**COMPUTER SECURITY DEEPA M MODULE II**

**STREAM CIPHERS AND RC4**

**Stream cipher structure**

A typical stream cipher encrypts plaintext one byte at a time, although a stream cipher may be designed to operate on one bit at a time or on units larger than a byte at a time.



Figure is a representative diagram of stream cipher structure. In this structure, a key is input to a pseudorandom bit generator that produces a stream of 8-bit numbers that are apparently random. The output of the generator, called a **keystream**, is combined one byte at a time with the plaintext stream using the bitwise exclusive-OR (XOR) operation.

The following important design considerations for a stream cipher.

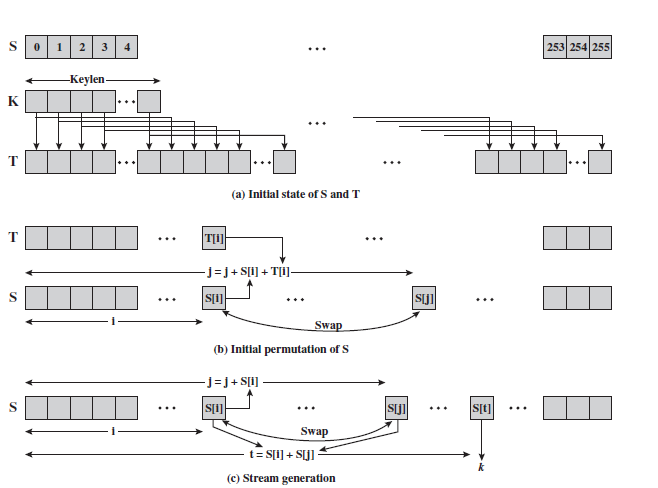
* The encryption sequence should have a large period.
* The keystream should approximate the properties of a true random number stream as close as possible.
* The output of the pseudorandom number generator is conditioned on the value of the input key

A potential advantage of a stream cipher is that stream ciphers are typically faster and use far less code than do block ciphers. One advantage of a block cipher is that we can reuse keys.

**RC4**

RC4 is a stream cipher designed in 1987 by Ron Rivest for RSA Security. It is a variable key size stream cipher with byte-oriented operations.The algorithm is based on the use of a random permutation.

RC4 is used in the Secure Sockets Layer/Transport Layer Security (SSL/TLS) standards that have been defined for communication between Web browsers and servers. It is also used in the Wired Equivalent Privacy (WEP) protocol and the newer WiFi Protected Access (WPA) protocol that are part of the IEEE 802.11 wireless LAN standard. RC4 was kept as a trade secret by RSA Security.



A variable- length key of from 1 to 256 bytes (8 to 2048 bits) is used to initialize a 256-byte state vector S, with elements S[0],S[1],**.....** ,S[255]. At all times, contains a permutation of all 8-bit numbers from 0 through 255. For encryption and decryption, a byte (see Figure 7.5) is generated from S by selecting one of the 255 entries in a systematic fashion. As each value of is generated, the entries in S are once again permuted.

**Initialization**

for i = 0 to 255 do

S[i] = i

T[i] = K[i mod keylen])

**Initial permutation of S**

j = 0

for i = 0 to 255 do

j = (j + S[i] + T[i]) (mod 256)

swap (S[i], S[j])

**Stream generation**

i = j = 0

for each message byte Mi

i = (i + 1) (mod 256)

j = (j + S[i]) (mod 256)

swap(S[i], S[j])

t = (S[i] + S[j]) (mod 256)

k = S[t]

Encryption,

Ci = Mi XOR k

Decryption,

Mi = Ci XOR k

**Initialization of S**

To begin, the entries of are set equal to the values from 0 through 255 in ascending order; that is S[0] = 0, S[1] = 1,**....**, S[255] = 255. A temporary vector, T, is also created. If the length of the key K is 256 bytes, then is transferred to T. Otherwise, for a key of length *keylen* bytes, the first *keylen* elements of T are copied from K, and then K is repeated as many times as necessary to fill out T.

**Initial permutation of S**

Next we use T to produce the initial permutation of S. This involves starting with S[0] and going through to S[255] and for each S[i], swapping with S[i] another byte in according to T[i].

**Stream Generation**

Once the S vector is initialized, the input key is no longer used. Stream generation involves cycling through all the elements of S[i], , and for each S[i], , swapping with S[i],another byte in S according to a scheme dictated by the current configuration of S.After S[255] is reached, the process continues, starting over again at S[0].

To encrypt, XOR the value with the next byte of plaintext.To decrypt,XOR the value with the next byte of ciphertext.

**CONFIDENTIALITY USING SYMMETRIC ENCRYPTION**

**Placement of Encryption**

Confidentiality means the protection of data from unauthorized access. If encryption is to be used to counter attacks on confidentiality,we need to decide what to encrypt and where the encryption function should be located. There are two fundamental encryption placement alternatives: link encryption and end-to-end encryption.

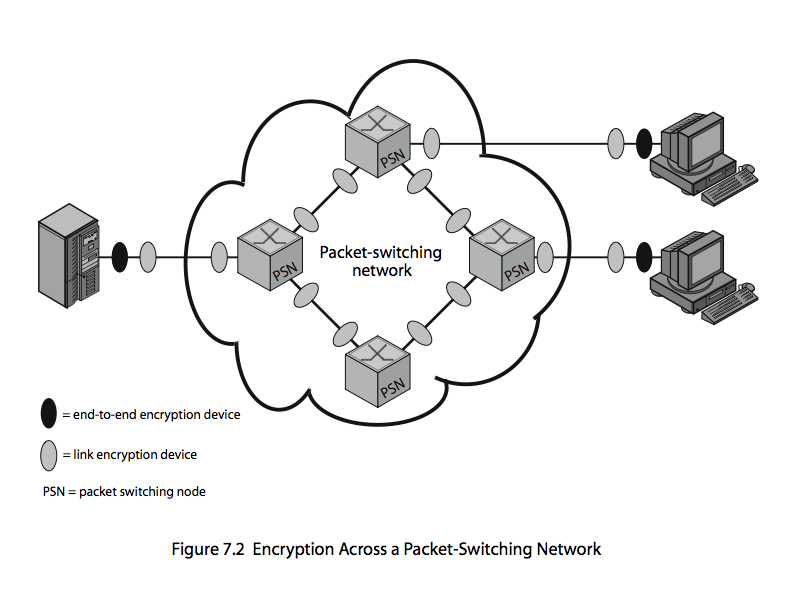
**Link Vs End –to-End Encryption**

The most powerful and most common approach to secure the points of vulnerability of attacks is encryption .There are two major methods for encryption-Link encryption and end to end encryption.

With link encryption, each vulnerable communications link is equipped on both ends with an encryption device.Thus all the traffic over all communication link is secure.One of the disadvantage is that message must be decrypted each time it enters a switch because switch must read the address in the packet header inorder to route the frame . All the potential links in a path from source to destination must use link encryption. Each pair of nodes that share a link should share a unique key, with a different key used on each link. Thus, many keys must be provided.

With end-to-end encryption, the encryption process is carried out at the two end systems. This needs devices at each end with shared keys. The user data is secure, but the traffic pattern is not because packet headers are transmitted in the clear.End-to-End encryption provides a degree of authentication.

To achieve more security both link and end-to-end encryption are needed.

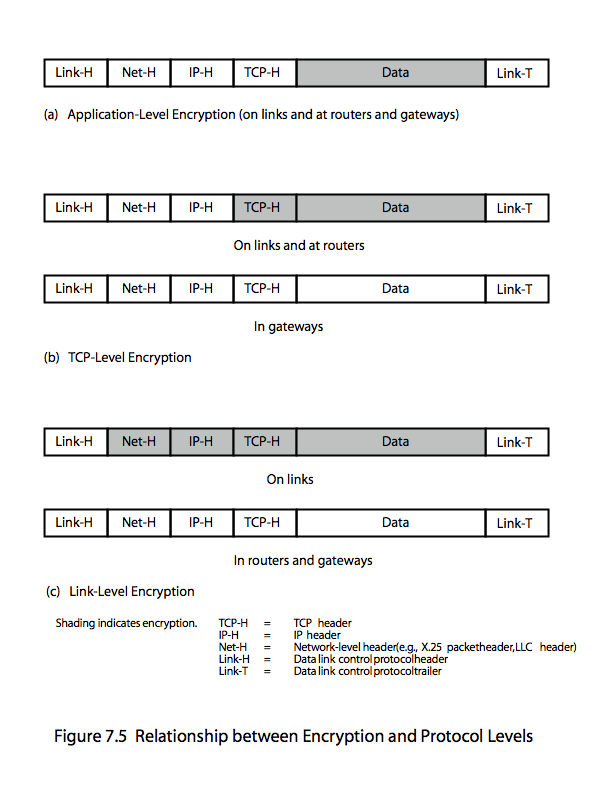


**Logical placement of Encryption function**

We can place encryption function at various layers in OSI Reference Model.

* link encryption occurs at layers 1 or 2
* end-to-end can occur at layers 3, 4, 6, 7
* as move higher less information is encrypted but it is more secure though more complex with more entities and keys

Figure illustrates the relationship between encryption and protocol level, using the TCP/IP architecture as an example, showing how much information in a packet is protected.With application level encryption only the user data portion of a TCP segment is encrypted.If encryption is performed at TCP level,the user data and TCP header are encrypted.Because the gateways are treated as destination, the encrypted portion of the data unit are decrypted at the gateways.Finally for link level encryption,the entire data unit except for the link header and trailor is encrypted on each link.But the entire data unit is in the clear at each router and gateway.

****

**TRAFFIC CONFIDENTIALITY**

The following types of information that can be derived from a traffic analysis attack:

* identities of partners
* how frequent the partners are communicating,
* message pattern,message length, or quantity of messages
* the events that correlate with special conversation between particular partners

Traffic analysis can also be used to create a covert channel.The channel is used to transfer information in a way that violates the security policy.With the use of link encryption, network-layer headers are encrypted, reducing the opportunity for traffic analysis,but the attacker can still access the amount of traffic. An effective countermeasure to this attack is traffic padding. If only end-to-end encryption is employed,traffic padding and null messages can be used .

**KEY DISTRIBUTION**

The strength of any cryptographic system rests with the *key distribution technique*, a term that refers to the means of delivering a key to two parties who wish to exchange data without allowing others to see the key. 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. In a distributed system, any given host or terminal may need to engage in exchanges with many other hosts and terminals over time.Thus, each device needs a number of keys supplied dynamically.

Option 3 is a possibility for either link encryption or endto- end encryption, but if an attacker ever succeeds in gaining access to one key, then all subsequent keys will be revealed.

For end-to-end encryption, 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.

The use of a key distribution center is based on the use of a hierarchy of keys. At a minimum, two levels of keys are used . Communication between end systems is encrypted using a temporary key, often referred to as a **session key**.Typically, the session key is used for the duration of a logical connection, such as a frame relay connection or transport connection, and then discarded. Each session key is obtained from the key distribution center over the same networking facilities used for end-user communication. Accordingly, session keys are transmitted in encrypted form, using a **master key** that is shared by the key distribution center and an end system or user

If there are N entities that wish to communicate in pairs, then, as was mentioned, as many as[N(N-1)]/2 session keys are needed at any one time. However, only N master keys are required

**A Key Distribution Scenario**

The scenario assumes that each user shares a unique master key with the key distribution center (KDC).


Ch07. KDC.pdf                                                  00156198  Mnementh                      BEAE7A2F:

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, , for this transaction, which we refer to as a **nonce.**
2. The KDC responds with a message encrypted using Ka.The message includes two items intended for A:

• The one-time session key,Ks , to be used for the session

• The original request message, including the nonce, to enable A to match this response with the appropriate request.

In addition, the message includes two items intended for B:

• The one-time session key, , to be used for the session

• An identifier of A,IDA

These last two items are encrypted with Kb.

1. A stores the session key for use in the upcoming session and forwards to B the information that originated at the KDC for B.
2. Using the newly minted session key for encryption, B sends a nonce,N2 , to A.
3. Also, using Ks ,A responds with f(N2) , where f is a function that performs some transformation on N2 (e.g., adding one).

**Key Distribution Issues**

Hierarchical Key Control

It is not necessary to limit the key distribution function to a single KDC. Indeed, for very large networks, it may not be practical to do so.As an alternative, a hierarchy of KDCs can be established. For example, there can be local KDCs, each responsible for a small domain of the overall internetwork, such as a single LAN or a single building. 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.

Session Key Lifetime

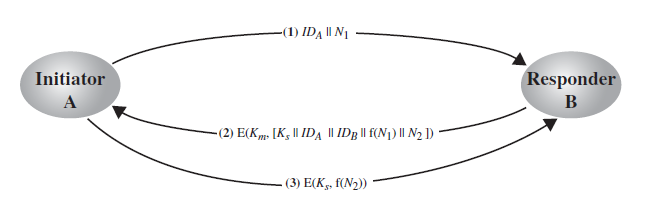
The more frequently session keys are exchanged, the more secure they are, because the opponent has less ciphertext to work with for any given session key. On the other hand, the distribution of session keys delays the start of any exchange and places a burden on network capacity.A security manager must try to balance these competing considerations in determining the lifetime of a particular session key.

A Transparent Key Control Scheme

The scheme is useful for providing end-toend 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 noteworthy element of this approach is a session security module (SSM), which may consist of functionality at one protocol layer, that performs end-to-end encryption and obtains session keys on behalf of its host or terminal.

Decentralized Key Control

The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distribution is fully decentralized. 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



Controlling Key Usage

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

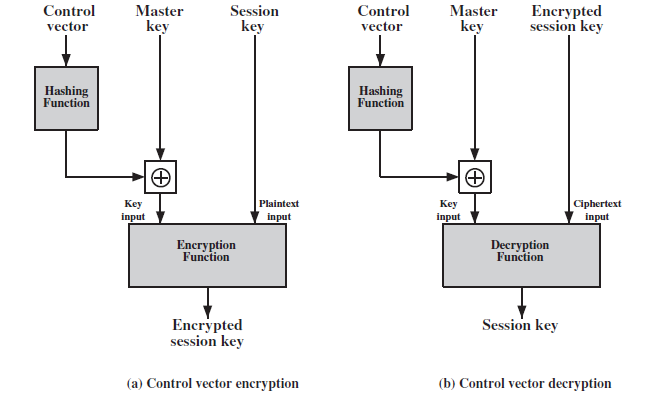
It may be desirable to institute controls in systems that limit the ways in which keys are used, based on characteristics associated with those keys. One simple plan is to associate a tag with each key .The proposed technique is for use with DES and makes use of the extra 8 bits in each 64-bit DES key.That is, the eight non-key bits ordinarily reserved for parity checking form the key tag.The bits have the following interpretation:

• One bit indicates whether the key is a session key or a master key.

• One bit indicates whether the key can be used for encryption.

• One bit indicates whether the key can be used for decryption.

• The remaining bits are spares for future use.



**Random Number generation**

A number of network security algorithms and protocols based on cryptography make use of random binary numbers.

* + nonces in authentication protocols to prevent replay
  + session keys
  + public key generation
  + keystream for a one-time pad
* in all cases its critical that these values be
  + statistically random, uniform distribution, independent
  + unpredictability of future values from previous values

**Randomness**:The following two criteria are used to validate that a sequence of numbers is random:

• **Uniform distribution:** The distribution of bits in the sequence should be uniform; that is, the frequency of occurrence of ones and zeros should be approximately equal.

• **Independence:** No one subsequence in the sequence can be inferred from the others.

**Pseudorandom Number Generators (PRNGs)**

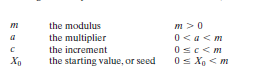
Cryptographic applications typically make use of algorithmic techniques for random number generation.These algorithms are deterministic and therefore produce sequences of numbers that are not statistically random. However, if the algorithm is good, the resulting sequences will pass many reasonable tests of randomness. Such numbers are referred to as **pseudorandom numbers**.

A **True random Number Generators (**TRNG )takes as input a source that is effectively random; the source is often referred to as an **entropy source.**The TRNG may simply involve conversion of an analog source to a binary output. In contrast, a PRNG takes as input a fixed value, called the **seed, and produces** a sequence of output bits using a deterministic algorithm

1. Linear Congruential Generator

* A widely used technique for pseudorandom number generation is an algorithm first proposed by Lehmer
* common iterative technique using:

***Xn*+1 = (*aX*n + *c*) mod *m***



* given suitable values of parameters can produce a long random-like sequence
* suitable criteria to have are:
  + function generates a full-period
  + generated sequence should appear random
  + efficient implementation with 32-bit arithmetic

PRNG Using Block Cipher Modes of Operation

Two approaches that use a block cipher to build a PNRG have gained widespread acceptance: the CTR mode and the OFB mode.

* Counter Mode

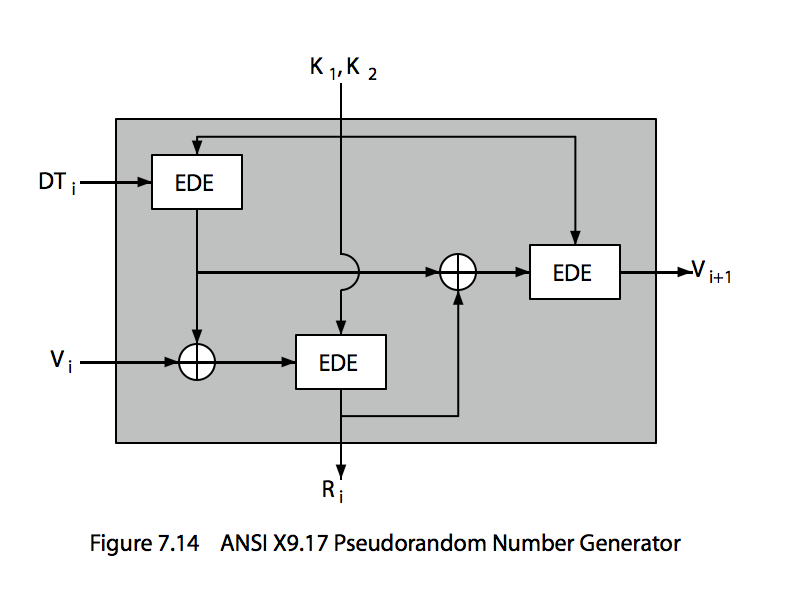
***Xi* = E*Km*[i]**

* Output Feedback Mode

***Xi* = E*Km*[*Xi-1*]**

1. ANSI X9.17 PRNG

* One of the strongest PRNGs is specified in ANSI X9.17.



* makes use of triple DES for encryption.
* The ingredients are as follows:
* Input: Two pseudorandom inputs drive the generator. One is a 64-bit representation of the current date and time, which is updated on each number generation.The other is a 64-bit seed value Vi
* Keys: The generator makes use of three triple DES encryption modules. All three make use of the same pair of 56-bit keys
* Output: The output consists of a 64-bit pseudorandom number and a 64-bit seed value.
* Ri = EDE([K1, K2], [Vi XOR EDE([K1, K2], DTi)])
* Vi+1= EDE([K1, K2], [Ri XOR EDE([K1, K2], DTi)])

**True random number generators**

A true random number generator (TRNG) uses a nondeterministic source to produce randomness. Most operate by measuring unpredictable natural processes,such as pulse detectors of ionizing radiation events, gas discharge tubes, and leaky capacitors

**Skew**

* A TRNG may produce an output that is biased in some way, such as having more ones than zeros or vice versa.
* Various methods of modifying a bit stream to reduce or eliminate the bias have been developed. These are referred to as deskewing algorithms.
* One approach to deskew is to pass the bit stream through a hash function, such as MD5 or SHA-1

**PUBLIC-KEY CRYPTOGRAPHY**

The development of public-key cryptography is the greatest and perhaps the only true revolution in the entire history of cryptography.

Asymmetric encryption is a form of cryptosystem in which encryptionand decryption are performed using the different keys—one a public key and one a private key. It is also known as public-key encryption or two key encryption. Asymmetric encryption transforms plaintext into ciphertext using a one of two keys and an encryption algorithm. Using the paired key and a decryption algorithm, the plaintext is recovered from the ciphertext. Asymmetric encryption can be used for confidentiality, authentication,or both.

**Principles of public key cryptography**

* public key cryptography was developed to address two key issues:
  + **key distribution** – how to have secure communications in general without having to trust a KDC with your key
  + **digital signatures** – how to verify a message comes intact from the claimed sender

**Public key cryptosystems**

Public-key/ two-key/ asymmetric cryptography involves the use of two keys which are related to each other:

* a public-key, which may be known by anybody, and can be used to encrypt messages, and verify signatures
* a private-key, known only to the recipient, used to decrypt messages, and sign (create) signatures

The algorithms have the following important characteristics:

* It is computationally infeasible to determine the decryption key given only the knowledge of the cryptographic algorithm and the encryption key.

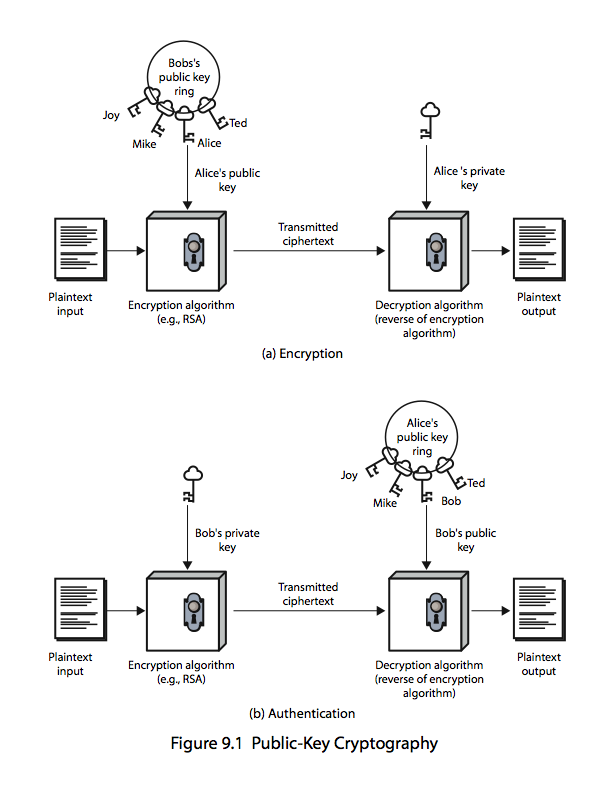
In addition, some algorithms, such as RSA, also exhibit the following characteristic:

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

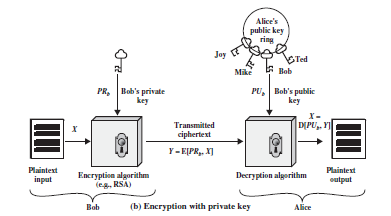
A public-key encryption scheme has six ingredients: **plaintext, encryption algorithm, public & private keys, ciphertext & decryption algorithm.**

The essential steps are the following:

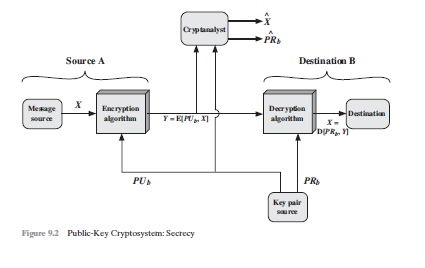
* Each user generates a pair of keys to be used for encryption and decryption of messages.
* Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private.
* If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice’s public key.
* When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice’s private key.



**(a) Encryption with public key**



Let the plaintext be X=[X1, X2, X3, …,Xm] where m is the number of letters in some finite alphabets. Suppose A wishes to send a message to B. B generates a pair of keys: a public key PUb and a private key PRb. PRb is known only to B, whereas PUb is publicly available and therefore accessible by A.



With the message X and encryption key PUb as input, A forms the cipher text Y=[Y1, Y2, Y3, … Yn].

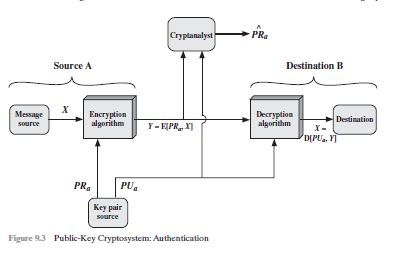
i.e., Y=E (PUb ,X)

The receiver can decrypt it using the private key PRb

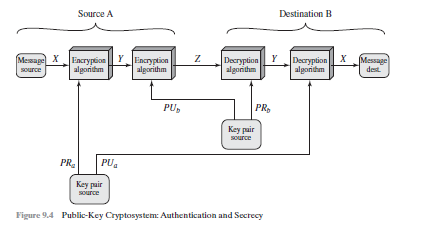
i.e., X=D (PRb ,Y).

The above figure provides confidentiality but not authentication.

Using sender’s private key for encryption provides authentication but not confidentiality as in figure below



It is possible to provide confidentiality and authentication together by the double use of public key scheme.



*Z* = E(*PUb*, E(*PRa*, *X*))

*X* = D(*PUa*, D(*PRb*, *Z*))

**Applications for Public-Key Cryptosystems**

* **Encryption /decryption:** The sender encrypts a message with the recipient’s public key.
* **Digital signature:** The sender “signs” a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
* **Key exchange:** Two sides cooperate to exchange a session key

**Public key cryptanalysis**

Public key encryption scheme is vulnerable to a brute force attack. The counter measure is to use large keys.