Lecture 23

With the increasing use of internet, security has become a major concern for organization. Things have even worsened, with the advent of Wireless Network; where a lot of research is going on to manage security of the network.

Presented below are some of the issues that the network security has to deal with

1. From outside user
2. From Inside user
3. Guarantying that the data transmission is authentic i.e. from a reliable source
4. To protect data during transmission

We will be discussing network security revolving around two topics

1. Authentication
2. Cryptography

Security Requirements

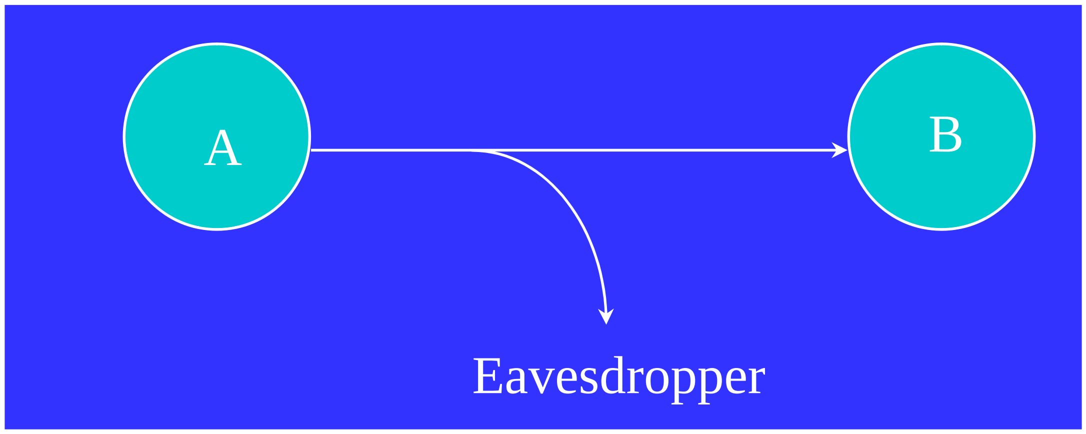
Following are the security requirements

Confidentiality: Requires that data only be accessible for reading by authorized parties.
Integrity: Requires that data can be modified only by authorized parties.

If someone tries to breach the security, we call such an event as attacks. Attacks have been classified as **Passive attack** and **Active attack**

Passive attack, as the name suggest, is an indirect on the security. If someone tries to access the data during transmission or someone tries to analyze the traffic behavior and thus be able to get the data, are the two types of passive attacks. First one is called as **Release of Message Contents** and the second one is called as **Traffic Analysis.**

To understand Release of message contents, imaging that user A sends some series of messages to user B, another user C is listening to the link, and thus gets a copy of all the messages from A and B, the confidentiality of message is not maintained, such an attack is one of the passive attack. Possible solution is encryption.



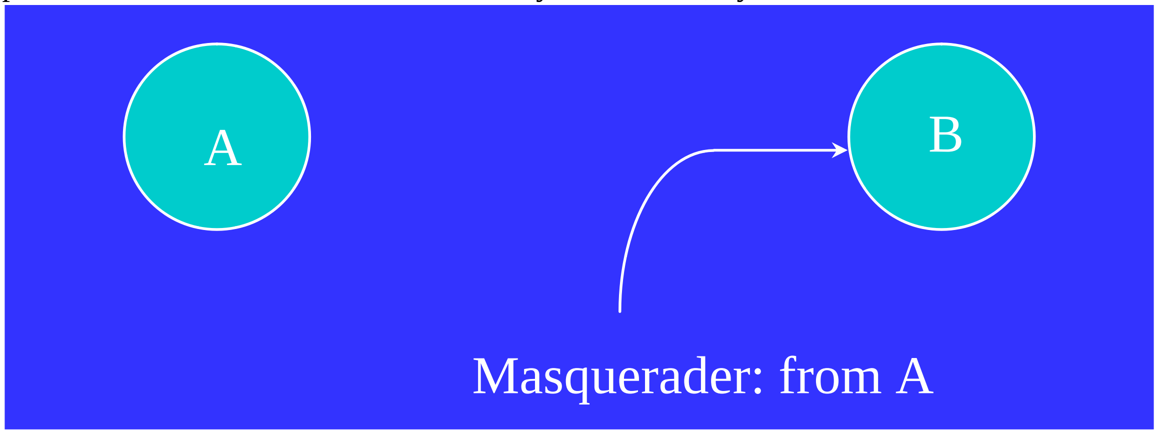
The second attack, Traffic Analysis, is more difficult for the attacker, suppose user A before transmitting masks the messages, such that even if the attacker is able to capture the message, he cannot extract information out of it (The common technique for masking the message is encryption). But the attacker can do the analysis of the masked message, as he can still access the masked (encrypted) message. By doing analysis of the masked message, he can somehow generate the actual message (unmasked message). Possible solution is good encryption algorithms, which is difficult to analyze, or which can be analyzed but requires lot of resources that are infeasible.

<u>Active Attack</u>, as the name suggest is kind of direct attack on the users and their data by doing some modification to the message or creating a false message thus destroying the authenticity of the message. We can further classify the active attack into three types

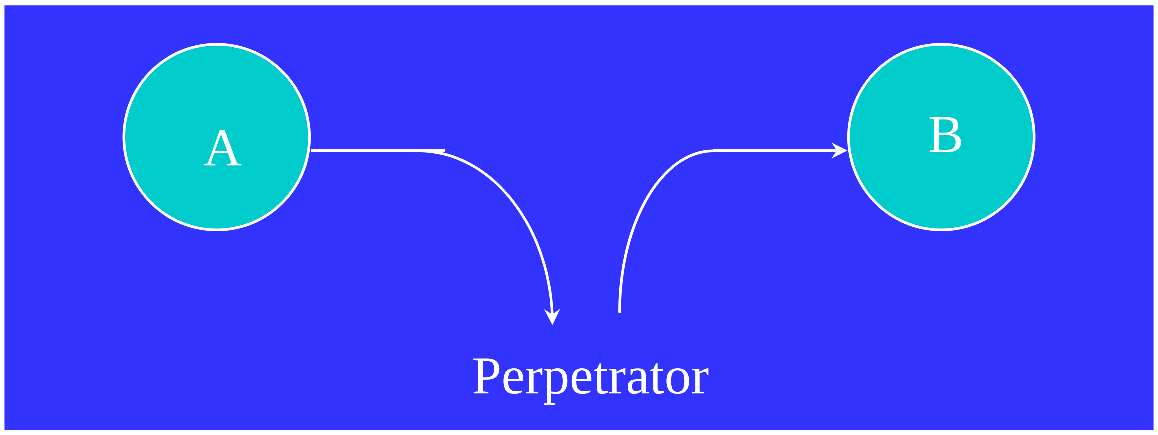
1. Masquerade
2. Replay
3. Modification of Message
4. Denial of Service

A **<u>masquerade</u>** takes place when one entity pretends to be a different entity. This can only be detected, by authentication. For example, User C sends a file to User B, pretending it is from User B.

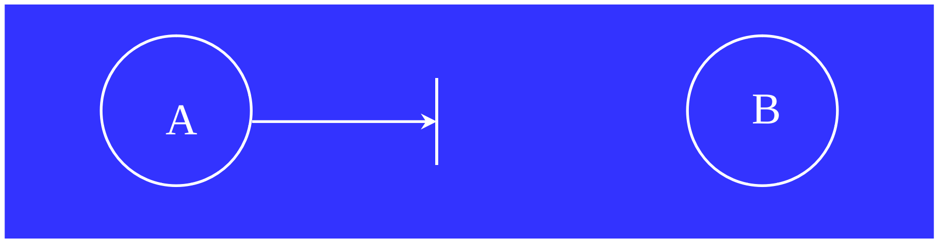
<u>Replay</u> involves the capture of messages during transit and its subsequent transmission to produce an unauthorized affect. It can only be detected, by authorization.



<u>Modification of messages</u> is a kind of replay, where the message is captured during transit, modified and again transmitted. Same as above.



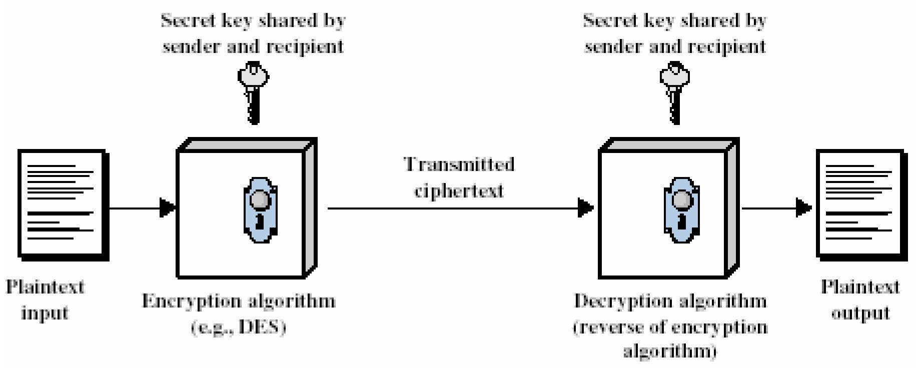
<u>Denial of Service</u>, attacker overload the server by sending false messages, thus server is congested and is unable to provide services. When condition like congestion are created deliberately by some entity or attacker, it my create denial of service for other users.



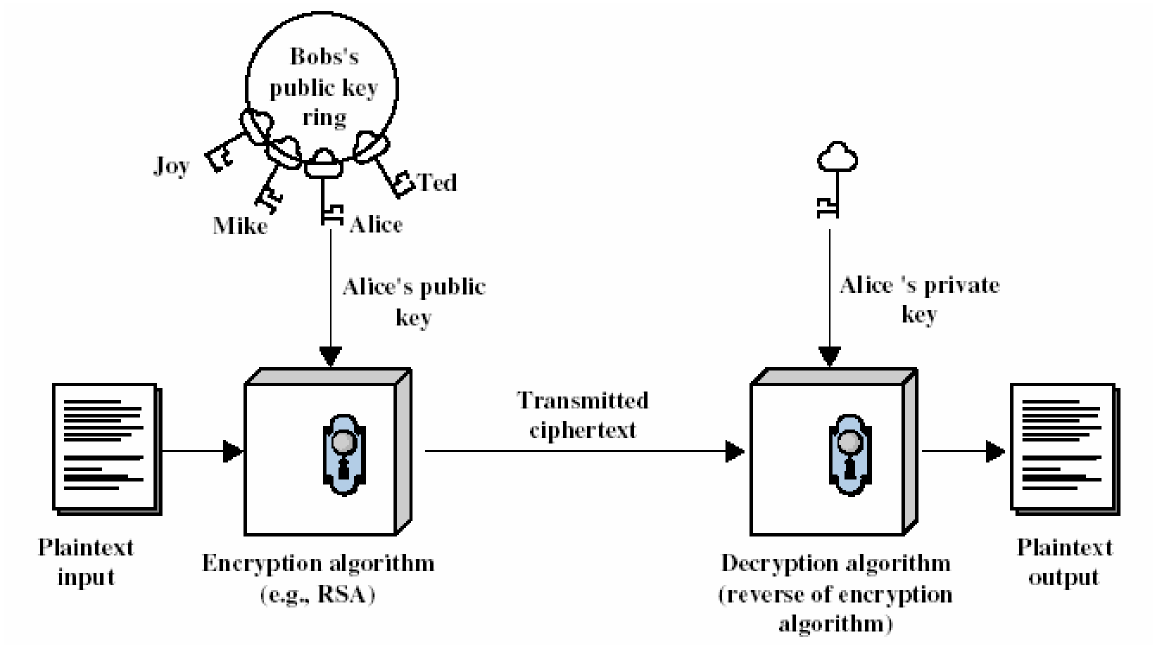
As discussed above, encryption is one of the ways to maintain the confidentiality of message. Encryption is classified into two types

1. Conventional encryption
2. Public Key encryption

Conventional encryption also referred to as the symmetric encryption or single key encryption i.e. both the sender and receiver has same pair of keys. Before transmission, sender encrypts the message using the common key, thus generating cipher text, which is transmitted through the communication channel, and received by the receiver. At the receiver end, cipher text is again decrypted to plaintext using the same key.


Secret key shared by sender and recipient
Secret key shared by sender and recipient
Transmitted ciphertext
Plaintext input
Encryption algorithm (e.g., DES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

In **Public Key encryption**, each user has multiple keys, out of these multiple keys, one is the private key and rests other are public keys of other users. This means that each user has his own private key, and public keys of all other users he want to have communication with.



For example, in the above figure, Bob's has two set of keys, one his own public key, and other private keys of Joy, Mike, Alice, Ted and all. Now Bob wants to communicate with the Alice, so he encrypts the message using Alice public key. The encrypted text is then transmitted to Alice, upon receipt by the Alice, it is then decrypted to plain text using Alice's own private key. This (private key, public key) form a pair; any text encrypted using public key can only be decrypted through corresponding private key only. Public key is made available to all the people, while the private key is only with the user who is actual owner of the key.

Conventional Encryption

A conventional encryption has five ingredients

1. Plain-text: This is the original message or data that is fed into the algorithm as input.
2. Encryption Algorithm: The encryption algorithm performs various substitution and transformation on message on the plaintext.
3. Secret Key: The secret key is also input to the encryption algorithm. The exact substitution and transformation performed by the algorithm depend on the key. For different keys, same plain-text, same algorithm, cipher text will be different.
4. Cipher- text: This is the scrambled message produced as output. It depends upon the plaintext and secret key.
5. Decryption algorithm: The decryption algorithm is exactly reverse of encryption algorithm and is used to obtain the plain text from the cipher text. It also requires the same key through which message has been encrypted.

There are two **requirements for secure use of conventional encryption**

1. We need a strong encryption algorithm. At a minimum we would like the algorithm to be such that an opponent who knows the algorithm and has access to one or more cipher-text would be unable to decipher the cipher-text or figure out the key.
2. Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure.

Attacking encrypted data, and doing analysis of the encrypted data, to generate the original plain-text is called as crypt-analysis. **Crypt-analysis** is a branch of cryptography, where researchers or hackers, decrypt the cipher-text without being aware of either the encryption algorithm or key.
Another approach to decryption is **brute-force attack**, where every possible key is applied on the cipher text to generate the plain text. It is like random generation of possible keys, and each one is tried until something intelligible or readable is obtained out of the cipher text.

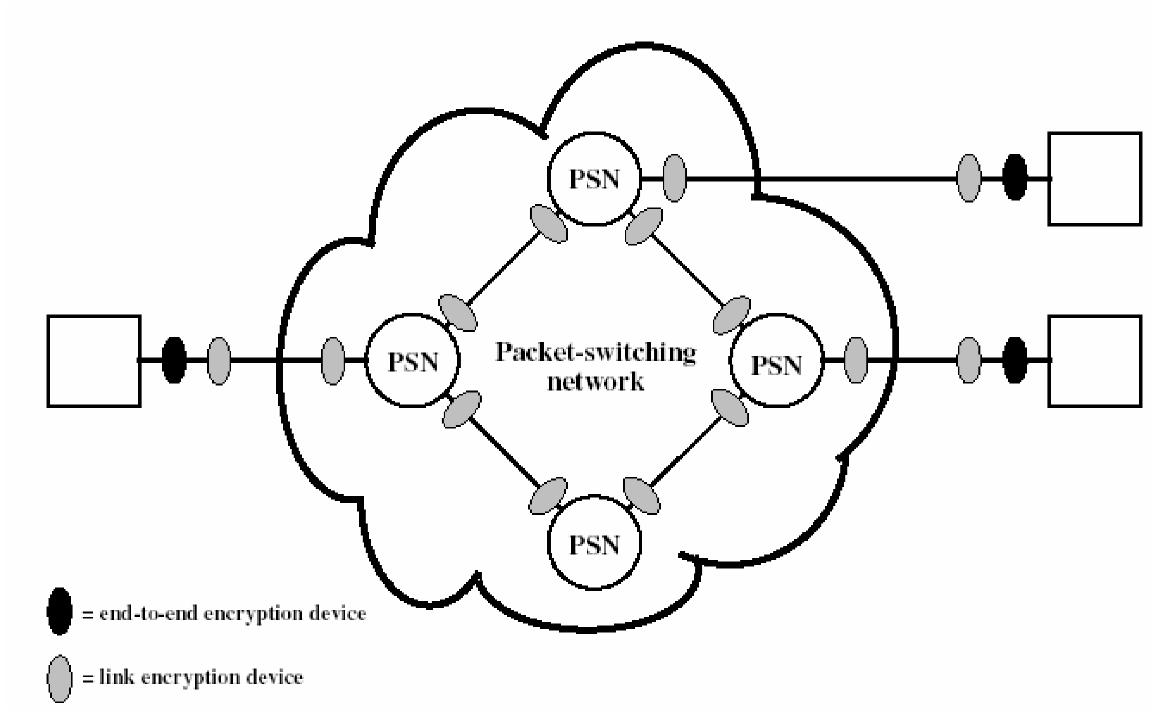
Location of Encryption Devices

In using encryption, one thing to decide is what to encrypt (which is obviously the data that is to be transmitted) and the other thing to decide is where to do the encryption. There are two places where encryption can be performed. One is **End-to-End encryption** device and the other one is **Link encryption** device

End-to-End encryption means that, encryption will be carried out at only two end points; one is at the source end and the other at the receiving end. The source host terminal encrypts the message and generates the cipher text, which is then transmitted over the link, after reaching the destination cipher-text is then decrypted. This approach seems fairly feasible but there is still a catch, since intermediate router has to take routing decision, for that they process the packet, and thus they also have to decrypt the packet, which they cannot do unless they have the key with them, so one of the solution is to provide them with key, so that they intermediate router can also decrypt the packet and can take routing decision. We cannot afford to supply key to every other router, then key will no longer be secure.

Another possible solution is to only encrypt the data using the key and leave the header unencrypted; by this we can achieve the confidentiality of message but not the authentication of message. This solution is more feasible and seems to be sensible and is being used in case of end-to-end encryption is done.

Another place of doing the encryption is **Link encryption**. With Link encryption, each end of the link has an encryption device as shown in figure below. At one end, packet is encrypted, and when the encrypted packet reaches the end of the link, it is decrypted again, where it is then given to the router for processing, after processing, router again gives the unencrypted packet to the encryption device for encryption, from where it is transmitted further and the process goes on.



Key Distribution in Conventional Encryption

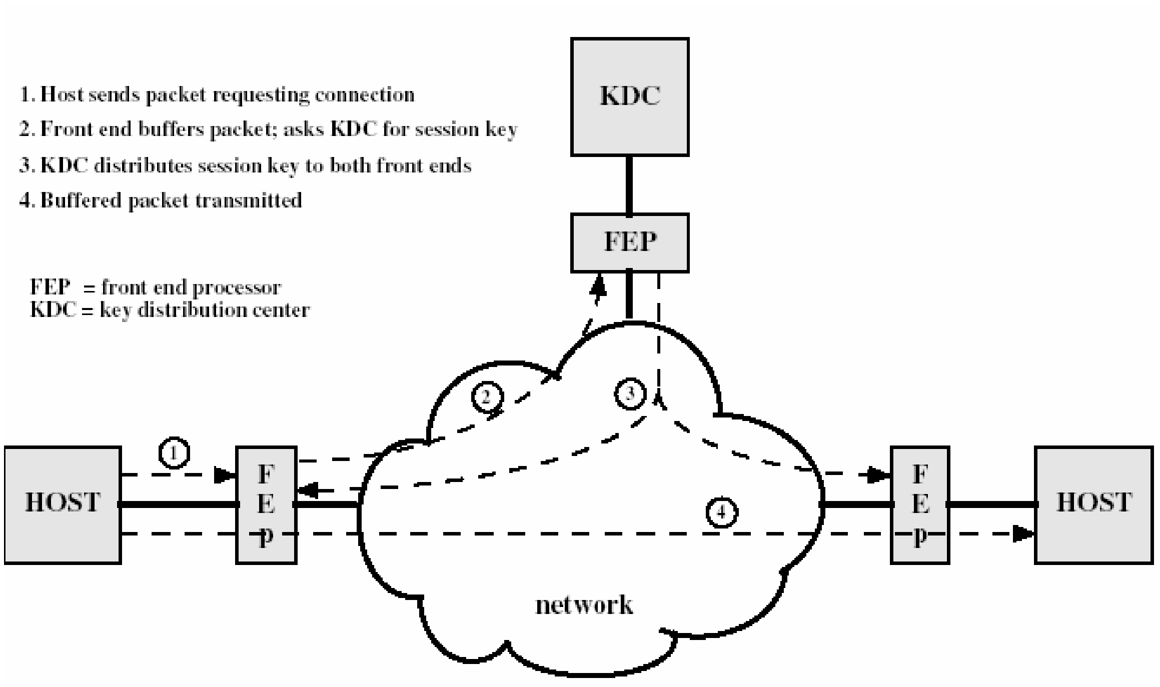
As we know that for conventional encryption, a key is required by both sender and receiver, sender encrypts the message using the key and then the cipher-text is decrypted at the receiver end using the same key. This means that there is another job of providing the sender and receiver with a key in a secured fashion, means that key should not be available to any other user. Therefore the strength of any cryptographic system relies with the key distribution technique, a term that refers to the means of delivering a key to two parties that wish to exchange data, without allowing others to see the key. There are various options to distribute keys to the user as explained below

1. A key could be selected by user A and physically delivered to user B.
2. A third party could select the key and physically deliver it to user A and user B.
3. If user A and user B have previously and recently used a key, one party could transmit a new key to the other, which is encrypted using the older key.
4. If user A and user B each share an encrypted connection to third party C, C could deliver the key on the encrypted links to A and B.

The first two approached are only feasible if the user A and user B are nearby; if they are far apart it is not possible to deliver the key physically.

Third approach has a catch, if some user C knows the older key; he can also know the new key being exchanged through encryption of older key.

Choice 4 presented above is the most viable one, i.e. user A and user B share an encrypted connection to third party C and third party C could deliver the key on the encrypted links to A and B, which is discussed below in detail and shown diagrammatically also.


1. Host sends packet requesting connection
2. Front end buffers packet; asks KDC for session key
3. KDC distributes session key to both front ends
4. Buffered packet transmitted
FEP = front end processor
KDC = key distribution center
KDC
FEP
HOST
FEP
network
FEP
HOST
1
2
3
4

For this scheme to work two kinds of keys are required

1. **Session Key**: When two end systems (hosts or terminals) wish to communicate, they establish a logical connection, for the duration of that logical connection all user data are encrypted with a one-time session key. At the conclusion of connection, session key is destroyed.
2. **Permanent Key**: A permanent key is a key used between two entities for the purpose of distributing session keys.

The configuration consist of the following elements

1. **Key distribution center**: The key distribution center determines which systems are allowed to communicate with each other. When permission is granted for two systems to establish a connection, the key distribution center provides a one-time session key for that connection.
2. **Front-end processor**: The front-end processor performs end-to-end encryption and obtains session keys on behalf of its host or terminal.

Steps involved in establishing a connection are as follows

1. When one host wishes to set up a connection to other host, it transmits a connection-request packet to front-end processor.
2. The front-end processor saves that packet and applies to the KDC for permission to establish the connection.
3. The communication between the FEP and KDC is encrypted using a master key (permanent key) shared only by the FEP and KDC. If KDC approves the connection request, it generates the session key and delivers it to both the FEP using a unique permanent key for each front-end.

4. The requesting FEP can now release the connection-request packet, which he had saved and now encrypted data transmission can take place.

Key management in Public Key Encryption

Another way to distribute the key to both the user is through public key encryption. If Bob wants to exchange message with Alice and other people, he generates a single pair of keys, one private and other one public. Public key is available for all the users, but private key is only for him which he doesn't share with anyone.

Alice can get Bob's public key, encrypt the secret key using Bob's public key, Bob after receiving the encrypted message, decrypts it using his own private key, and get the secret key, which he will use for conventional encryption.

Data Encryption Standard (DES)

DES is a ***block cipher***--meaning it operates on plaintext blocks of a given size (64-bits) and returns cipher text blocks of the same size. **DES operates on the 64-bit blocks using *key* sizes of 56- bits**.

Step 1: Takes data in 64-bit block, apply the initial permutation to the block of text **M,** and divide the resultant data block into two parts each of 32 bits, Left part is denoted by L_0 and Right part is denoted by R_0.

Step 2: Then key of 64-bit is taken, which is converted into 56 bit, because every 8th bit in the key is not used (i.e. bits numbered 8, 16, 24, 32, 40, 48, 56, and 64).

Step 3: This key is also divided into 2 parts, here each one of 28 bits. One part is called as C_0 and other one as D_0.

Step 4: With C_0 and D_0 defined, we now create sixteen blocks C_n and D_n, 1<=n<=16. Each pair of blocks C_n and D_n is formed from the previous pair C_{n-1} and D_{n-1}, respectively, for n = 1, 2,..., 16, using the following schedule of "left shifts" of the previous block.

Iteration Number	Number of Left Shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Step 5: We now form the keys K_n, for 1<=n<=16, by applying the following permutation table (PC-2) to each of the concatenated pairs C_nD_n. Each pair has 56 bits, but **PC-2** only uses 48 of these. So the final result is keys K_1, K_2, ..., K_n where each key is 48-bit.

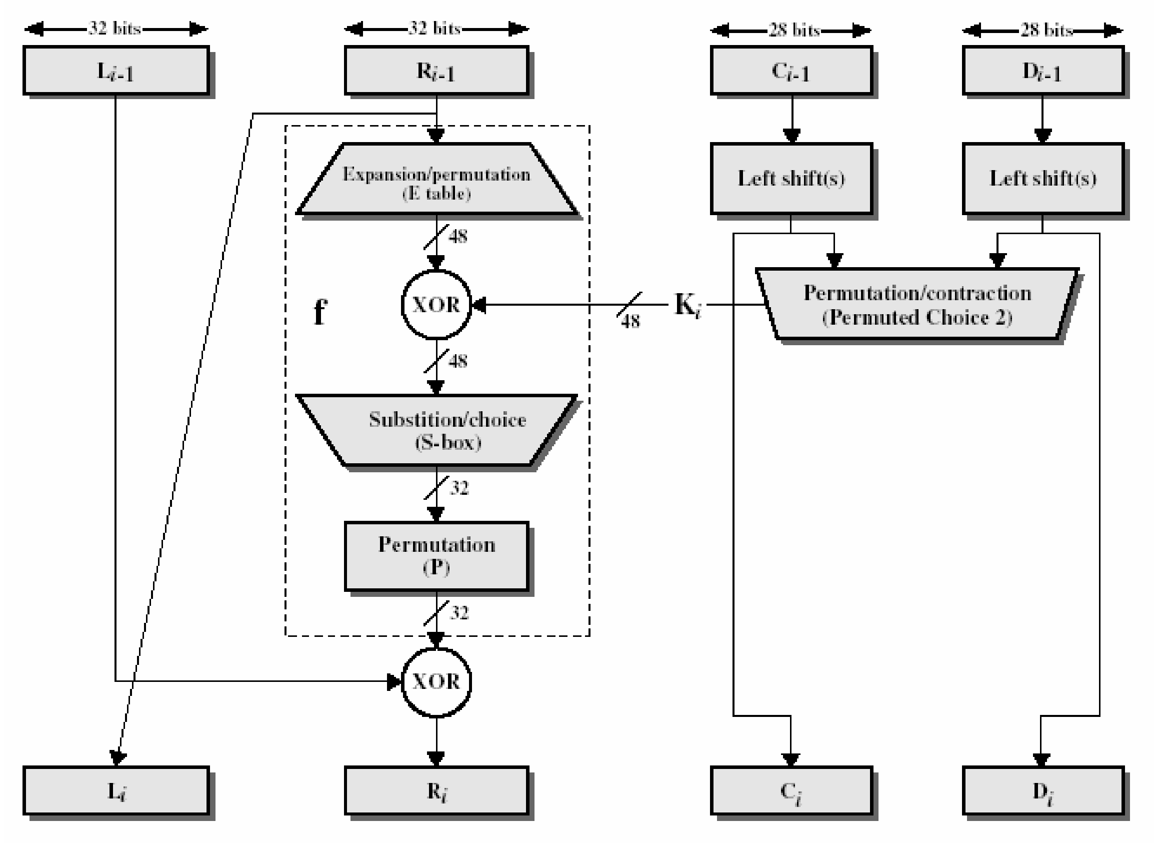
Step 6: We now proceed through 16 iterations, for 1<=n<=16, using a function f which operates on two blocks--a data block of 32 bits and a key K_n of 48 bits--to produce a block of 32 bits. **Let + denote XOR addition, (bit-by-bit addition modulo 2)**. Then for **n** going from 1 to 16 we calculate

$L_n = R_{n-1}$

$R_n = L_{n-1} + f(R_{n-1}, K_n)$

This results in a final block, for n = 16, of $L_{16}R_{16}$.
In each iteration we take the right 32 bits of the previous result and make them the left 32 bits of the current step. For the right 32 bits in the current step, we XOR the left 32 bits of the previous step with the calculation f.

Step 7: We then ***reverse*** the order of the two blocks into the 64-bit block $R_{16}L_{16}$ and apply a final permutation $\mathbf{IP}^{-1}$. The final result is the encrypted text.



Triple DES: TDES uses three keys and three execution of DES algorithm. The function follows an Encrypt-Decrypt-Encrypt sequence.

$$C = E_{K3} [D_{K2} [E_{K1}[P]]]$$

Where C= cipher-text
D= plain-text
$E_k[X]$ = encryption of X using key K.
$D_k[X]$ = decryption of X using key K.

Message Authentication and hash function

Encryption can only deal with confidentiality of data; it cannot guarantee the authenticity of data. It means whether the data is coming from the alleged source or not. If it is coming from specified source, it may be modified during transit. There are different message authentication techniques being used to ensure the authenticity of messages. One technique is using hash functions, second technique is using hash function with key (MAC), another technique is SHA. Another issue is message timeliness i.e. message may have been artificially delayed.

It is possible to cover one aspect of authenticity of data, i.e. whether data is from the alleged source, because both sender and receiver have a common shared key. If data is encrypted by alleged source using shared key, then only the recipient can decrypt using shared key otherwise not.
We can also include Error Detection code in the message, in case some intruding user modifies the data during transit, which will be treated as error at the receiver end.
We can also have sequence no with every message, thus ensuring the timeliness of one message relative to another one.
But through these techniques we can only detect but cannot prevent. We will discuss authentication algorithms without encryption which won't let any malfunctioning to happen with the data at the first place.

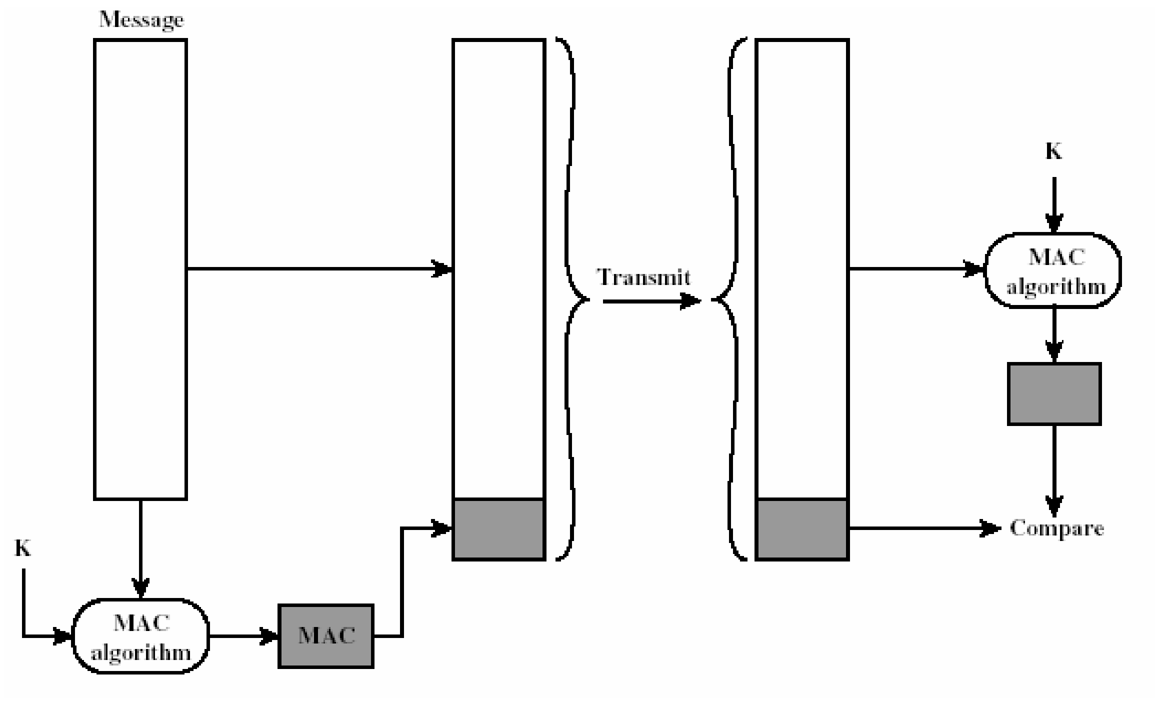
Secure Hash Functions

A hash function H is just like any other function with a few extra ordinary characteristics.
1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical and feasible.
4. For any given code h, it is computationally infeasible to find x such that H(x)=h, i.e. hash function should be such that, one should not able to locate value of domain from the value of range or in other words, reverse direction should be difficult.
5. For any given block of x, it is computationally infeasible to find $y \neq x$ with H(y)=H(x).

Message Authentication Code (MAC)

This authentication technique makes use of hash function and secret key also. When user A has a message to send to user B, it calculates the Message Authentication Code as a function of the message and the key: i.e. $MAC_M = F (K_{AB},M)$.

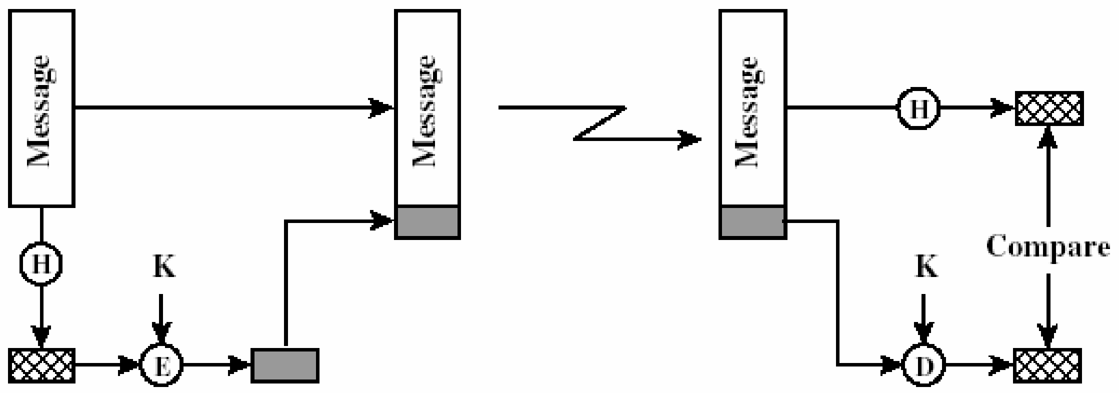
This MAC is appended at the end of message and is transmitted to the receiver. Receiver performs the same calculation on the message only (since he also has the secret key), generates the message authentication code and compares with the received MAC. If both of them match, then message is from the alleged sender otherwise not.



A number of algorithms can be used for generating the code, even DES can be used.

There are three ways in which message authentication can be done.

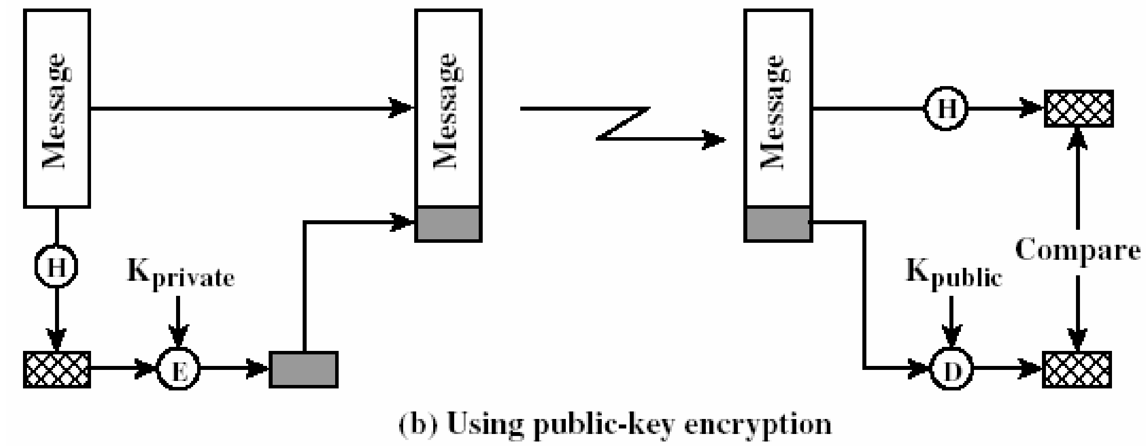
1. The message is first subjected to hashing function, which generated the MAC or message digest which is then encrypted using conventional encryption (both the sender and receiver share the secret key). The procedure is depicted below.



(a) Using conventional encryption

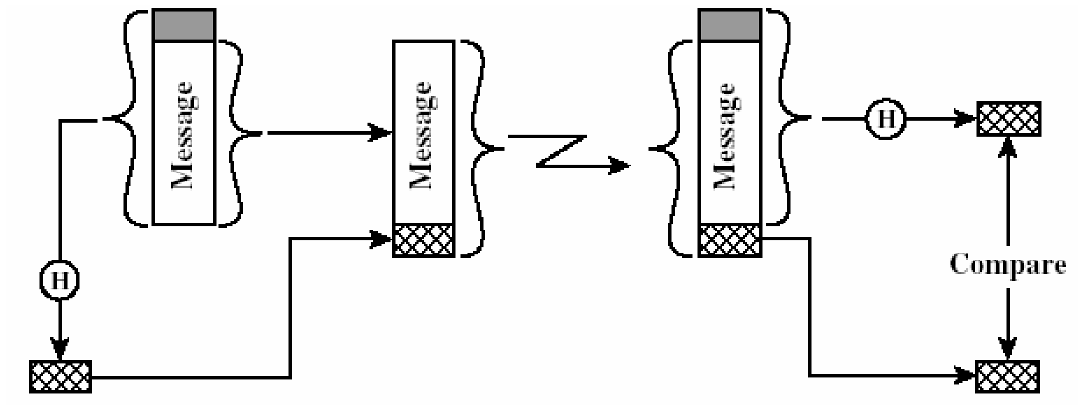
2. Message is first subjected to hashing function, which generates the MAC or message digest, which is then encrypted using public key encryption. The procedure

is depicted below diagrammatically. This approach provides two benefits, message authentication and digital signature.



(b) Using public-key encryption

3. This technique doesn't involve use of encryption algorithm. Both the users share a common secret value (S_{AB}). When user A has a message to send to B, it calculates the Hash function over the concatenation of the secret value and the message i.e. $MD_M = H(S_{AB}||M)$. It then sends $M||MD_M$. Because B also possesses S_{AB}, it can compute $H(S_{AB}||M)$ and verify MD_M. This technique uses a shared secret value. This technique is adopted in SNMPv3 and IP security (discussed later).



(c) Using secret value

The Secure Hash Function (SHA-1)

This algorithm takes as input a message with a maximum length of less than 2^{64} bits and produces as output a 160-bit message (or message digest). The input is processed in blocks of 512-bit. The purpose of message padding is to make the total length of a padded message a multiple of 512 bits. As a summary, a "1" followed by ***m*** "0"s followed by a 64-bit integer are appended to the end of the message to produce a padded message of length 512 * n. The 64-bit integer is the length of the original message. The padded message is then processed by the SHA-1 as n 512-bit blocks or block of 16 word each.

The SHA-1 is called secure because it is computationally infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest. Any change to a message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify.

The processing consists of the following steps:
For more detail discussion refer to
http://www.itl.nist.gov/fipspubs/fip180-1.htm

1. As discussed above the whole message is divided into multiples of 512 bit, i.e. there will be n blocks and each block consisting of 512 bits. This objective is achieved by padding. Padding is done at the end of the message, by appending 1 bit, followed by 0's, which is followed by length of the data. 64-bit block is appended at the end which contains the length of the data before padding.
   For example:
   Suppose the original message is the bit string
   01100001 01100010 01100011 01100100 01100101
   Length of the message is 40 bit, which is not a multiple of 512-bit. So we will append 1 to it. After appending 1 to the message, length will be 41-bit.
   01100001 01100010 01100011 01100100 01100101 **1**

   Now we can append only 407 0's to it because (512-41-64=407). This gives (in hex)
   61626364 65800000 00000000 00000000
   00000000 00000000 00000000 00000000
   00000000 00000000 00000000 00000000
   00000000 00000000 00000000 00000028.

The padded message will contain 16 * n words for some n > 0. The padded message is regarded as a sequence of n blocks M_0 , M_1, ... , M_{n-1}, where each M_i contains 16 words.

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

   A = 67452301

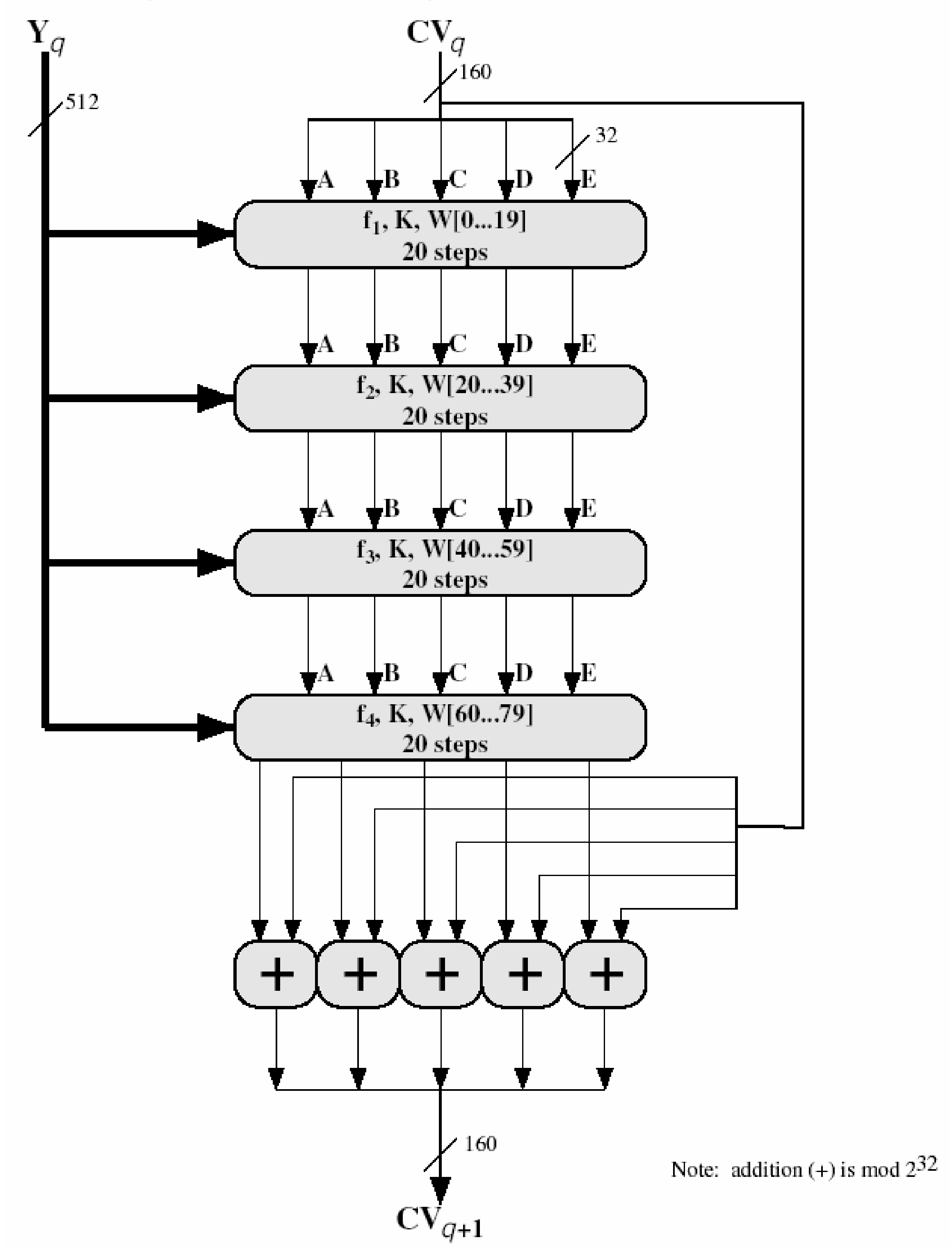
   B = EFCDAB89

   C = 98BADCFE

   D = 10325476

   E = C3D2E1F0

3. Processing message in 512-bit blocks: This module consists of 4 rounds of processing and each round consisting of 20 steps each that makes a total of 80



4. The four rounds have a similar structure but each uses different logical function. Actually there are five logical functions, which are explained in more detail at the URL specified above, (not part of the course). Each round takes as the input the

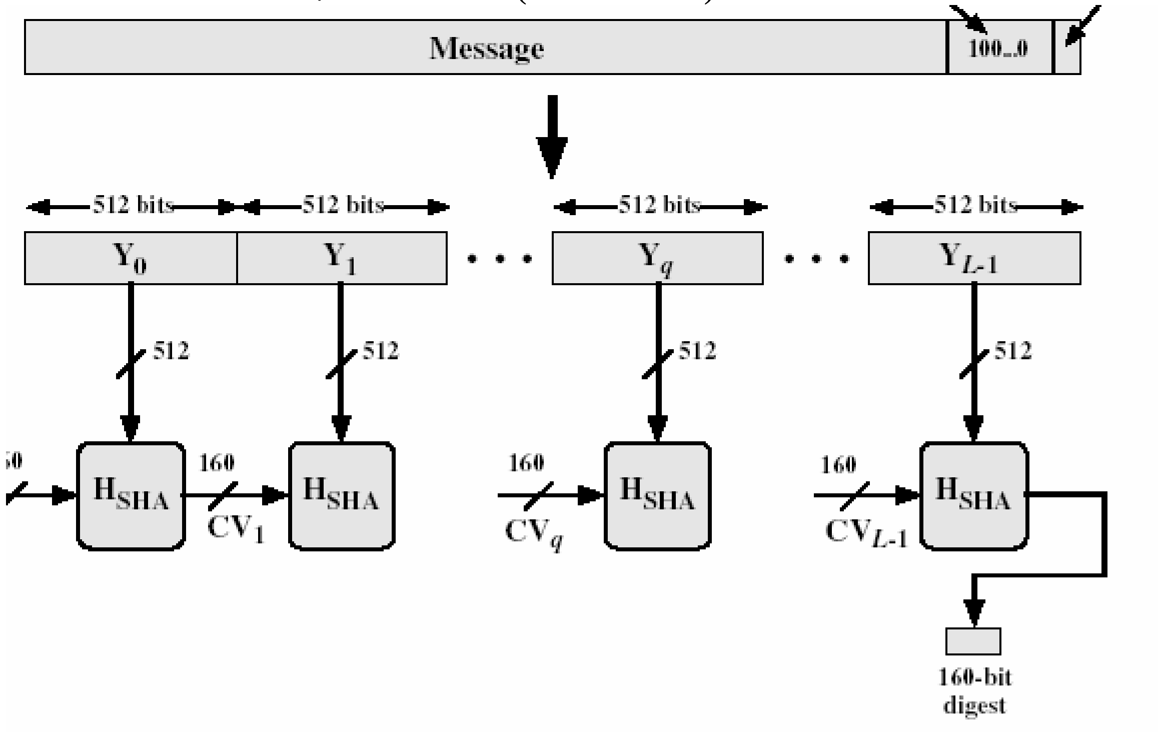
current 512 bit block being processed (Y_q) Each round also makes use of additive constant K_t where $0<t<79$.

K_t = 5A827999 (0 <= t <= 19)

K_t = 6ED9EBA1 (20 <= t <= 39)

K_t = 8F1BBCDC (40 <= t <= 59)

K_t = CA62C1D6 (60 <= t <= 79).



The output of the fourth round is added as the input to the first round is added to the input to the first round to produce CV_{q+1}.

RSA (Public-Key Cryptography)

First we will discuss about public key cryptography, and then RSA algorithm which is based on public key cryptography.
A public-key encryption scheme has six ingredients
Plain-text: This is the readable message that is the input to the algorithm

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

Public and Private Key: This is a pair of keys that have been selected so that if one is used for encryption the other is used for decryption.

Cipher-text: This is the scrambled message produced as output, it depends upon the plaintext and the key.

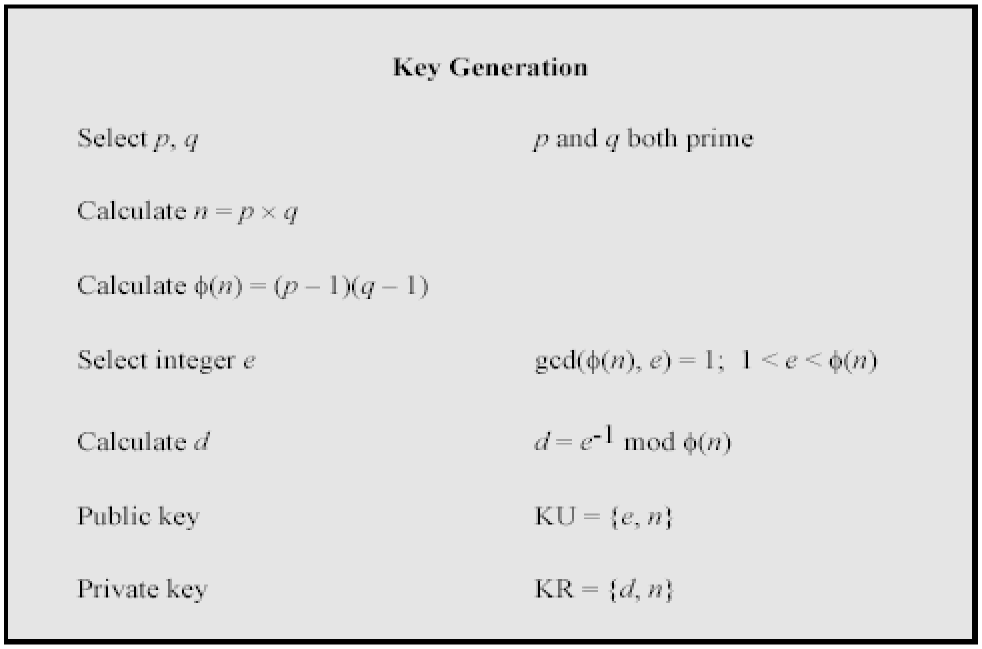
Decryption Algorithm: This algorithm accepts the cipher-text and the matching key and produces the original plain-text.

The essential steps for encryption are as follows
1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register. This is public key and is available to all the users.
3. If Bob wants to send a message to Alice, he will first obtain the public key of Alice, encrypt the message using the public key of Alice, and send it to Alice.
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message, because private key of Alice is not available to anyone.

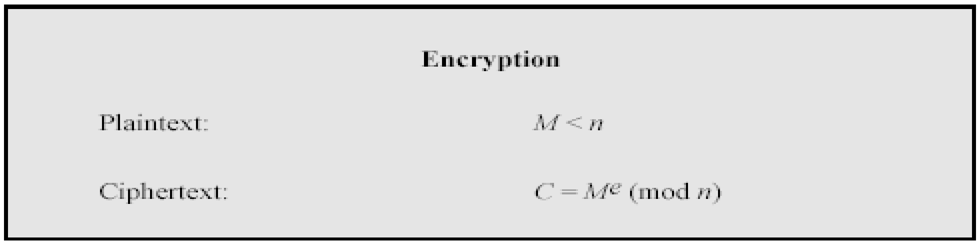
RSA algorithm revolves around generating such pair of keys, that message encrypts by one key (which is generally made available for the public) and can only be decrypted by its corresponding key (which is kept as private).

RSA Public-Key Encryption Algorithm

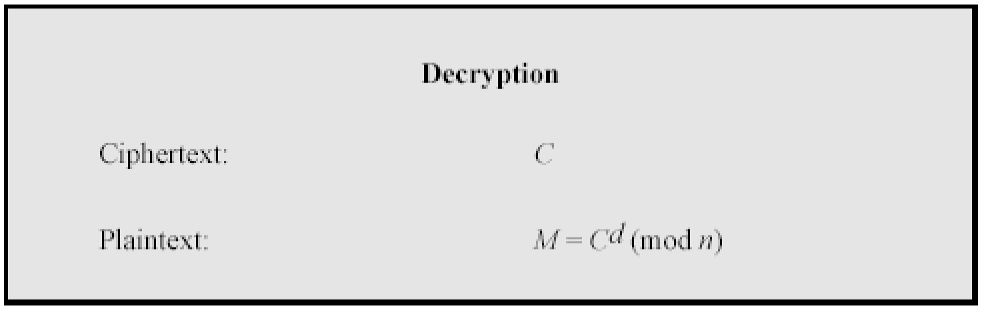
It was developed at MIT in 1977 by Ron Rivest, Adi Shamir, and Len Adelman. Below is the process to generate the key.



Key Generation	
Select p, q	p and q both prime
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$d = e^{-1} \bmod \phi(n)$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$



Encryption	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod{n}$

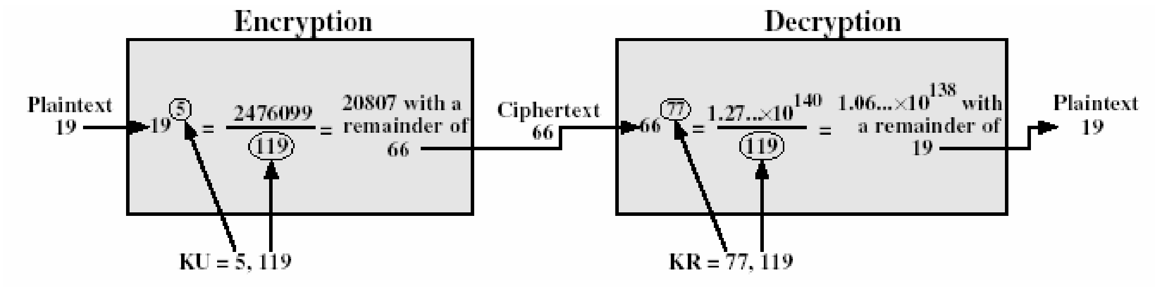


Decryption	
Ciphertext:	C
Plaintext:	$M = C^d \pmod{n}$

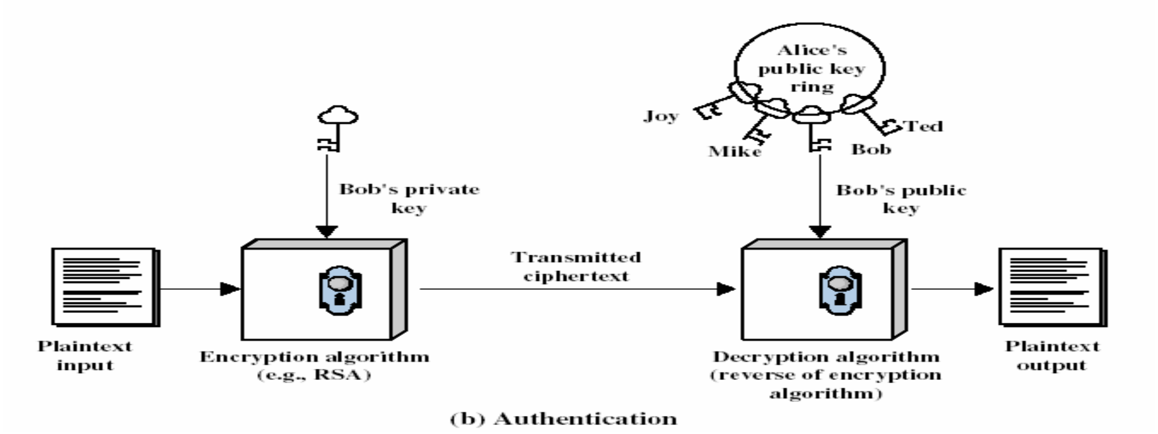
Here is an example of RSA Algorithm

1. Select two prime numbers, p=7 and q=17
2. Calculate n = p. q = 7.17 =119
3. Calculate ϕ(n) = (p - 1). (q - 1) = 96
4. Select e, such that e is relatively prime to ϕ(n) = 96, which comes out to be equal to 5.
5. Determine d such that de = 1 mod 96. The correct value of d turns out to be 77. i.e. d=77 because 77 * 5 = 385 = 4 * 96 + 1

So the public key is {5, 119} and private key is {77, 119}.


Encryption
Decryption
Plaintext 19
$19^{5} = \frac{2476099}{119} =$ 20807 with a remainder of 66
Ciphertext 66
$66^{77} = \frac{1.27... \times 10^{140}}{119} =$ 1.06...$\times 10^{138}$ with a remainder of 19
Plaintext 19
KU = 5, 119
KR = 77, 119

Public key encryption algorithms do the job of encryption and authentication also, as shown in figure below.



Authentication means, whether a genuine sender is sending the message or not, there is a term called as IP spoofing, which means, a sender will send a message which will not contain his IP address but some one else IP address, IP spoofing is done to send malicious messages and viruses through email to other users by hiding their identity. To authenticate whether the user is genuine, we can user Public Key Encryption Algorithm.
For example, If Alice makes it a point that he will only receive data from genuine senders or authentic sender, this can be achieved if Alice ask every sender to only send the messages after encrypting it with their private keys, and Alice will decrypt the message by the corresponding public key of that user. If Alice is able to decrypt the cipher-text perfectly with the corresponding Public Key of the respective user, which means message is from the genuine sender.