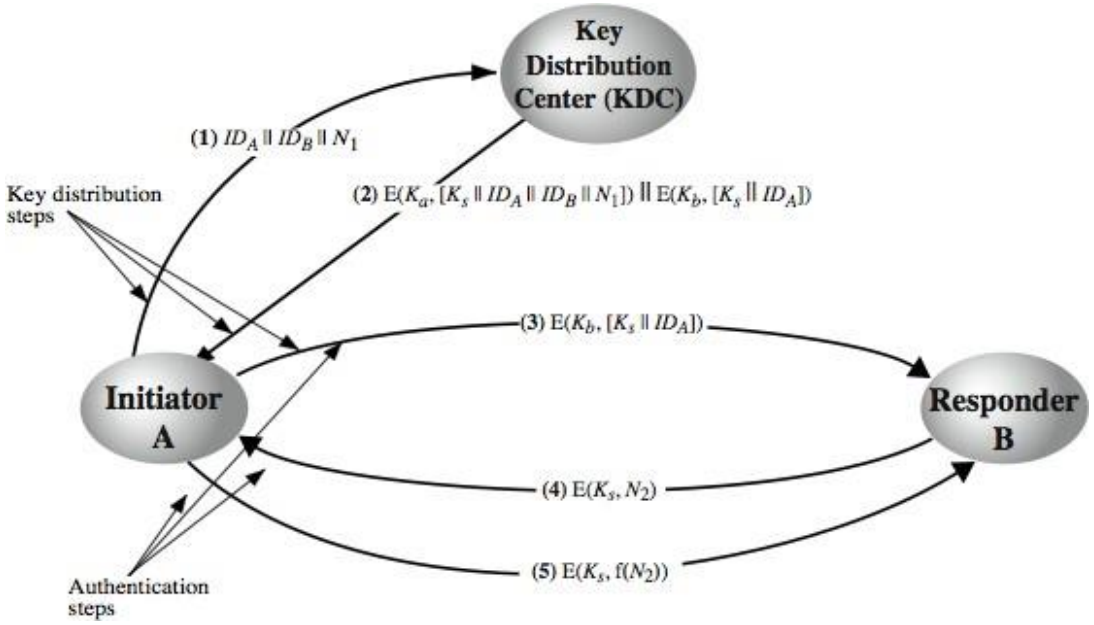
(b) Verifying

Key distribution is the function that delivers a key to two parties who wish to exchange secure encrypted data. Some sort of mechanism or protocol is needed to provide for the secure distribution of keys.

Symmetric Key Distribution Using Symmetric Encryption

- For symmetric encryption two parties must share the same key securely.
- Frequent key changes are usually desirable to limit the amount of data compromised.
- Therefore, the strength of any cryptographic system rests with the *key distribution technique*.
- For two parties A and B, key distribution can be achieved in a number 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 new key 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 on the encrypted links to A and B.
- Options 1 and 2 call for manual delivery of a key.
- For **end-to-end encryption** over a network, manual delivery is awkward.
- If end-to-end encryption is done at a network or IP level, then a key is needed for each pair of hosts on the network that wish to communicate.
- Thus, if there are N hosts, the number of required keys is $[N(N-1)]/2$.
- If encryption is done at the application level, then a key is needed for every pair of users or processes that require communication.
- Option 3 is a possibility for either link encryption or end-to-end encryption, but if an attacker ever succeeds in gaining access to one key, then all subsequent keys will be revealed.
- Some variation on option 4 has been widely adopted.
- In this scheme, a key distribution center is responsible for distributing keys to pairs of users (hosts, processes, applications) as needed.
- Each user must share a unique key with the key distribution center for purposes of key distribution.

A Key Distribution Scenario



- The key distribution concept can be deployed in a number of ways.
- A typical scenario is illustrated in Figure.
- The scenario assumes that each user shares a unique master key with the key distribution center (KDC).
- Let us assume that user A wishes to establish a logical connection with B and requires a one-time session key.
- The following steps occur.
  1. A issues a request to the KDC for a session key to protect a logical connection to B. The message includes the identity of A and B and a unique identifier, N_1 (**nonce)**. The nonce may be a timestamp, a counter, or a random number; the minimum requirement is that it differs with each request.
  2. The KDC responds with a message encrypted using K_a. Thus, A is the only one who can successfully read the message, and A knows that it originated at the KDC. The message includes two items intended for A:
     - The one-time session key, K_s, to be used for the session
     - The original request message, including the nonce, to enable A to match this response with the appropriate request

     Thus, A can verify that its original request. Also, the message includes two items intended for B:

     - The one-time session key, K_s, to be used for the session
     - An identifier of A ID_A,

     These last two items are encrypted with K_b (the master key that the KDC shares with B). They are to be sent to B to establish the connection and prove A's identity.
  3. A stores the session key and forwards the $E(K_b, [K_s||ID_A])$ to B. Because this information is encrypted with K_b, it is protected from eavesdropping. B now knows the session key K_s, knows that the other party is A (from ID_A), and knows that the information originated at the KDC (because it is encrypted using K_b).
  4. Using the newly minted session key for encryption, B sends a nonce, N_2, to A.
  5. Also, using K_s, A responds with $f(N_2)$, where f is a function that performs some transformation on N_2.
- Note that the actual key distribution involves only steps 1 through 3, but that steps 4 and 5, as well as step 3, perform an authentication function.

Hierarchical Key Control

- It is not necessary to limit the key distribution function to a single KDC.
- Indeed, for very large networks, a hierarchy of KDCs can be established.
- For communication among entities within the same local domain, the local KDC is responsible for key distribution.
- If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a global KDC.
- In this case, any one of the three KDCs involved can actually select the key.
- The hierarchical concept can be extended to three or even more layers.

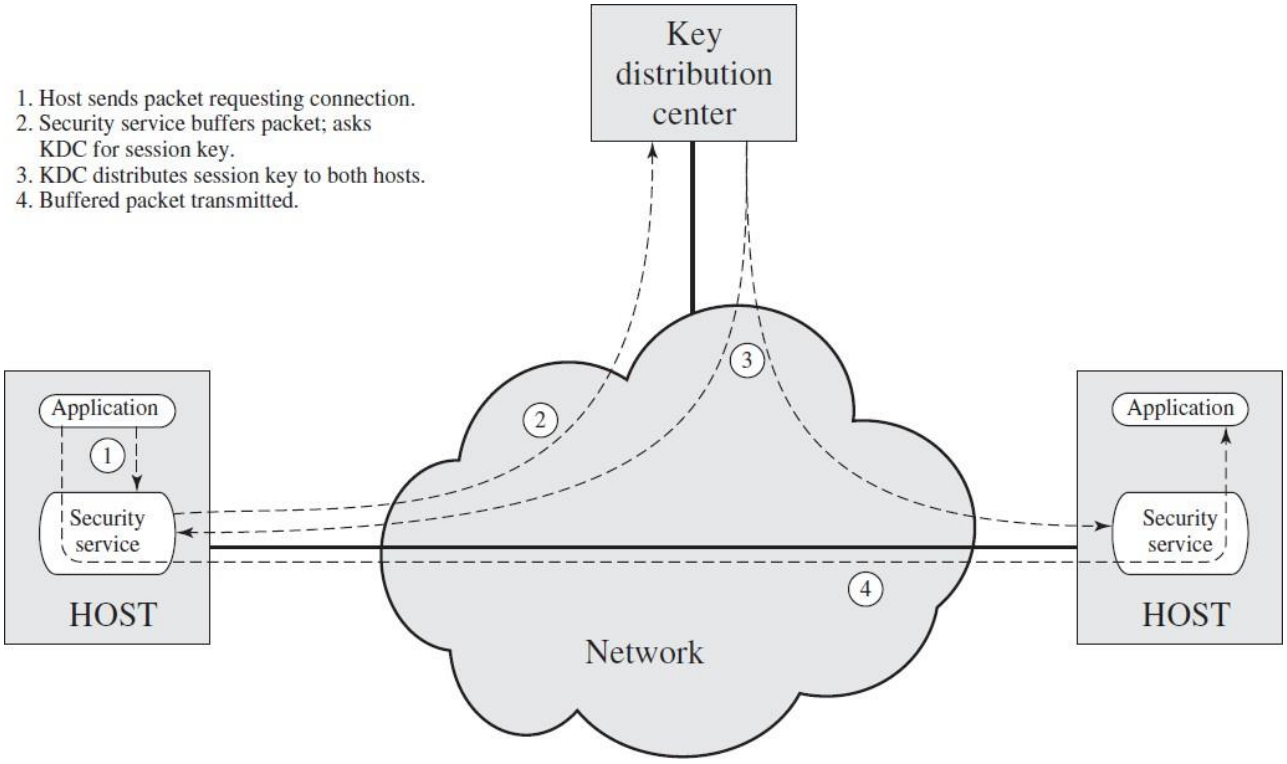
Session Key Lifetime

- The more frequently session keys are exchanged, the more secure they are, because the opponent has less cipher text to work with for any given session key.
- On the other hand, the distribution of session keys delays the start of any exchange and places a burden on network capacity.

- A security manager must try to balance these competing considerations in determining the lifetime of a particular session key.
- For connection-oriented protocols, one obvious choice is to use the same session key for the length of time that the connection is open.
- If a logical connection has a very long lifetime, then it would be prudent to change the session key periodically.
- For a connectionless protocol, it is not obvious how often one needs to change the session key.
- The most secure approach is to use a new session key for each exchange.
- However, this negates one of the principal benefits of connectionless protocols, which is minimum overhead and delay for each transaction.
- A better strategy is to use a given session key for a certain fixed period only or for a certain number of transactions.

A Transparent Key Control Scheme

- The scheme is useful for providing end-to-end encryption at a network or transport level in a way that is transparent to the end users.

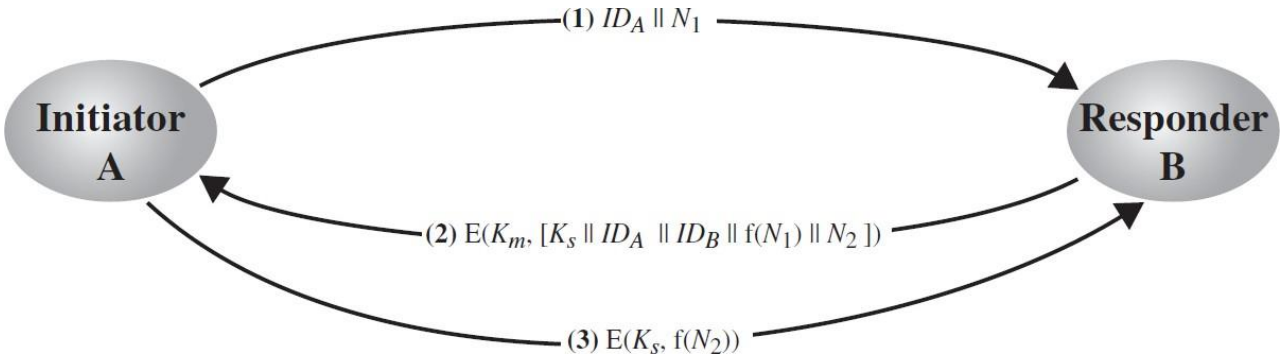


- The approach assumes that communication makes use of a connection-oriented end-to-end protocol, such as TCP.
- The steps involved in establishing a connection are shown in Figure.
- When one host wishes to set up a connection to another host, it transmits a connection request packet (step 1).
- The SSM saves that packet and applies to the KDC for permission to establish the connection (step 2).
- The communication between the SSM and the KDC is encrypted using a master key shared only by this SSM and the KDC.
- If the KDC approves the connection request, it generates the session key and delivers it to the two appropriate SSMs, using a unique permanent key for each SSM (step 3).

- The requesting SSM can now release the connection request packet, and a connection is set up between the two end systems (step 4).
- All user data exchanged between the two end systems are encrypted by their respective SSMs using the onetime session key.

Decentralized Key Control

- The use of a KDC imposes the requirement that the KDC be trusted and be protected.
- This requirement can be avoided if key distribution is fully decentralized.
- Although full decentralization is not practical for larger networks using symmetric encryption only, it may be useful within a local context.
- A decentralized approach requires that each end system be able to communicate in a secure manner with all potential partner end systems for purposes of session key distribution.
- Thus, there may need to be as many as master keys for a configuration with end systems.



- A session key may be established with the following sequence of steps.

1. A issues a request to B for a session key and includes a nonce, N_1.
2. B responds with a message that is encrypted using the shared master key. The response includes the session key selected by B, an identifier of B, the value $f(N_1)$, and another nonce, N_2.
3. Using the new session key, A returns $f(N_2)$ to B.

Controlling Key Usage

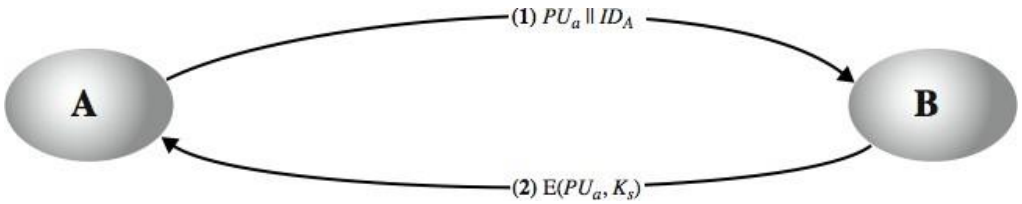
- The concept of a key hierarchy and the use of automated key distribution techniques greatly reduce the number of keys that must be manually managed and distributed.
- It also may be desirable to impose some control on the way in which automatically distributed keys are used.
- For example, in addition to separating master keys from session keys, we may wish to define different types of session keys on the basis of use, such as
  - Data-encrypting key, for general communication across a network
  - PIN-encrypting key, for personal identification numbers (PINs) used in electronic funds transfer and point-of-sale applications
  - File-encrypting key, for encrypting files stored in publicly accessible locations

Symmetric Key Distribution Using Asymmetric Encryption

One of the most important uses of a public-key cryptosystem is to encrypt secret keys for distribution.

Simple Secret Key Distribution

- If A wishes to communicate with B, the following procedure is employed:



1. A generates a public/private key pair 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.

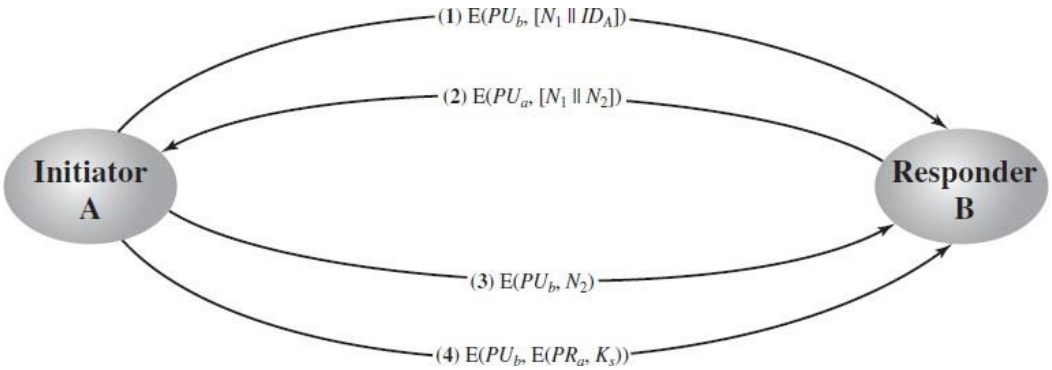
- A and B can now securely communicate using conventional encryption and the session key K_s.
- At the completion of the exchange, both A and B discard K_s.
- No keys exist before and after the communication.
- The protocol depicted is insecure against man-in-the-middle attack.

man-in-the-middle attack

- In this case, if an adversary, E, has control of the intervening communication channel, then E can compromise the communication in the following fashion without being detected.
  1) A generates a public/private key pair {PU_a, PR_a} and transmits a message intended for B consisting of PU_a and an identifier of A, ID_A.
  2) E intercepts the message, creates its own public/private key pair {PU_e, PR_e} and transmits $PU_e||ID_A$ to B.
  3) B generates a secret key, K_s, and transmits $E(PU_e, K_s)$.
  4) E intercepts the message and learns K_s by computing $D(PR_e, E(PU_e, K_s))$.
  5) E transmits $E(PU_a, K_s)$ to A.
- The result is that both A and B know K_s and are unaware that K_s has also been revealed to E.

Secret Key Distribution with Confidentiality and Authentication

- We assumed that A and B have exchanged public key.



- Then the following steps occur.
  1. A uses B's public key to encrypt a message to B containing an identifier of A (ID_A) and a nonce N_1, which is used to identify this transaction uniquely.
  2. B sends a message to A encrypted with PU_a and containing A's nonce N_1 as well as a new nonce generated by B N_2. Because only B could have decrypted message (1), the presence of N_1 in message (2) assures A that the correspondent is B.

3. A returns N_2, encrypted using B's public key, to assure B that its correspondent is A.
4. A selects a secret key K_s and sends $M=E(PU_b, E(PR_a, K_s))$ to B. Encryption of this message with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it.
5. B computes $D(PU_a, D(PR_b, M))$ to recover the secret key.

- Scheme ensures both confidentiality and authentication in the exchange of a secret key.

A Hybrid Scheme

- Yet another way to use public-key encryption to distribute secret keys is a hybrid approach in use on IBM mainframes.
- This scheme retains the use of a key distribution center (KDC) that shares a secret master key with each user and distributes secret session keys encrypted with the master key.
- A public key scheme is used to distribute the master keys.
- The following rationale is provided for using this three-level approach:
  - ✓ **Performance**
  - ✓ **Backward compatibility**

Distribution of Public Keys

Several techniques have been proposed for the distribution of public keys. Virtually all these proposals can be grouped into the following general schemes:

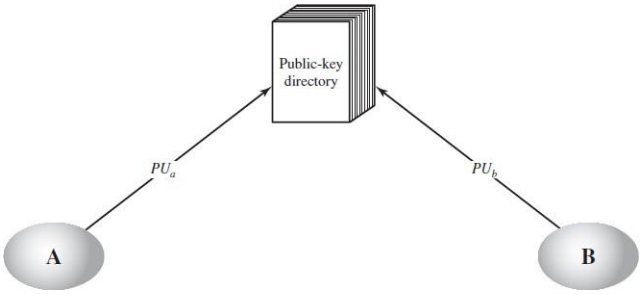
Public Announcement of Public Keys

- If there is some broadly accepted public-key algorithm, such as RSA, any participant can broadcast the key to the community at large.

- Although this approach is convenient, it has a major weakness.
- Some user could pretend to be user A and send a public key to another participant or broadcast such a public key.
- Until such time as user A discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A and can use the forged keys for authentication.

Publicly Available Directory

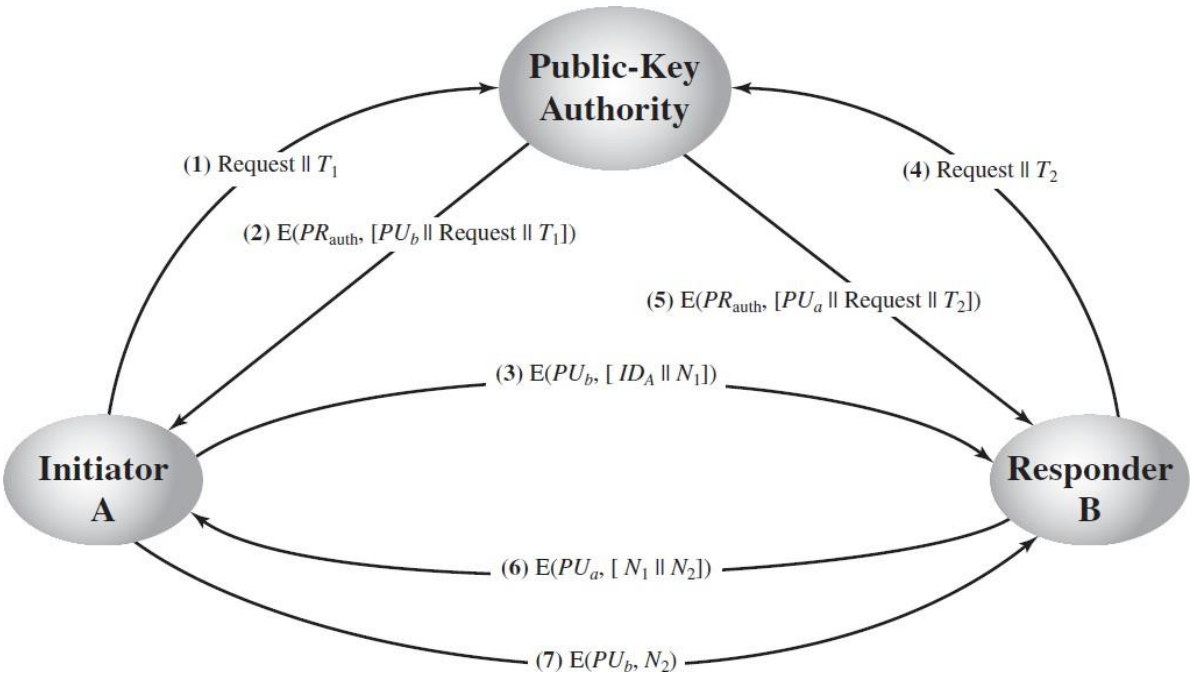
- A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys.
- Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization.



- Such a scheme would include the following elements:
  1. The authority maintains a directory with a {name, public key} entry for each participant.
  2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
  3. A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way.
  4. Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.
- This scheme is clearly more secure than individual public announcements but still has vulnerabilities.
- If anyone succeeds in obtaining the private key of the directory authority then he can pass false public key.
- Another way to achieve the same end is for the adversary to tamper with the records kept by the authority.

Public-Key Authority

- Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory.
- A typical scenario is illustrated in Figure.

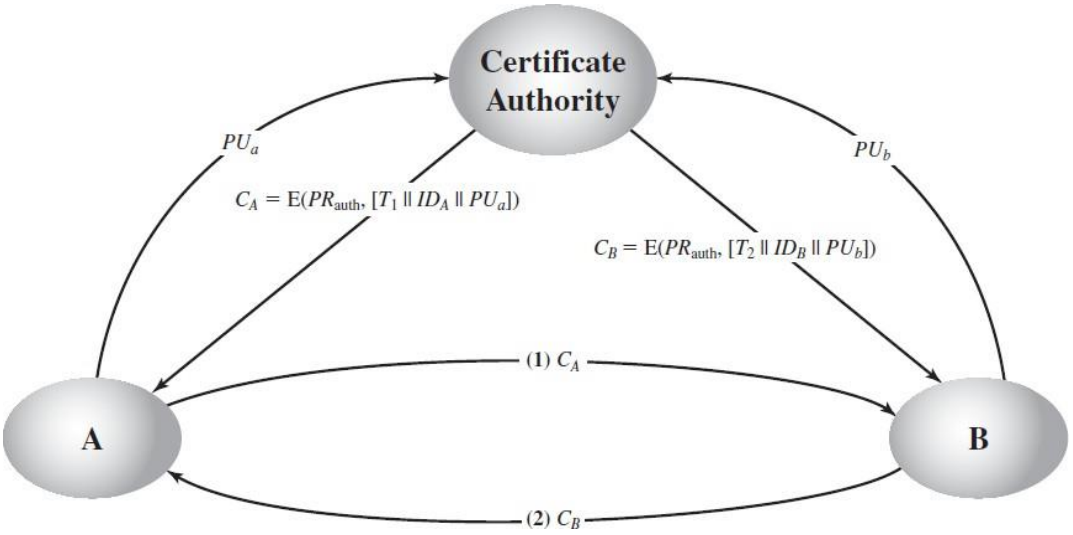


- Assumes that a central authority maintains a dynamic directory of public keys of all participants.
- Each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key.
- The following steps occur.
  1. A sends a timestamped message to the public-key authority containing a request for the current public key of B.

2. The authority responds with a message that is encrypted using the authority's private key, PR_{auth}.Thus,A is able to decrypt the message using the authority's public key. Therefore, A is assured that the message originated with the authority. The message includes the following:
   - B's public key, PU_b, which A can use to encrypt messages destined for B
   - The original request used to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority
   - The original timestamp given so A can determine that this is not an old message from the authority containing a key other than B's current public key
3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.
4. -Step 4 and 5 are similar to 2 and 3.
5. B retrieves A's public key from the authority in the same manner as A retrieved B's public key.
6. B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2). Because only B could have decrypted message (3), the presence of in message (6) assures A that the correspondent is B.
7. A returns N_2, which is encrypted using B's public key, to assure B that its correspondent is A.

Public-Key Certificates

- The directory of names and public keys maintained by the authority is vulnerable to tampering.
- An alternative approach, first suggested by Kohnfelder, is to use certificates.
- In essence, a certificate consists of a public key, an identifier of the key owner, and the whole block signed by a trusted third party.
- Typically, the third party is a certificate authority, such as a government agency or a financial institution that is trusted by the user community.
- A user can present his or her public key to the authority in a secure manner and obtain a certificate.
- The user can then publish the certificate. Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature.
- A participant can also convey its key information to another by transmitting its certificate.
- Other participants can verify that the certificate was created by the authority.
- We can place the following requirements on this scheme:

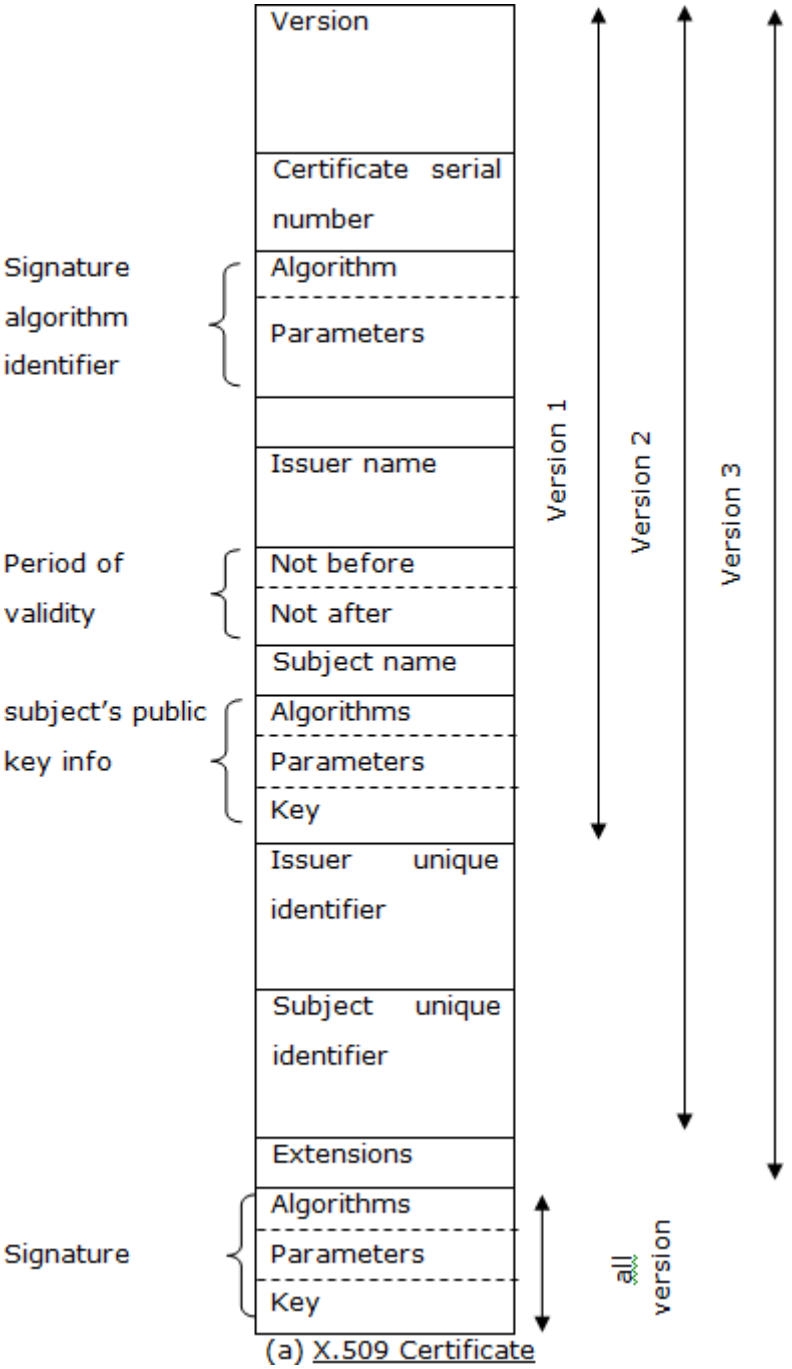


1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
3. Only the certificate authority can create and update certificates.
4. Any participant can verify the certificate.

X.509 Certificates

- X.509 provides authentication services and defines authentication protocols.
- X.509 uses X.500 directory which contains:
  - Public key certificates
  - Public key of users signed by certification authority
- X.509 certificate format is used in S/MIME, IP Security, and SSL/TLS.
- X.509 is based on the use of public-key cryptography (preferably RSA) and digital signatures.

X.509 includes the following elements.



(a) X.509 Certificate

- **Version**: Differentiates among successive versions of the certificate format; the default is version 1. Two other versions (2 and 3) are also available as shown in the figure.

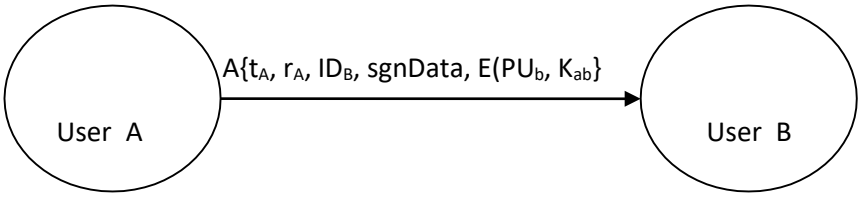
- **Serial number**: An integer value, unique within the issuing CA, different for each certificate.
- **Signature algorithm identifier**: The algorithm used to sign the certificate, together with any associated parameters.
- **Issuer name**: X.500 name of the CA that created and signed this certificate.
- **Period of validity**: Consists of two dates: the first and last on which the certificate is valid.
- **Subject name**: The name of the user to whom this certificate refers.
- **Subject's public-key information**: The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.
- **Issuer unique identifier**: An optional bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.
- **Subject unique identifier**: An optional bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.
- **Extensions**: A set of one or more extension fields.
- **Signature**: Covers all of the other fields of the certificate; it contains the hash code of the other fields, encrypted with the CA's private key. This field includes the signature algorithm identifier.

Authentication Procedures

- X.509 supports three types of authenticating using public key signatures. The types of authentication are
  1. One-way authentication
  2. Two- way authentication
  3. Three- way authentication

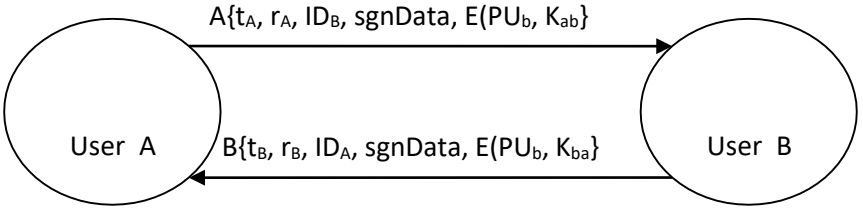
One-way authentication

- It involves single transfer of information from one user (say A) to other (B).
- This method authenticates the identity of A to B and the integrity of message.
- Here, message in the {} is signed by A.
- sgnData is the information that needs to be conveyed.
- $\mathbf{t_A}$ is timestamp and rA is the nonce.



Two- way authentication

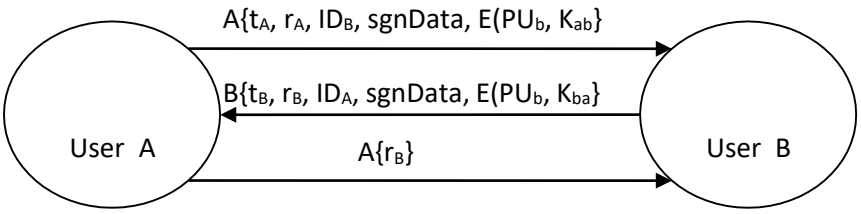
- Two- way authentication allows both parties to communicate and verify the identity of each other.



Three- way authentication

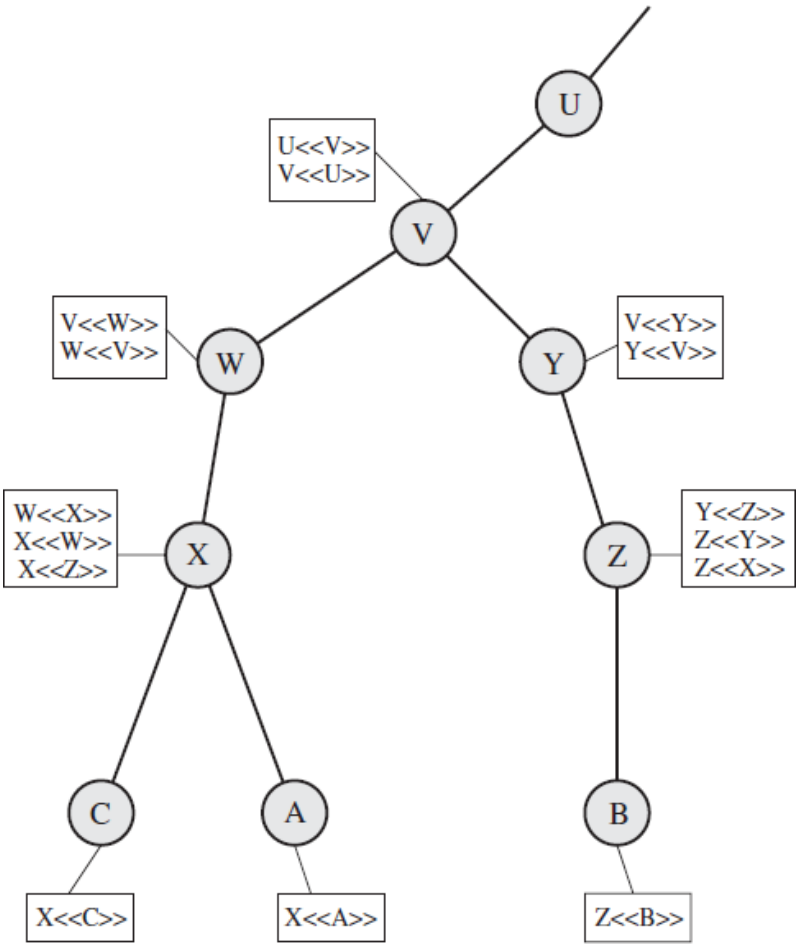
- Three- way authentication is used where synchronized clocks are not available.

- This method includes an additional message from A.



Obtaining Certificates in X.509

- Any user can verify a certificate if he has the public key of the CA that issued the certificate.
- Since certificates are unforgeable, they are simply stored in the directory.
- The directory entry for each CA includes two types of certificates:
  - Forward certificates: Certificates of X generated by other CAs.
  - Reverse certificates: Certificates generated by X that are the certificates of other CAs.
- Users subscribed to same CA can obtain certificate from the directory.
- A user may directly send the certificate to the user.
- However, multiple CAs are there and users subscribed to different CAs may want to communicate with each other.



- Suppose, A has obtained a certificate from certification authority X1 and B has obtained a certificate from CA X2.

- If A does not know the public key of X2, then B's certificate, issued by X2, is useless to A because A can read B's certificate, but A cannot verify the signature.
- But if the two CAs have securely exchanged their own public keys, the following procedure will enable A to obtain B's public key:
  - A obtains the certificate of X2 signed by X1 from the directory. A securely knows X1's public key, so A can obtain X2's public key from its certificate and verify X1's signature on the certificate.
  - A then obtains the certificate of B signed by X2. A now has a copy of X2's public key, so A can verify the signature and securely obtain B's public key.
  - In this case, A has used a chain of certificates to obtain B's public key. In the notation of X.509, this chain is expressed as:

X1<<X2>> X2 <<B>>

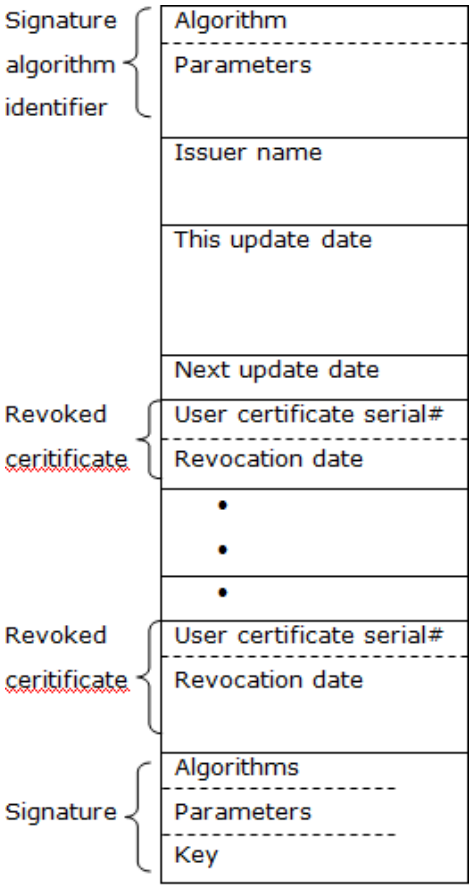
- Any level of hierarchy can be followed to produce a chain in this way. For example, in the figure given below, A can establish a certification path to B in the following way:

X<<W>> W <<V>> V <<Y>> <<Z>> Z <<B>>

- When A has obtained these certificates, it can decrypt the certification path in sequence to recover a copy of B's public key.
- Using this public key, A can send encrypted messages to B.
- If B requires A's public key, it can be obtained in the similar way.

Z<<Y>> Y <<V>> V <<W>> W <<X>>X <<A>>

Revocation of Certificates

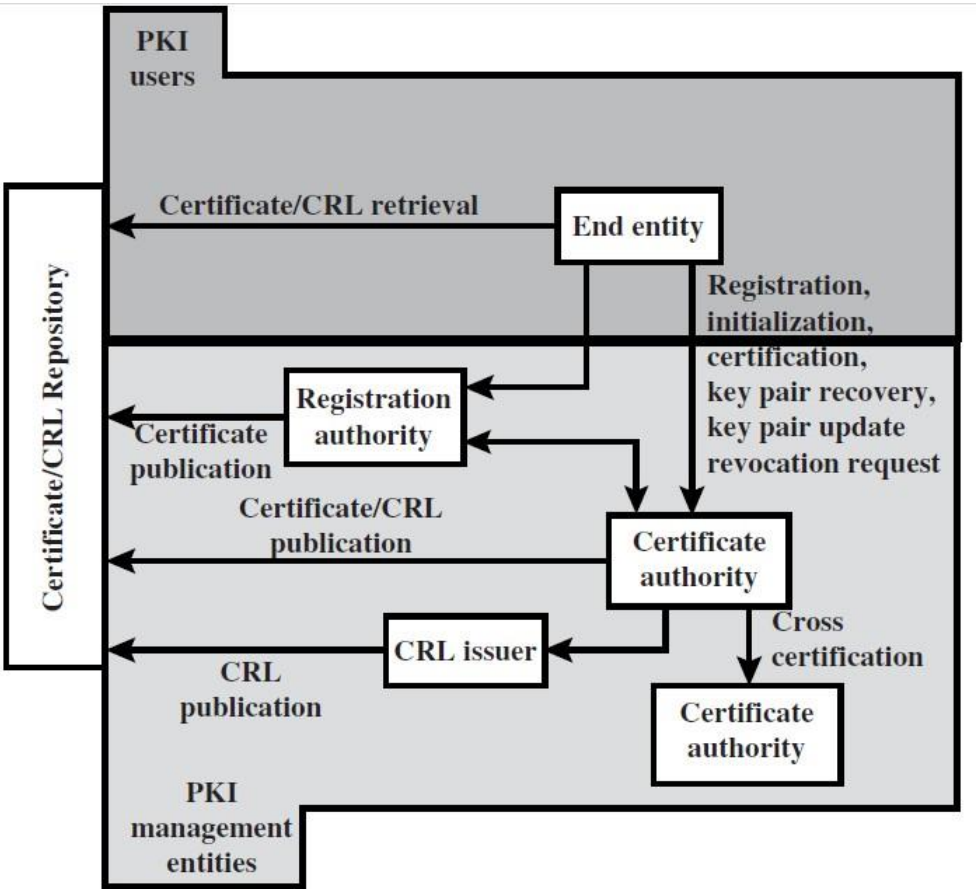


(b) Certificate Revocastion List

- The certificates have an expiry time.
- However, certificates need to be revoked if,
  - The user's private key has been compromised.
  - The user's certificate has been compromised.
  - The user is no longer certified by the CA.
- Each CA must maintain a list consisting of all revoked but not expired certificates issued by that CA, including both those issued to users and to other CAs.
- Each certificate revocation list (CRL) posted to the directory is signed by the issuer and includes
  - the issuer's name,
  - the date the list was created,
  - the date the next CRL is scheduled to be issued, and
  - an entry for each revoked certificate.
- The certificate revocation list is shown in the figure.
- Every user must check the CRL before using other user's public key.

Public-Key Infrastructure

- **Public-key infrastructure (PKI)** is the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.
- The principal objective for developing a PKI is to enable secure, convenient, and efficient acquisition of public keys.
- The Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) working group has been the driving force behind setting up a formal (and generic) model based on X.509.
- Figure shows the interrelationship among the key elements of the PKIX model.



- These elements are
  - **End entity:** A generic term used to denote end users, devices (e.g., servers, routers), or any other entity that can be identified in the subject field of a public key certificate.
  - **Certification authority (CA):** The issuer of certificates and (usually) certificate revocation lists (CRLs). It may also support a variety of administrative functions, although these are often delegated to one or more Registration Authorities.
  - **Registration authority (RA):** An optional component that can assume a number of administrative functions from the CA. The RA is often associated with the end entity registration process but can assist in a number of other areas as well.
  - **CRL issuer:** An optional component that a CA can delegate to publish CRLs.
  - **Repository:** A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by end entities.

PKIX Management Functions

- PKIX identifies a number of management functions that potentially need to be supported by management protocols which are:
  - Registration
  - Initialization
  - Certification
  - Key pair recovery
  - Key pair update
  - Revocation request
  - Cross certification

PKIX Management Protocols

- The PKIX working group has defines two alternative management protocols.
- RFC 2510 defines the certificate management protocols (CMP).
- Within CMP, each of the management functions is explicitly identified by specific protocol exchanges.
- CMP is designed to be a flexible protocol able to accommodate a variety of technical, operational, and business models.
- RFC 2797 defines certificate management messages over CMS (CMC).
- Where CMS refers to RFC 2630, and cryptographic message syntax.
- CMC is built on earlier work and is intended to leverage existing implementations.
- Although all of the PKIX functions are supported, the functions do not all map into specific protocol exchanges.

Remote user authentication is a mechanism in which the **remote** server verifies the legitimacy of a **user** over an insecure communication channel.

Remote User-Authentication using Symmetric Encryption

Mutual Authentication

- Two-level hierarchy of symmetric encryption keys can be used to provide confidentiality for communication in a distributed environment.
- In general, this strategy involves the use of a trusted key distribution center (KDC).
- Each party in the network shares a secret key, known as a master key, with the KDC.
- The KDC is responsible for generating keys to be used for a short time over a connection between two parties, known as session keys, and for distributing those keys using the master keys to protect the distribution.
- Proposal initially put forth by Needham and Schroeder for secret key distribution using a KDC includes authentication features.
- The protocol can be summarized as follows.
  1. $A \rightarrow KDC$: $ID_A||ID_B||N_1$
  2. $KDC \rightarrow A$: $E(K_a, [K_s||ID_B||N_1||E(K_b, [K_s||ID_A])])$
  3. $A \rightarrow B$: $E(K_b, [K_s||ID_A])$
  4. $B \rightarrow A$: $E(K_s, N_2)$
  5. $A \rightarrow B$: $E(K_s, f(N_2))$
- The protocol is still vulnerable to a form of replay attack.
- Suppose that an opponent, X, has been able to compromise an old session key.
- X can impersonate A and trick B into using the old key by simply replaying step 3.
- Unless B remembers indefinitely all previous session keys used with A, B will be unable to determine that this is a replay.
- If X can intercept the handshake message in step 4, then it can impersonate A's response in step 5.
- From this point on, X can send bogus messages to B that appear to B to come from A using an authenticated session key.
- Denning proposes to overcome this weakness by a modification to the Needham/Schroeder protocol that includes the addition of a timestamp to steps 2 and 3.
- Her proposal assumes that the master keys, K_a and K_b, are secure, and it consists of the following steps.
  1. $A \rightarrow KDC$: $ID_A||ID_B$
  2. $KDC \rightarrow A$: $E(K_a, [K_s||ID_B||T||E(K_b, [K_s||ID_A||T])])$
  3. $A \rightarrow B$: $E(K_b, [K_s||ID_A||T])$
  4. $B \rightarrow A$: $E(K_s, N_1)$
  5. $A \rightarrow B$: $E(K_s, f(N_1))$
- T is a timestamp that assures A and B that the session key has only just been generated.
- Thus, both A and B know that the key distribution is a fresh exchange.
- A and B can verify timeliness by checking that

$|Clock - T| < \Delta t_1 + \Delta t_2$

- Where Δt_1, is the estimated normal discrepancy between the KDC's clock and the local clock (at A or B) and Δt_2 is the expected network delay time.
- A new concern is raised: namely, that this new scheme requires reliance on clocks that are synchronized throughout the network points out a risk involved.
- The risk is based on the fact that the distributed clocks can become unsynchronized as a result of sabotage on or faults in the clocks or the synchronization mechanism.
- The problem occurs when a sender's clock is ahead of the intended recipient's clock. In this case, an opponent can intercept a message from the sender and replay it later when the timestamp in the message becomes current at the recipient's site.
- This replay could cause unexpected results.

- Gong refers to such attacks as **suppress-replay attacks**.
- One way to counter suppress-replay attacks is to enforce the requirement that parties regularly check their clocks against the KDC's clock.
- The other alternative, which avoids the need for clock synchronization, is to rely on handshaking protocols using nonces.
- This latter alternative is not vulnerable to a suppress-replay attack, because the nonces the recipient will choose in the future are unpredictable to the sender.
- The Needham/Schroeder protocol relies on nonces only but, as we have seen, has other vulnerabilities.
- Improved strategy was presented in this protocol is
  1. $A \rightarrow B$: $ID_A||N_a$
  2. $B \rightarrow KDC$: $ID_B||N_b||E(K_b, [ID_A||N_a||T_b])$
  3. $KDC \rightarrow A$: $E(K_a, [ID_B||N_a||K_s||T_b])||E(K_b, [ID_A||K_s||T_b])||N_b$
  4. $A \rightarrow B$: $E(K_b, [ID_A||K_s||T_b])||E(K_s, N_b)$
- This protocol provides an effective, secure means for A and B to establish a session with a secure session key.
- Furthermore, the protocol leaves A in possession of a key that can be used for subsequent authentication to B, avoiding the need to contact the authentication server repeatedly.
- Suppose that A and B establish a session using the aforementioned protocol and then conclude that session.
- Subsequently, but within the time limit established by the protocol, A desires a new session with B.
- The following protocol ensues:
  1. $A \rightarrow B$: $E(K_b, [ID_A||K_s||T_b])||N_a{}'$
  2. $B \rightarrow A$: $N_b{}'||E(K_s, N_a{}')$
  3. $A \rightarrow B$: $E(K_s, N_b{}')$
- When B receives the message in step 1, it verifies that the ticket has not expired.
- The newly generated nonces and assure each party that there is no replay attack.

One-Way Authentication

- With some refinement, the KDC strategy is a candidate for encrypted electronic mail.
- Because we wish to avoid requiring that the recipient (B) be on line at the same time as the sender (A), steps 4 and 5 must be eliminated.
- For a message with content , the sequence is as follows:
  1. $A \rightarrow KDC$: $ID_A||ID_B||N_1$
  2. $KDC \rightarrow A$: $E(K_a, [K_s||ID_B||N_1||E(K_b, [K_s||ID_A])])$
  3. $A \rightarrow B$: $E(K_b, [K_s||ID_A])||E(K_s, M)$
- This approach guarantees that only the intended recipient of a message will be able to read it.
- It also provides a level of authentication that the sender is A.
- The protocol does not protect against replays.

Kerberos

- Kerberos is an authentication protocol.
- It provides a way to authenticate clients to services to each other through a trusted third party.

Requirements of Kerberos

- **Secure**: Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
- **Reliable**: For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ distributed server architecture, with one system able to back up another.

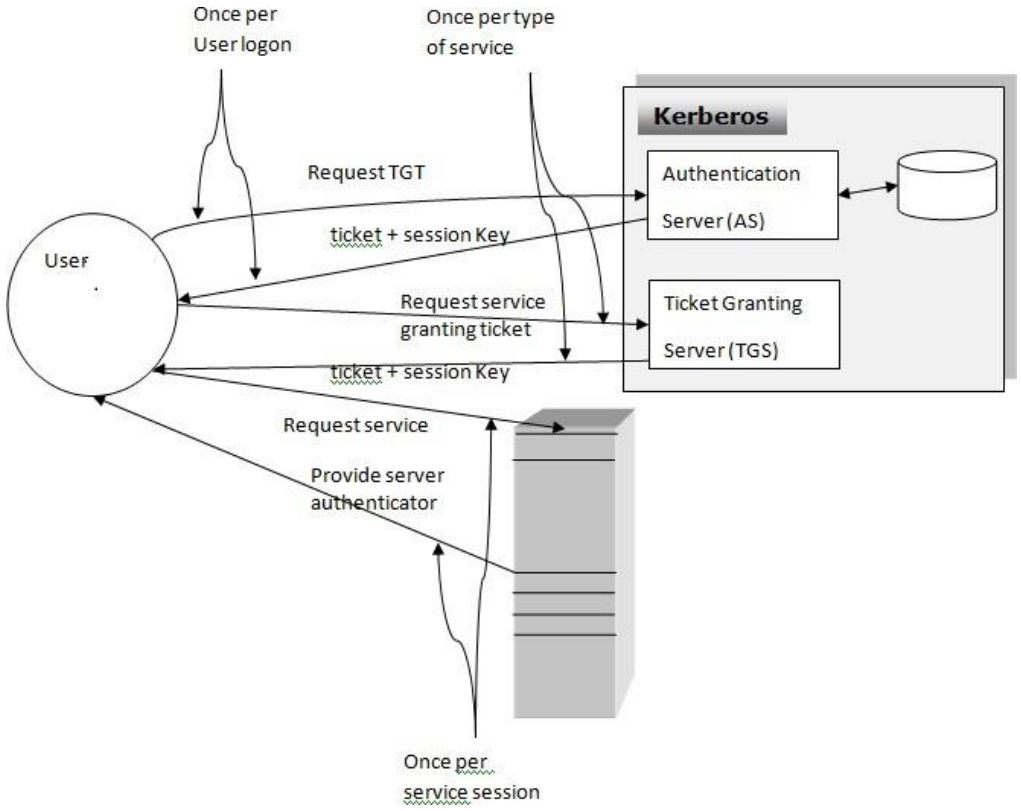
- **Transparent**: Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password.
- **Scalable**: The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture.

Kerberos protocol Terminology

1. **Authentication Server (AS)**: A server that issues tickets for a desired service which are in turn given to users for access to the service.
2. **Client**: An entity on the network that can receive a ticket from Kerberos.
3. **Credentials**: A temporary set of electronic credentials that verify the identity of a client for a particular service. It also called a ticket.
4. **Credential cache or ticket file**: A file which contains the keys for encrypting communications between a user and various network services.
5. **Crypt hash**: A one-way hash used to authenticate users.
6. **Key**: Data used when encrypting or decrypting other data.
7. **Key distribution center (KDC)**: A service that issue Kerberos tickets and which usually run on the same host as the ticket-granting server (TGS).
8. **Realm**: A network that uses Kerberos composed of one or more servers called KDCs and a potentially large number of clients.
9. **Ticket-granting server (TGS)**: A server that issues tickets for a desired service which are in turn given to users for access to the service. The TGS usually runs on the same host as the KDC.
10. **Ticket-granting ticket (TGT)**: A special ticket that allows the client to obtain additional tickets without applying for them from the KDC.

Overview of Kerberos

- The overview of Kerberos is shown and described below:



- User logs onto workstation and request on host for TGT.
- AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
- Workstation prompts the user for password and uses password to decrypt incoming message, then sends ticket and authentication that contain user's name, network address and time to TGS.
- TGS decrypts the ticket and authenticator, verifies request, then creates ticket for requested server.
- Workstation sends ticket and authentication to server.
- Server verifies that ticket and if the authenticator matches, then grants access to service. If mutual authentication is required, server returns an authenticator.
- The figure shows the above exchange.

Kerberos Authentication Dialogue

- The following message exchanges take place for authentication through Kerberos: **Authentication service exchange to obtain Ticket granting ticket (TGT)**
- This exchange takes place only once per a user logon session.
- User obtains a **Ticket Granting Ticket** from the authentication server. This ticket is sent to the **Ticket Granting Server** to obtain service tickets.

C → AS: IDc || IDtgs || TS1

AS → C: E(Kc, [Kc, tgs || IDtgs || TS2 || Lifetime2 || Tickettgs])

Tickettgs = E(Ktgs, [Kc, tgs || IDC || ADC ||IDtgs || TS2 || Lifetime2])

Ticket-Granting service exchange to obtain Service Granting Ticket

- This exchange takes place for each type of service.
- Here, the user presents the TGT to the ticket granting server. The TGS return a **Service Granting Ticket** to the user after proper authentication.
- An authenticator is added in the message which is encrypted using the key shared by the user and TGS

C → TGS: IDv || Tickettgs || Authenticatorc

TGS → C: E(Kc, tgs, [Kc, v || IDV || TS4 || Lifetime4 || Ticketv])

Ticketv = E(Kv, [Kc, v || IDC || ADC ||IDV || TS4 || Lifetime4])

Authenticatorc = E(Kc, tgs, [IDC ||ADC || TS3])

Client-Server authentication exchange to obtain service

- The user sends the Service Granting Ticket to the application server (of which the service is needed).
- The message also contains authenticator which proves the sender's identity to the server. Moreover, the server replies with the timestamp present in the authenticator. This authenticates the server to the user.

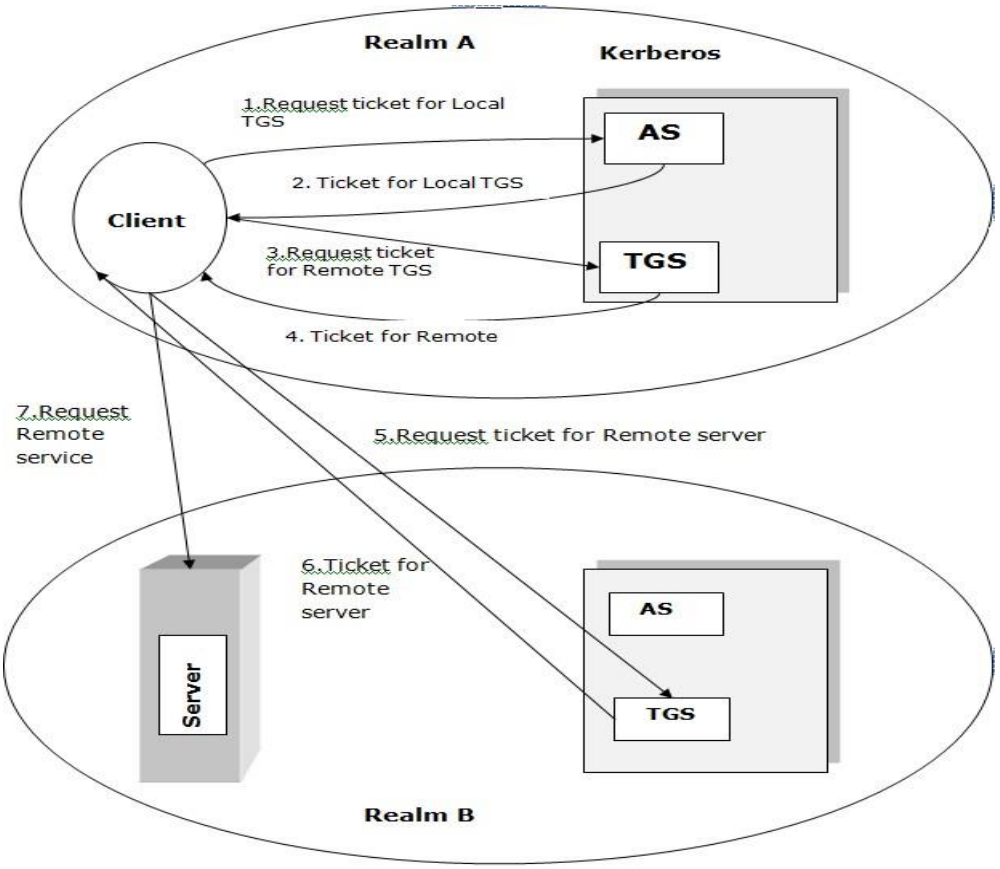
C → V: Ticketv || Authenticatorc

V → C: E(Kc, v, TS5 +1)

Authenticatorc = E(Kc, v, [IDC ||ADC || TS5])

Kerberos Realm

- A Kerberos realm is a set of managed nodes that share the same Kerberos database.
- The Kerberos database resides on the Kerberos master computer system, which should be kept in a physically secure room.
- A read-only copy of the Kerberos database might also reside on other Kerberos computer systems.
- However, all changes to the database must be made on the master computer system using Kerberos master password.
- A Kerberos principal, is a service or user that is known to the Kerberos system. Each Kerberos principal is identified by its principal name.
- Networks of clients and servers under different administrative organizations constitute different realms.
- For inter realm communication, the Kerberos servers in the two realms must be authenticated and registered to each other.



Request for Service in Another Realm

- A user wishing service on a server in another realm obtains a ticket for that server as given below:
  1. **C →AS: IDc||IDtgs||TS1**
  2. **AS→ C: E(Kc, [Kc,tgs||IDtgs||TS2||Lifetime2||Tickettgs]**
  3. **C→ TGS: IDtgsrem||Tickettgs||Authenticatorc**
  4. **TGS→ C: E(Kc,tgs, [Kc,tgsrem||IDtgsrem||TS4||Tickettgsrem])**
  5. **C→ TGSrem: IDvrem||Tickettgsrem||Authenticatorc**
  6. **TGSrem→ C: E(Kc,tgsrem , [Kc,vrem||IDvrem||TS6||Ticketvrem])**
  7. **C→ Vvrem: Ticketvrem||Authenticatorc**

  where **IDtgsrem** is the identity of remote TGS,
  Tickettgsrem is the TGT for remote TGS,
  IDvrem is the identity of remote server and
  Ticketvrem is the Service granting ticket for remote server.

Comparison between Kerberos Version 4 And Version 5

- The environmental shortcomings of Kerberos version 4 and their corresponding improvements in version 5 are listed below:
  1. **Encryption system dependence**: Version 4 requires the use of DES whereas version 5 includes a ciphertext tag with an encryption type identifier so that any encryption technique may be used.
  2. **Internet protocol dependence**: Version 4 requires the use of Internet Protocol (IP) addresses. Version 5 allows any network address type to be used.
  3. **Message byte ordering**: In version 4, the sender of a message employs a byte ordering of its own choosing but in version 5, all message structures are defined using Abstract Syntax Notation One (ASN.1) and Basic Encoding Rules (BER), which provide an unambiguous byte ordering.
  4. **Ticket lifetime**: Lifetime values in version 4 are encoded in an 8-bit, each unit of 5 minutes. Thus, the maximum lifetime that can be expressed is 28 x 5 = 1280 minutes, or a little over 21 hours. In version 5, tickets include an explicit start time and end time, allowing tickets with arbitrary lifetimes.
  5. **Authentication forwarding**: Version 4 does not allow credentials issued to one client to be forwarded to some other host and used by some other client. For example, a client issues a request to a print server that then, cannot access the client's file from a file server, using the client's credentials for access. Version 5 provides this capability.
  6. **Inter realm authentication**: In version 4, interoperability among realms requires many Kerberos-to-Kerberos relationships but version 5 supports a method that requires fewer relationships.
- Technical deficiencies of version 4 and its alternate in version 5 are:
  1. **Double encryption**: The tickets provided to clients are encrypted twice, once with the secret key of the target server and then again with a secret key known to the client. The second encryption is not necessary and is computationally wasteful.
  2. **PCBC encryption**: Encryption in version 4 makes use of a nonstandard mode of DES known as propagating cipher block chaining (PCBC) which is vulnerable to attack. Version 5 allows the standard CBC mode to be used for encryption.
  3. **Session keys**: Each ticket includes a session key used for encrypting messages. However, because the same ticket may be used repeatedly, replay attack is possible. In version 5, it is possible for a client and server to negotiate a subsession key, which is to be used only for that one connection.
  4. **Password attacks**: Both versions are vulnerable to a password attack. The message from the AS to the client includes material encrypted with a key based on the client's password. An opponent can capture this message and attempt to decrypt it by trying various passwords. Thus the opponent can discover the client's password and may subsequently use it to gain authentication credentials from Kerberos.

Remote User Authentication using Asymmetric Encryption

Mutual Authentication

- This protocol assumes that each of the two parties is in possession of the current public key of the other.
- A protocol using timestamps is:
  1. $A \rightarrow AS$: $ID_A||ID_B$
  2. $AS \rightarrow A$: $E(PR_{as}, [ID_A||PU_a||T])||E(PR_{as}, [ID_B||PU_b||T])$
  3. $A \rightarrow b$: $E(PR_{as}, [ID_A||PU_a||T])||E(PR_{as}, [ID_B||PU_b||T])||E(PU_b, E(PR_a, [K_s||T]))$
- In this case, the central system is referred to as an authentication server (AS), because it is not actually responsible for secret-key distribution.
- AS provides public-key certificates.
- The session key is chosen and encrypted by A; hence, there is no risk of exposure by the AS.
- The timestamps protect against replays of compromised keys.

- This protocol is compact but, as before, requires the synchronization of clocks.
- Another approach, proposed by Woo and Lam, makes use of nonces. The protocol consists of the following steps.
  1. $A \rightarrow KDC$: $ID_A||ID_B$
  2. $KDC \rightarrow A$: $E(PR_{auth}, [ID_B||PU_b])$
  3. $A \rightarrow B$: $E(PU_b, [N_a||ID_A])$
  4. $B \rightarrow KDC$: $ID_A||ID_B||E(PU_{auth}, N_a)$
  5. $KDC \rightarrow B$: $E(PR_{auth}, [ID_A||PU_a])||E(PU_b, E(PR_{auth}, [N_a||K_s||ID_B]))$
  6. $B \rightarrow A$: $E(PU_a, [E(PR_{auth}, [N_a||K_s||ID_B])||N_b])$
  7. $A \rightarrow B$: $E(K_s, N_b)$
- This seems to be a secure protocol that takes into account the various attacks.
- However, the authors themselves spotted a flaw and submitted a revised version of the algorithm:
  1. $A \rightarrow KDC$: $ID_A||ID_B$
  2. $KDC \rightarrow A$: $E(PR_{auth}, [ID_B||PU_b])$
  3. $A \rightarrow B$: $E(PU_b, [N_a||ID_A])$
  4. $B \rightarrow KDC$: $ID_A||ID_B||E(PU_{auth}, N_a)$
  5. $KDC \rightarrow B$: $E(PR_{auth}, [ID_A||PU_a])||E(PU_b, E(PR_{auth}, [N_a||K_s||ID_A||ID_B]))$
  6. $B \rightarrow A$: $E(PU_a, [E(PR_{auth}, [N_a||K_s||ID_A||ID_B])||N_b])$
  7. $A \rightarrow B$: $E(K_s, N_b)$
- The identifier of A, ID_A, is added to the set of items encrypted with the KDC's private key in steps 5 and 6.
- This binds the session key to the identities of the two parties that will be engaged in the session.
- This inclusion of accounts for the fact that the nonce value is considered unique only among all nonces generated by A, not among all nonces generated by all parties.

One-Way Authentication

- We have already presented public-key encryption approaches that are suited to electronic mail.
- These approaches require that either the sender know the recipient's public key (confidentiality), the recipient know the sender's public key (authentication), or both (confidentiality plus authentication).
- In addition, the public-key algorithm must be applied once or twice to what may be a long message.
- If confidentiality is the primary concern, then the following may be more efficient:

$A \rightarrow B$: $E(PU_b, K_s)||E(K_s, M)$

- In this case, the message is encrypted with a one-time secret key.
- A also encrypts this one-time key with B's public key.
- Only B will be able to use the corresponding private key to recover the one-time key and then use that key to decrypt the message.
- This scheme is more efficient than simply encrypting the entire message with B's public key.
- If authentication is the primary concern, then a digital signature may suffice:

$A \rightarrow B$: $M||E(PR_a, H(M))$

- This method guarantees that A cannot later deny having sent the message.
- However, this technique is not provide confidentiality.
- To counter such a problem, both the message and signature can be encrypted with the recipient's public key:

$A \rightarrow B$: $E(PU_b\ , [M||E(PR_a, H(M))])$

- The latter two schemes require that B know A's public key and be convinced that it is timely.
- An effective way to provide this assurance is the digital certificate. Now we have

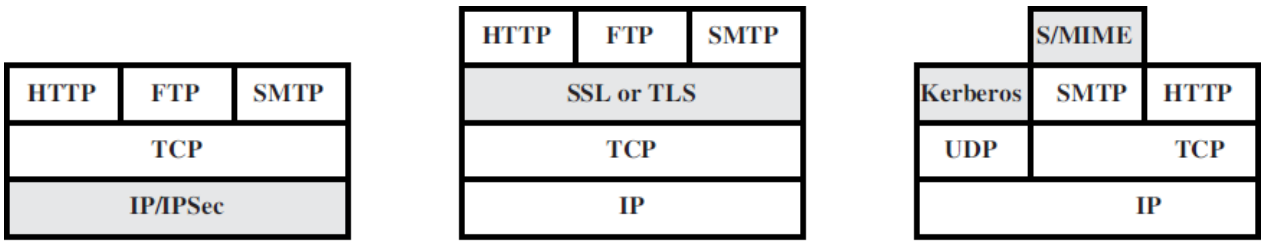
$A \rightarrow B$: $M||E(PR_a, H(M))||E(PR_{as}, [T||ID_A||PU_a])$

Web Security Threats

- The web provides the following threats which makes web security a must:
  - The Internet is two way. Even unimportant systems like electronic publishing systems, voice response, or fax-back are vulnerable to attacks on the Web servers over the Internet.
  - The Web is increasingly serving as a platform for corporate and product information and as the platform for business transactions. Reputations can be damaged and money can be lost if the Web servers are subverted.
  - Although Web browsers, web servers are very easy to use and manage and web content is easy to develop, the underlying software is extraordinarily complex. This complex software may hide many potential security flaws and hence is more vulnerable to a variety of security attacks.
  - A Web server can be exploited to gain access to data and systems not part of the Web itself but connected to the server at the local site.
  - Casual and untrained users are common clients for Web-based services. Such users are not always aware of the security risks.

	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

Web Traffic Security Approaches



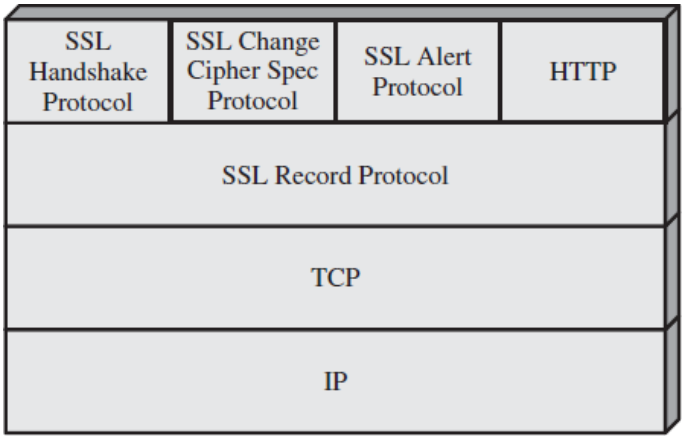
(a) Network level (b) Transport level (c) Application level

- Figure illustrates that one way to provide Web security is to use IP security (IPsec).
- The advantage of using IPsec is that it is transparent to end users and applications and provides a general-purpose solution.
- Furthermore, IPsec includes a filtering capability so that only selected traffic need incur the overhead of IPsec processing.
- Another relatively general-purpose solution is to implement security just above TCP.
- The foremost example of this approach is the Secure Sockets Layer (SSL) and the follow-on Internet standard known as Transport Layer Security (TLS).
- At this level, there are two implementation choices.
- For full generality, SSL (or TLS) could be provided as part of the underlying protocol suite and therefore be transparent to applications.
- Alternatively, SSL can be embedded in specific packages.
- For example, Netscape and Microsoft Explorer browsers come equipped with SSL, and most Web servers have implemented the protocol.
- Application-specific security services are embedded within the particular application.
- Figure shows examples of this architecture.
- The advantage of this approach is that the service can be tailored to the specific needs of a given application.

Secure Socket Layer

SSL Architecture

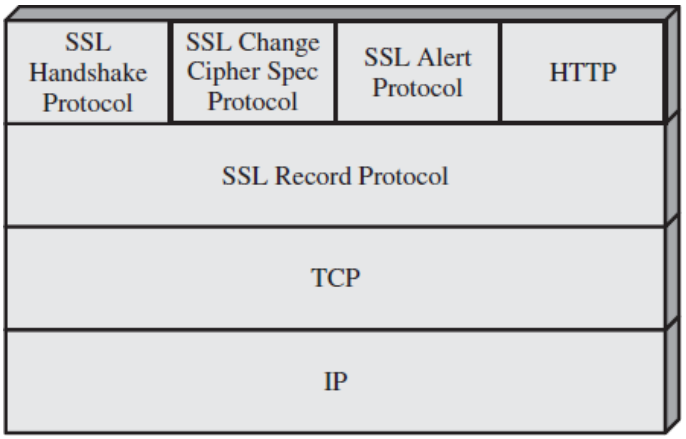
- SSL is designed to make use of TCP to provide a reliable end-to-end secure service.
- SSL is not a single protocol but rather two layers of protocols, as illustrated in Figure below.



- 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**.
- 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 that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. The connections are transient. Every connection is associated with one session.
  - **Session:** An SSL session is an association between a client and a server.
- There are a number of states associated with each session. Once a session is established, there is a current operating state for both read and write (i.e., receive and send).
- In addition, during the Handshake Protocol, pending read and write states are created. Upon successful conclusion of the Handshake Protocol, the pending states become the current states.
- A session state is defined by the following parameters.
  - **Session identifier**: A random byte sequence chosen by the server to identify an active or resumable session state.
  - **Peer certificate**: An X509.v3 certificate of the peer. It may be null.
  - **Compression method**: The algorithm used to compress data.
  - **Cipher spec**: Specifies the data encryption algorithm (such as null, AES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation.
  - **Master secret**: 48-byte secret shared between the client and server.
  - **Is resumable**: A flag indicating whether or not 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 conventional encryption key for data encrypted by the server and decrypted by the client.
  - **Client write key**: The conventional 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 initialized by the SSL Handshake Protocol.
  - **Sequence numbers**: Each party maintains separate sequence numbers for transmitted and received messages for each connection. When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero.

SSL Protocol

- SSL protocol is implemented just above the TCP to provide web security.
- SSL is designed to make use of TCP to provide a reliable end-to-end secure service.
- SSL is not a single protocol but two layers of protocols.
- The SSL Record Protocol provides basic security services to various higher layer protocols.

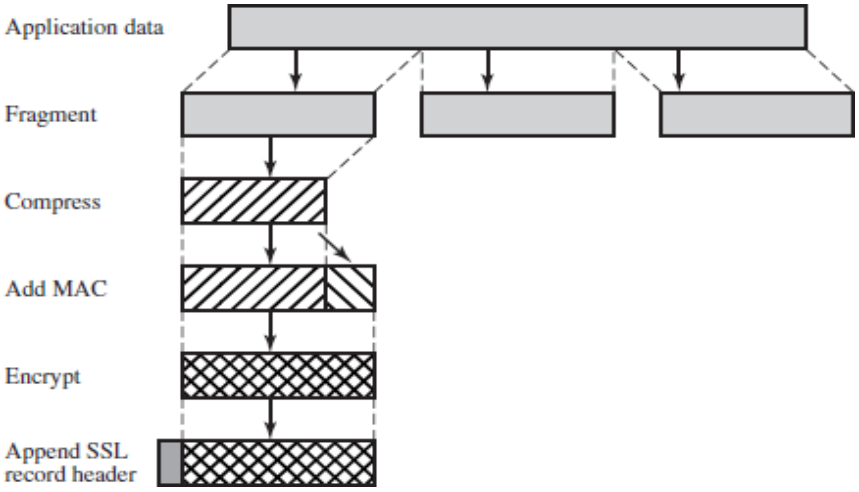


SSL Protocol stack

- Three higher-layer protocols are defined in SSL:
  - The Handshake Protocol
  - The Change Cipher Spec Protocol
  - The Alert Protocol

SSL Record Protocol

- The SSL Record Protocol provides two services for SSL connections: **Confidentiality** and **Message Integrity**.



SSL Record protocol Operation

- The overall operation of Record Protocol is:
  - **Fragmentation**: Each upper-layer message is fragmented into blocks of 214 bytes (16384 bytes) or less.
  - **Compression**: Compression is optionally applied. Compression must be lossless and may not increase the content length by more than 1024 bytes.
  - **Add message authentication code**: MAC is calculated over the compressed data by the following expression.

    (optional){MAC = hash(MAC_write_secret || pad_2|| hash(MAC_write_secret || pad_1|| seq_num|| SSLCompressed.type|| SSLCompressed.length || SSLCompressed.fragment))

where

|| = concatenation,

MAC_write_secret = shared secret key,

hash = cryptographic hash algorithm,

pad_1 = the byte 0x36 (0011 0110),

pad_2 = the byte 0x5C (0101 1100),

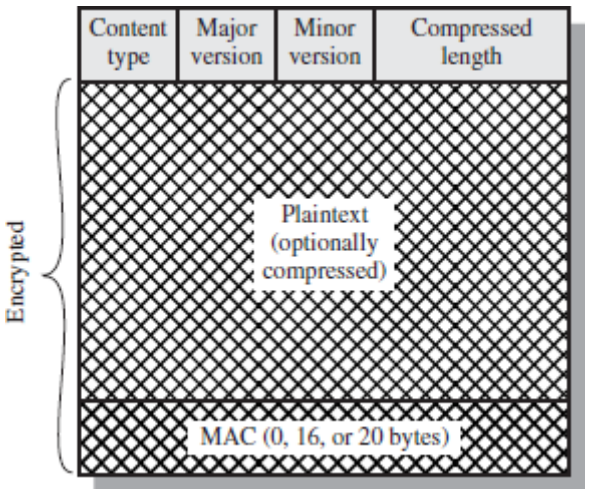
seq_num = the sequence number for this message

SSLCompressed.type = the higher-level protocol used to process this fragment

SSLCompressed.length = the length of the compressed fragment

SSLCompressed.fragment = the compressed fragment or plain text (if compression is not used) }

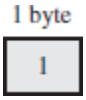
- **Encryption**: The compressed message plus the MAC are encrypted using symmetric encryption. Algorithms supported are AES, RC4-40, IDEA, RC2, DES, 3DES and Fortezza.
- **Add SSL Header**: A header is prepared and added to the message. The header consists of the following fields:
  - ✓ **Content Type (8 bits)**: The higher-layer protocol used to process the fragment.
  - ✓ **Major Version (8 bits)**: Indicates major version of SSL in use. For SSLv3, the value is 3.
  - ✓ **Minor Version (8 bits)**: Indicates minor version in use. For SSLv3, the value is 0.
  - ✓ **Compressed Length (16 bits)**: The length in bytes of the fragment.



SSL Record format

Change Cipher Spec Protocol

- This protocol consists of a single message of a single byte with the value 1.

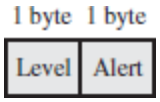


Change cipher spec protocol

- The purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher suite to be used on this connection.

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:
  - The first byte takes the value warning (1) or fatal (2) to convey the severity of the message.
  - The second byte contains a code that indicates the specific alert.



Alert protocol

- If the level is fatal, SSL immediately terminates the connection. Other connections on the same session may continue, but no new connections are established.
- Some of the alerts of fatal types are **unexpected_message, bad_record_mac, decompression_failure** etc.
- Alerts of level warning include **close_notify, no_certificate, bad_certificate** etc.

Handshake Protocol

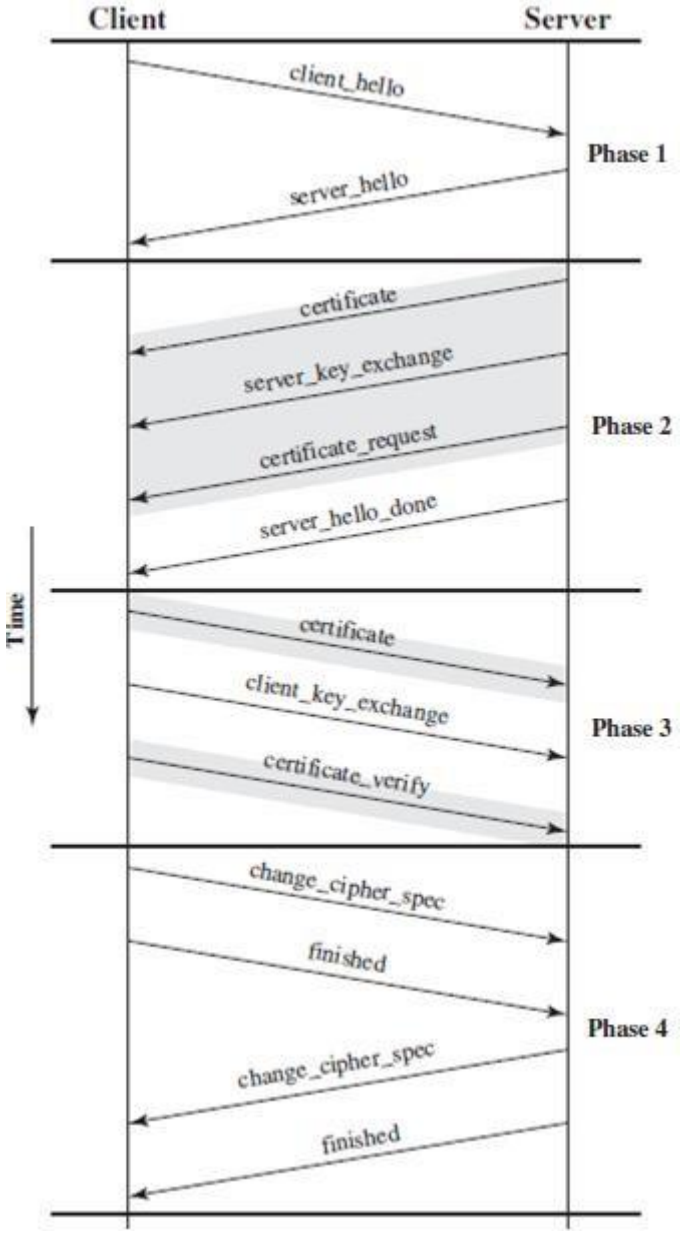
- This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys.
- The Handshake Protocol is used before any application data is transmitted.
- A handshake message has the following format:
  - **Type (1 byte)**: Indicates one of 10 messages of handshake protocol.
  - **Length (3 bytes)**: The length of the message in bytes.
  - **Content (bytes)**: The parameters associated with this message.

Handshake protocol

- The algorithm has four phases.
- **Phase 1. Establish Security Capabilities**: This phase is used to initiate a logical connection and to establish the security capabilities that will be associated with it.
  - 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 number which serves as nonce.
    - ✓ **Session ID**: A variable-length session identifier. A nonzero value indicates that the client wishes to update the parameters of an existing 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 cryptographic algorithms (key exchange, encryption and MAC) supported by the client, in decreasing order of preference.
    - ✓ **Compression Method**: This is a list of the compression methods the client supports.

- After sending the client_hello message, the client waits for the server_hello message, which contains the same parameters as the client_hello message. The parameters contains the values which client had sent to the server and the server has chosen to use.

- **Phase 2: Server Authentication and Key Exchange**: This phase provides authentication of server to the client.
  - The server sends its certificate (one or more) if it needs to be authenticated.
  - The server sends a **server_key_exchange** message which contains the list of secret keys to be used for the subsequent data.
  - The **certificate_request** message is sent next which includes two parameters: certificate_type and certificate_authorities.
  - The final message in phase 2, and one that is always required, is the **server_done** message, which is sent by the server to indicate the end of the server hello and associated messages.
  - After sending this message, the server will wait for a client response. This message has no parameters.
- **Phase 3. Client Authentication and Key Exchange**: This phase provides client authentication to the server.
  - The client verifies the server certificates and checks whether the server_hello parameters are acceptable.
  - If all is satisfactory, the client sends a **certificate** message if the server has requested a certificate. If no suitable certificate is available, the client sends a no_certificate alert.
  - Next is the **client_key_exchange** message which has the same parameters as the server_key_exchange message.
  - The client may send a **certificate_verify** message to provide explicit verification of a client certificate. The client encrypts all the previous messages and master secret with its private key.
- **Phase 4. Finish**: This phase completes the setting up of a secure connection.
  - The client sends a **change_cipher_spec** message and copies the pending **CipherSpec** into the current **CipherSpec**.
  - The client then immediately sends the **finished** message.
  - The server sends its own **change_cipher_spec** message, transfers the pending to the current **CipherSpec**, and sends its **finished** message.
- At this point, the handshake is complete and the client and server may begin to exchange application-layer data.



Handshake protocol message exchange

Cryptographic Computations

- Two further items are of interest:
  - The creation of a shared master secret by means of the key exchange
    - ✓ The shared master secret is a 48-byte value unique to this session.
      - ✓ First, a pre_master_secret is exchanged.
      - ✓ Then, the master_secret is calculated by both parties.
  - The generation of cryptographic parameters from the master secret.
    - ✓ The parameters include a client write MAC secret, a server write MAC secret, a client write key, a server write key, a client write IV, and a server write IV, which are generated from the master secret.
    - ✓ These parameters are generated from the master secret by hashing the master secret into a sequence of secure bytes of sufficient length for all needed parameters.

Transport Layer Security (TLS)

- TLS is an IETF standardization initiative whose goal is to produce an Internet standard version of SSL.
- TLS is defined as a Proposed Internet Standard in RFC 5246. Which is very similar to SSLv3.
- We highlight the differences.

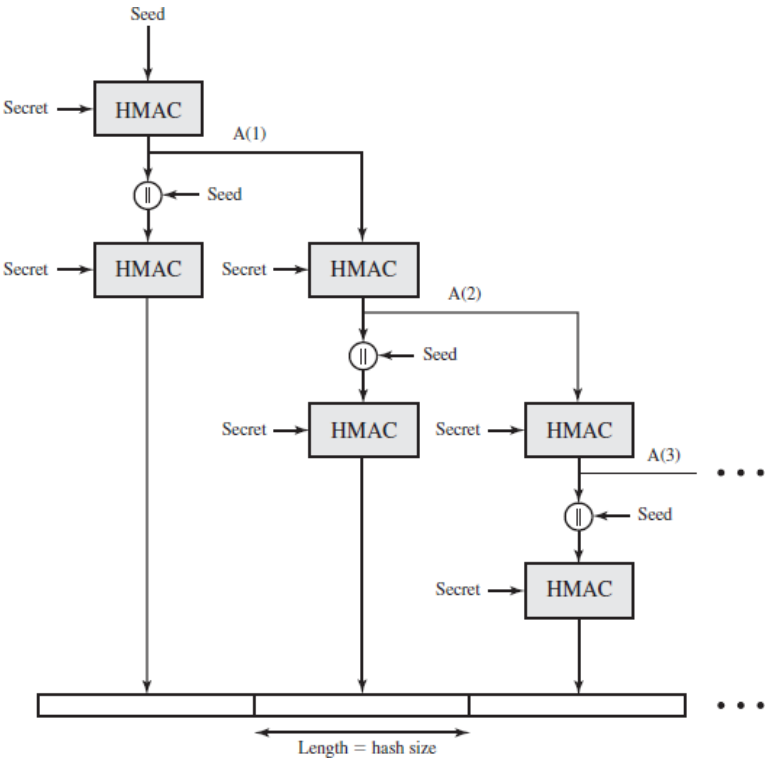
Version Number

- 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

- There are two differences between the SSLv3 and TLS MAC schemes:
- The actual algorithm and the scope of the 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.
- The level of security should be about the same in both cases.
- For TLS, the MAC calculation encompasses the fields indicated in the following expression: `MAC(MAC_write_secret, seq_num || TLSCompressed.type || TLSCompressed.version || TLSCompressed.length || TLSCompressed.fragment)`
- The MAC calculation covers all of the fields covered by the SSLv3 calculation, plus the field `TLSCompressed.version`, which is the version of the protocol being employed.

Pseudorandom Function



- TLS makes use of a pseudorandom function referred to as PRF to expand secrets into blocks of data for purposes of key generation or validation.

- The objective is to make use of a relatively small shared secret value but to generate longer blocks of data in a way that is secure from the kinds of attacks made on hash functions and MACs.
- The PRF is based on the data expansion function (Figure) given as
  `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))
- PRF is defined as
  `PRF(secret, label, seed) = P_hash(S1,label || seed)`
- PRF takes as input a secret value, an identifying label, and a seed value and produces an output of arbitrary length.

Alert Codes

- TLS supports all of the alert codes defined in SSLv3 with the exception of `no_certificate`.
- A number of additional codes are defined in TLS; of these, the following are always fatal.
  - **record_overflow**
  - **unknown_ca**
  - **access_denied**
  - **decode_error**
  - **protocol_version**
  - **insufficient_security**
  - **unsupported_extension**
  - **internal_error**
  - **decrypt_error**
  - The remaining alerts include the following.
  - **user_canceled**
  - **no_renegotiation**

Cipher Suites

- There are several small differences between the cipher suites available under SSLv3 and under TLS:
- **Key Exchange:** TLS supports all of the key exchange techniques of SSLv3 with the exception of Fortezza.
- **Symmetric Encryption Algorithms:** TLS includes all of the symmetric encryption algorithms found in SSLv3, with the exception of Fortezza.

Client Certificate Types

- TLS defines the following certificate types to be requested in a `certificate_request` message: `rsa_sign`, `dss_sign`, `rsa_fixed_dh`, and `dss_fixed_dh`.
- These are all defined in SSLv3. In addition, SSLv3 includes `rsa_ephemeral_dh`, `dss_ephemeral_dh`, and `fortezza_kea`.
- Ephemeral Diffie-Hellman involves signing the Diffie-Hellman parameters with either RSA or DSS.
- For TLS, the `rsa_sign` and `dss_sign` types are used for that function; a separate signing type is not needed to sign Diffie-Hellman parameters.
- TLS does not include the Fortezza scheme.

certificate_verify and Finished Messages

- In the TLS `certificate_verify` message, the MD5 and SHA-1 hashes are calculated only over `handshake_messages`.
- The hash calculation also included the master secret and pads.
- These extra fields were felt to add no additional security.
- As with the finished message in SSLv3, the finished message in TLS is a hash based on the shared `master_secret`, the previous handshake messages, and a label that identifies client or server.

Cryptographic Computations

- The `pre_master_secret` for TLS is calculated in the same way as in SSLv3.
- As in SSLv3, the `master_secret` in TLS is calculated as a hash function of the `pre_master_secret` and the two hello random numbers.
- The form of the TLS calculation is different from that of SSLv3 and is defined as `master_secret=PRF(pre_master_secret,"master secret",ClientHello.random ||ServerHello.random)`
- The algorithm is performed until 48 bytes of pseudorandom output are produced.
- The calculation of the key block material (MAC secret keys, session encryption keys, and IVs) is defined as `key_block = PRF(master_secret,"key expansion", SecurityParameters. server_random || SecurityParameters.client_random)`
- As with SSLv3, the `key_block` is a function of the `master_secret` and the client and server random numbers, but for TLS, the actual algorithm is different.

Padding

- In SSL, the padding added prior to encryption of user data is the minimum amount required so that the total size of the data to be encrypted is a multiple of the cipher's block length.
- In TLS, the padding can be any amount that results in a total that is a multiple of the cipher's block length, up to a maximum of 255 bytes.
- A variable padding length may be used to frustrate attacks based on an analysis of the lengths of exchanged messages.

HTTPS

- HTTPS (HTTP over SSL) refers to the combination of HTTP and SSL to implement secure communication between a Web browser and a Web server.
- The HTTPS capability is built into all modern Web browsers. Its use depends on the Web server supporting HTTPS communication.
- For example, search engines do not support HTTPS.
- The principal difference seen by a user of a Web browser is that URL (uniform resource locator) addresses begin with https:// rather than http://.
- A normal HTTP connection uses port 80. If HTTPS is specified, port 443 is used, which invokes SSL.
- When HTTPS is used, the following elements of the communication are encrypted:
  - URL of the requested document
  - Contents of the document
  - Contents of browser forms (filled in by browser user)
  - Cookies sent from browser to server and from server to browser
  - Contents of HTTP header

- There is no fundamental change in using HTTP over either SSL or TLS, and both implementations are referred to as HTTPS.

Connection Initiation

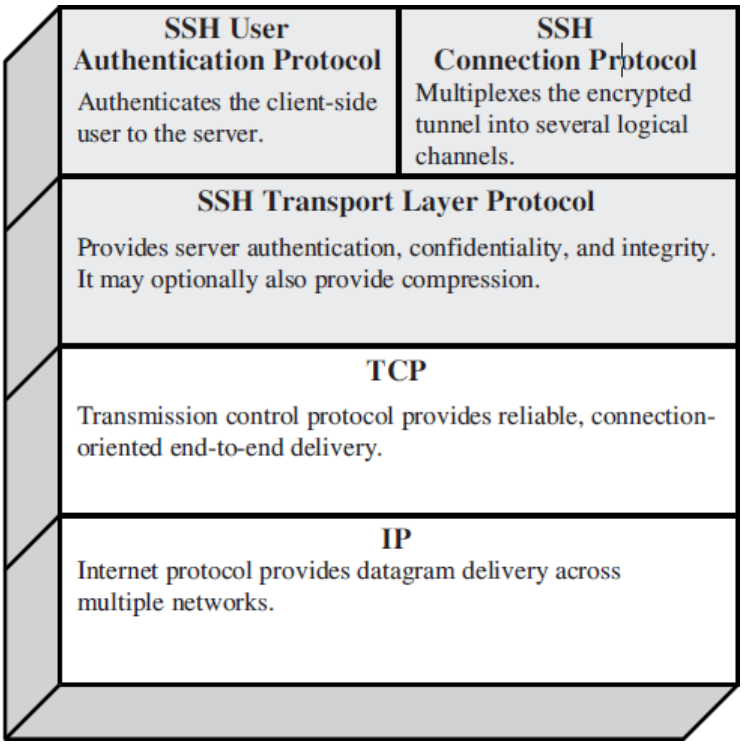
- The client initiates a connection to the server on the appropriate port and then sends the TLS ClientHello to begin the TLS handshake.
- When the TLS handshake has finished, the client may then initiate the first HTTP request.
- All HTTP data is to be sent as TLS application data.
- Normal HTTP behavior, including retained connections, should be followed.
- We need to be clear that there are three levels of awareness of a connection in HTTPS.
  - At the HTTP level
  - At the level of TLS
  - At the level of TCP

Connection Closure

- An HTTP client or server can indicate the closing of a connection by including the following line in an HTTP record: `Connection: close`.
- This indicates that the connection will be closed after this record is delivered.
- At the TLS level, the proper way to close a connection is for each side to use the TLS alert protocol to send a `close_notify` alert.
- TLS implementations must initiate an exchange of closure alerts before closing a connection.
- A TLS implementation may, after sending a closure alert, close the connection without waiting for the peer to send its closure alert, generating an “incomplete close”.
- Note that an implementation that does this may choose to reuse the session.
- This should only be done when the application knows (typically through detecting HTTP message boundaries) that it has received all the message data that it cares about.
- HTTP clients also must be able to cope with a situation in which the underlying TCP connection is terminated without a prior `close_notify` alert and without a `Connection: close` indicator.
- Such a situation could be due to a programming error on the server or a communication error that causes the TCP connection to drop.
- However, the unannounced TCP closure could be evidence of some sort of attack.
- So the HTTPS client should issue some sort of security warning when this occurs.

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 and e-mail.
- A new version, SSH2, fixes a number of security flaws in the original scheme.
- 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 is rapidly becoming one of the most pervasive applications for encryption technology outside of embedded systems.
- SSH is organized as three protocols that typically run on top of TCP (Figure):



- **SSH Transport Layer Protocol:** Provides server authentication, data confidentiality, and data integrity with forward secrecy (i.e., if a key is compromised during one session, the knowledge does not affect the security of earlier sessions).The transport layer may optionally provide compression.
- **SSH User Authentication Protocol:** Authenticates the user to the server.
- **SSH Connection Protocol:** Multiplexes multiple logical communications channels over a single, underlying SSH connection.