Traffic confidentiality and key distribution

By: Dr. Upma Jain

Traffic Confidentiality

In some cases, users are concerned about security from traffic analysis. Knowledge about the number and length of messages between nodes may enable an opponent to determine who is talking to whom. This can have obvious implications in a military conflict.

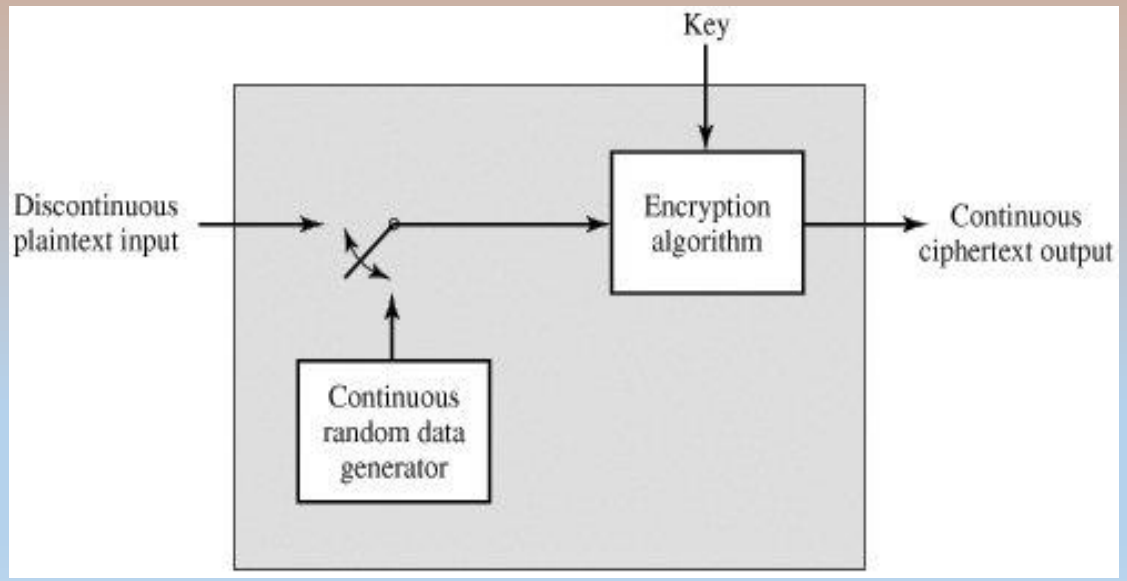
Even in commercial applications, traffic analysis may yield information that the traffic generators would like to conceal. **[MUFT89] lists the following types of information that can be derived from a traffic analysis attack:**

- Identities of partners
- How frequently the partners are communicating
- Message pattern, message length, or quantity of messages that suggest important information is being exchanged
- The events that correlate with special conversations between particular partners.

- Another concern related to traffic is the use of traffic patterns to create a **covert channel**. A covert channel is a means of communication in a fashion unintended by the designers of the communications facility. Typically, the channel is used to transfer information in a way that violates a security policy. For example, an employee may wish to communicate information to an outsider in a way that is not detected by management and that requires simple eavesdropping on the part of the outsider. The two participants could set up a code in which an apparently legitimate message of a less than a certain length represents binary zero, whereas a longer message represents a binary one. Other such schemes are possible.

- **Link Encryption Approach**

With the use of link encryption, network-layer headers (e.g., frame or cell header) are encrypted, reducing the opportunity for traffic analysis. However, it is still possible in those circumstances for an attacker to assess the amount of traffic on a network and to observe the amount of traffic entering and leaving each end system. An effective countermeasure to this attack is traffic padding, illustrated in Figure.



Link to link encryption

- **Traffic padding** produces ciphertext output continuously, even in the absence of plaintext. A continuous random data stream is generated. When plaintext is available, it is encrypted and transmitted. When input plaintext is not present, random data are encrypted and transmitted. This makes it impossible for an attacker to distinguish between true data flow and padding and therefore impossible to deduce the amount of traffic.

End-to-End Encryption Approach

- Traffic padding is essentially a link encryption function. If only end-to-end encryption is employed, then the measures available to the defender are more limited.
- For example, if encryption is implemented at the application layer, then an opponent can determine which transport entities are engaged in dialogue. If encryption techniques are housed at the transport layer, then network-layer addresses and traffic patterns remain accessible.
- One technique that might prove useful is to pad out data units to a uniform length at either the transport or application level. In addition, null messages can be inserted randomly into the stream. These tactics deny an opponent knowledge about the amount of data exchanged between end users and obscure the underlying traffic pattern.

Key Distribution

- For symmetric encryption to work, the two parties to an exchange must share the same key and that key must be protected from access by others.
- Furthermore frequent changes are usually desirable to limit the amount of data compromised if an attacker learns the key.
- Therefore, 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 & 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. 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 key on the encrypted links to A and B.

Options 1 and 2 call for manual delivery of a key.

For link encryption, this is a reasonable requirement, because each link encryption device is going to be exchanging data only with its partner on the other end of the link.

However, for end to-end encryption, 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. The problem is especially difficult in a wide area distributed system.

The scale of the problem depends on the number of communicating pairs than must be supported. It end-to-end encryption is done at a network or IP level, then 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**. Thus, a network may have hundreds of hosts but thousands of users and processes.
- A network using node-level encryption **with 1000 nodes would conceivably need to distribute as many as half a million keys.**
- If that same net work supported 10,000 applications, then as many as 50 million keys may be required for application-level encryption.
- **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**.
- Furthermore, the initial distribution of potentially millions of keys must still be made.
- **For end-to-end encryption, some variation on option 4 has been widely adopted**. In this scheme, **a key distribution centre** 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 centre for purposes of key distribution**."

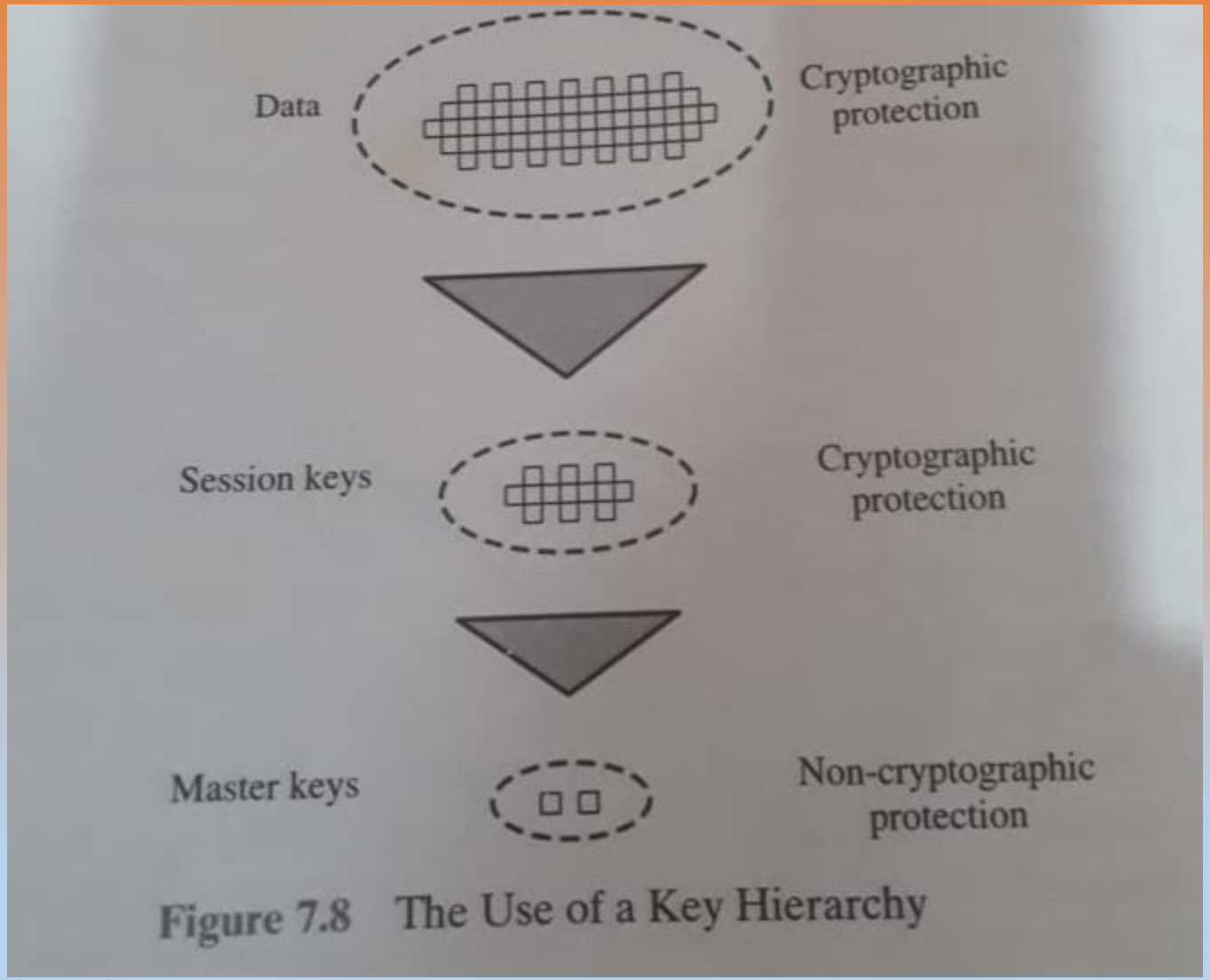
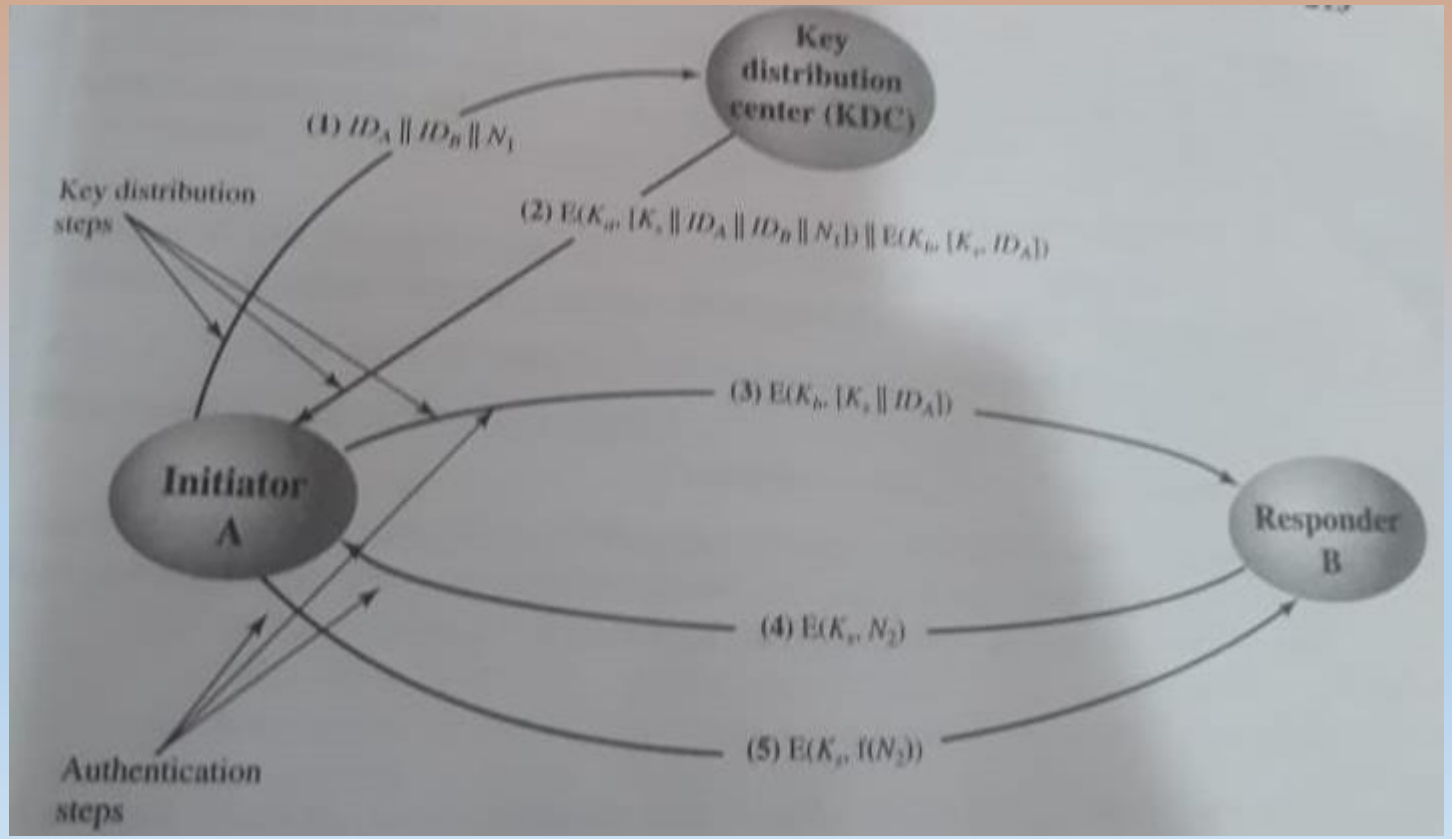


Figure 7.8 The Use of a Key Hierarchy

- The **use of a key distribution centre is based on the use of hierarchy of keys**.
- At a **minimum, two levels of keys are** used.
- Communication between end systems is encrypted using a **temporary key**, often referred as **session key.**
- Typically, **the session key is** used for the duration of a logical connection such as frame relay or transport connection and then discarded.
- **Session key is** obtained from the key distribution centre over the same networking facilities used for end-user communication.
- **Session keys** are transmitted in encrypted form encrypted form, using **a master key** that is shared by the key distribution centre and end system or user.
- For **each end system of user, there is a unique master key that it shares the key distribution centre.**
- Of course, these **master keys must be distributed in some fashion**. However, the **scale of the problem is vastly reduced**. 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 a one for each entity**. **Thus, master keys can be distributed in some noncryptographic way, such as physical delivery.**

A key distribution Scenario

- The key distribution concept can be deployed in a number of ways A typical wea illustrated in Figure.



- Let us assume that **user A wishes to establish a logical connection with B and requires a one-time session key** to protect the data transmitted over the connection.
- A has a master **key, Ka** known only to itself and the KDC, similarly, B has **master key Kb**, with the KDC. 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 N1 for this transaction, which we refer to as a nonce. The nonce may be a timestamp or counter, or a random number; the minimum is that it differs with each request. Also, to prevent masquerade, it should be difficult for opponent to. Thus, a random number is a good choice for nonce.**

2. The KDC responds with a message encrypted using Ka. Thus A is the only one who can successfully read the message, and A knows that is originated at the KDC. 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.
- Thus, A can verify that its original request was not altered before reception by the KDC and, because of the nonce, that this is not a replay of some previous request.
- In addition the message includes two items intended for B.
  - One time session key, ks, to be used for session.
  - An identifier of A, IDA.

  These last two items are encrypted with kb. They are sent to B to establish the connection and prove A's identity.

3. A stores the session key for use in the upcoming session and forward to B the information that originated at the KDC for B, namely, E(kb, [ks || IDA]). Because the information is encrypted with kb, it is protected from eavesdropping. B now knows the session key (ks), knows that the other party is A, and knows that the information originated at the KDC.(because it is encrypted using Kb).

At this point, a session key has been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable.

4. Using the newly minted session key for encryption. B sends a nonce. N2, to A.

5. Also using ks, A responds with f(N2), where f is a function that performs some transformation on N2 (eg, adding one).

These steps assures B that original message it received was not a replay.

Note that the actual key distribution involves only steps 1 through but th steps 4 and 5, as well as 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, 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 desired 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, depending on the size of the user population and the geographic scope of the internet.
- Hierarchical scheme minimizes the effort involved in master key distribution because most master keys are those shared by a local KDC with its local entities. Furthermore, such a scheme limits the damage of a faulty or subverted KDC local area only.

Session Key Lifetime

- The more **frequently session keys are exchanged more secure they are**, because the opponent has less ciphertext to work with for any session key.
- On the other hand, the distribution of session keys delays the start of any exchange and places the burden on network capacity. A security officer 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, connection** is open, using a new session key for each new session. new session.
- If a logical **connection has a very long lifetime, then it would be prudent to change the session key periodically, perhaps every** time the PDU (protocol da unit) sequence number cycles.
- For a connectionless protocol, such as a transaction-oriented protocol, there is no explicit connection initiation or termination. Thus, 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

- KDC has many variations, one of which is described here.
- It 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 noteworthy element of this approach is a session security module (SSM), which may consists of functionality at one protocol layer, that performs end-to-end encryption and obtains session keys on behalf of its host or terminal.
- **The steps involved in establishing a connection are shown in the figure.**

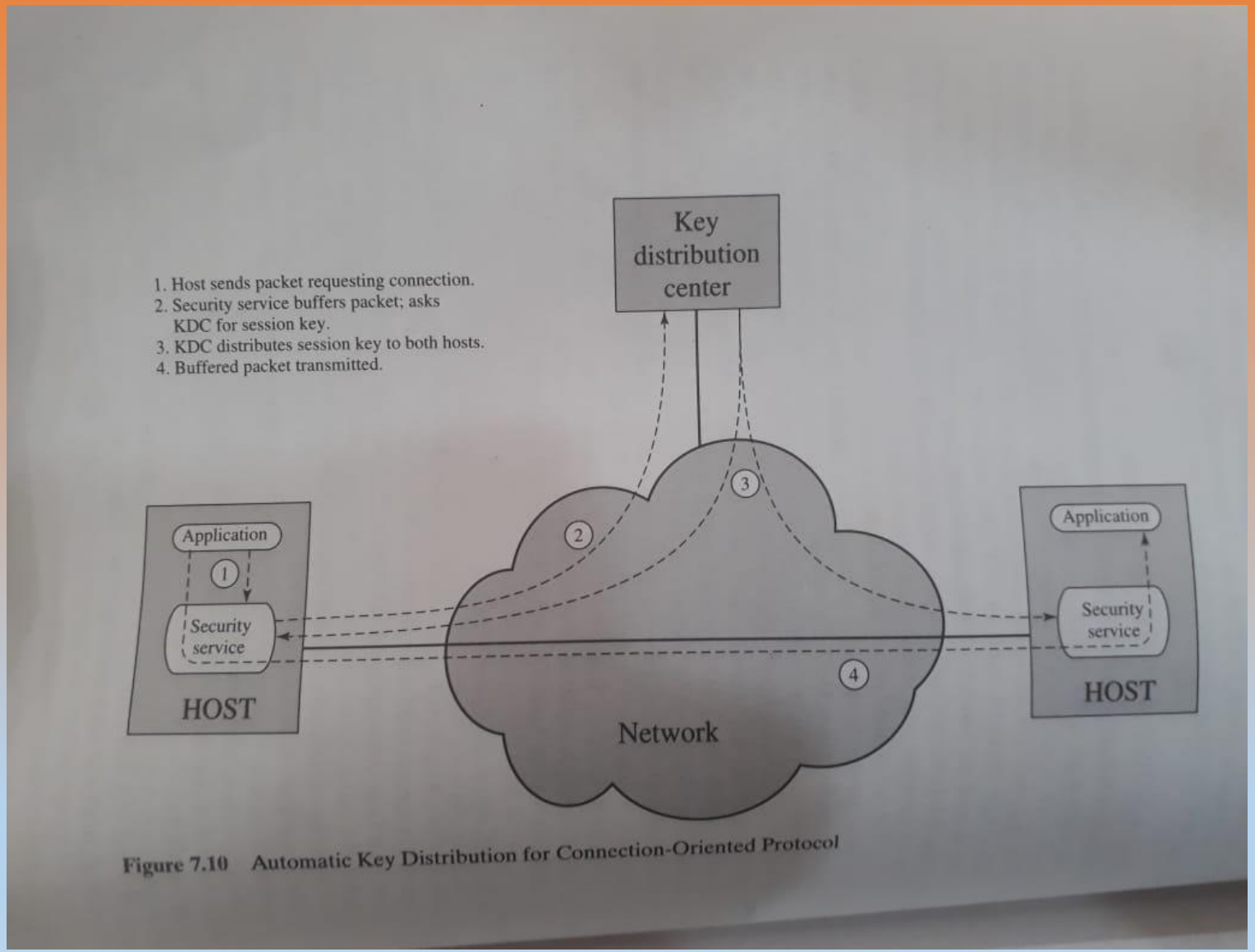


Figure 7.10 Automatic Key Distribution for Connection-Oriented Protocol

- Step 1: When one host wishes to set up a connection to another host, it transmits a connection request packet.
- Step 2: The SSM saves that packet and applies to the KDC for permission to establish the connection (step 2).
- step 3: 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
- The requesting SSM can now release the connection request packet, and a connection is setup between the two end systems (step 4).
- All user data exchanged between the two end systems are encrypted by their respective SSMs using the one-time session key.
- The automated key distribution approach provides the flexibility and dynamic characteristics needed to allow a number of terminal users to access a number of hosts and for the hosts to data with each other.

Decentralized Key Control

- The use of a key distribution centre imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distribution is fully decentralized.
- Although full decentralization is not practical for large 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 n(n-1)/2 master keys for a configuration with n 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, N1.
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 of f(N1), and another nonce N2.
3. Using the new session key, A returns f(N2) to B.

Thus, although each node must maintain at most (n-1) master keys, as many keys as required may be generated and used. Because the message transferred using the master key are short, cryptanalysis is difficult. As before, session keys are used for only a limited time to protect them.

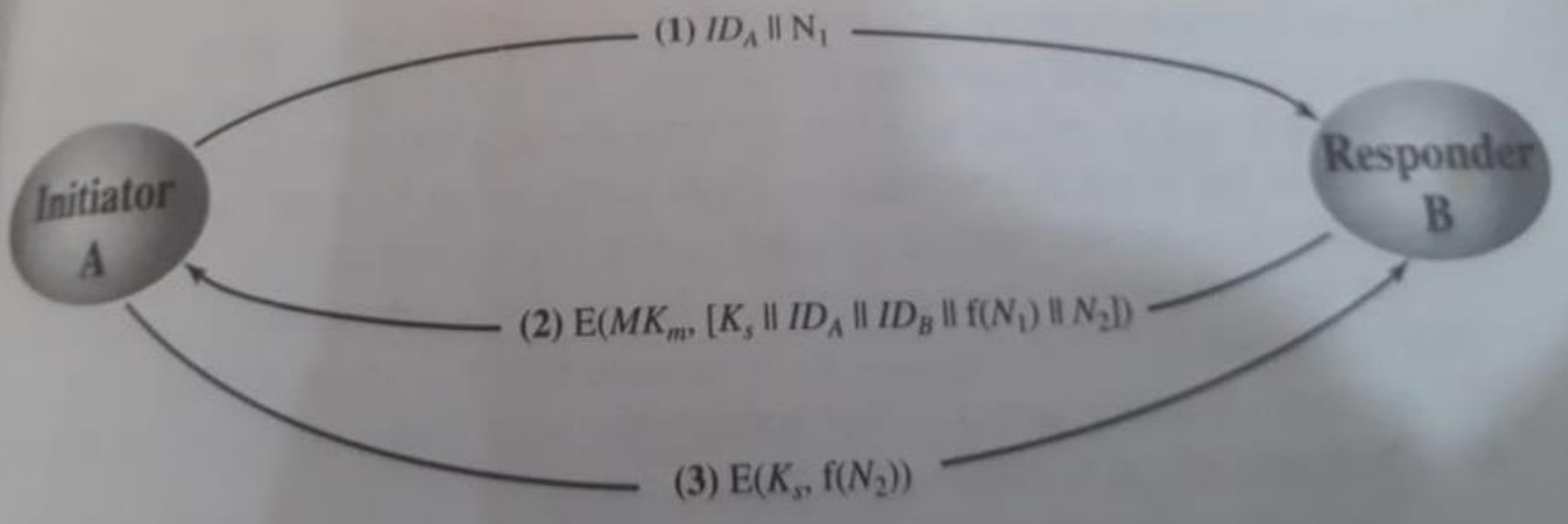


Figure 7.11 Decentralized Key Distribution

Controlling Key Usage

- 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 may also be desirable impose some control on the way in which automatically distributed keys are used. For example, in addition to separating master keys from 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.

- To illustrate the value of separating keys by type, consider the risk that a master key is imported as a data-encrypting key into a device Normally, the master key is physically secured within the cryptographic hardware of the key distribution center and of the end systems Session keys encrypted with this master key are available to application programs, as are the data encrypted with such session keys.
- However, it master key is treated as a session key, it may be possible for an unauthorized application to obtain plaintext of session keys encrypted with that master key.

- **Thus, 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**.
  - Associate key with a tag
  - Control vector
- **Key with tag**: The proposed technique is for use with DES and makes use of the extra 8 hits in each 64-bit DES key. That is, the 8 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.

  Because the tag is embedded in the key, it is encrypted along with key is distributed, the providing protection .The drawbacks of this, the tag length is limited to 8 bits, limiting its flexibility and function because the tag is not transmitted in clear form, it can be used only decryption, limiting the ways in which key use can be controlled.

- Control vectors: In this scheme each **session key has an associated control vector** consisting of a number of fields that specify the uses and restrictions for that session key.
- The **length of the control vector may vary**.
- The control vector is cryptographically coupled with the key at the time of key generation at the KDC. The coupling and decoupling processes is shown in Figure.
- As a first step the control vector is passed **through a hash function,** that produces a value whose length is equal to the encryption key length.
- A hash function values from a larger range into a smaller range, with a reasonably uniform spread.

- The hash value is then XORed with the master key to produce an output that is used as the key input for encrypting the session key. Thus,

Hash value = H= h(CV)

Key input = Km XOR H

Ciphertext = E([Km XOR H], Ks)),

where Km is the master key and Ks, is the session key. The session key is recovered in plaintext by the reverse operation:

D= ([km XOR H], E([km XOR H], ks))

When a session key is delivered to a user from the KDC, it is accompanied by the control vector in clear form. The session key can be recovered only by both the master key that the user shares with the KDC and the control vector. Thusthe linkage between the session key and its control vector is maintained.

Use of the control vector has two advantages over use of an 8-bit tag. First there is no restriction on length of the control vector, which enables arbitrarily complex controls to be imposed on key use. **Second**, the control vector is available in clear form at all stages of operation. Thus control of key use can be exercised in multiple locations.

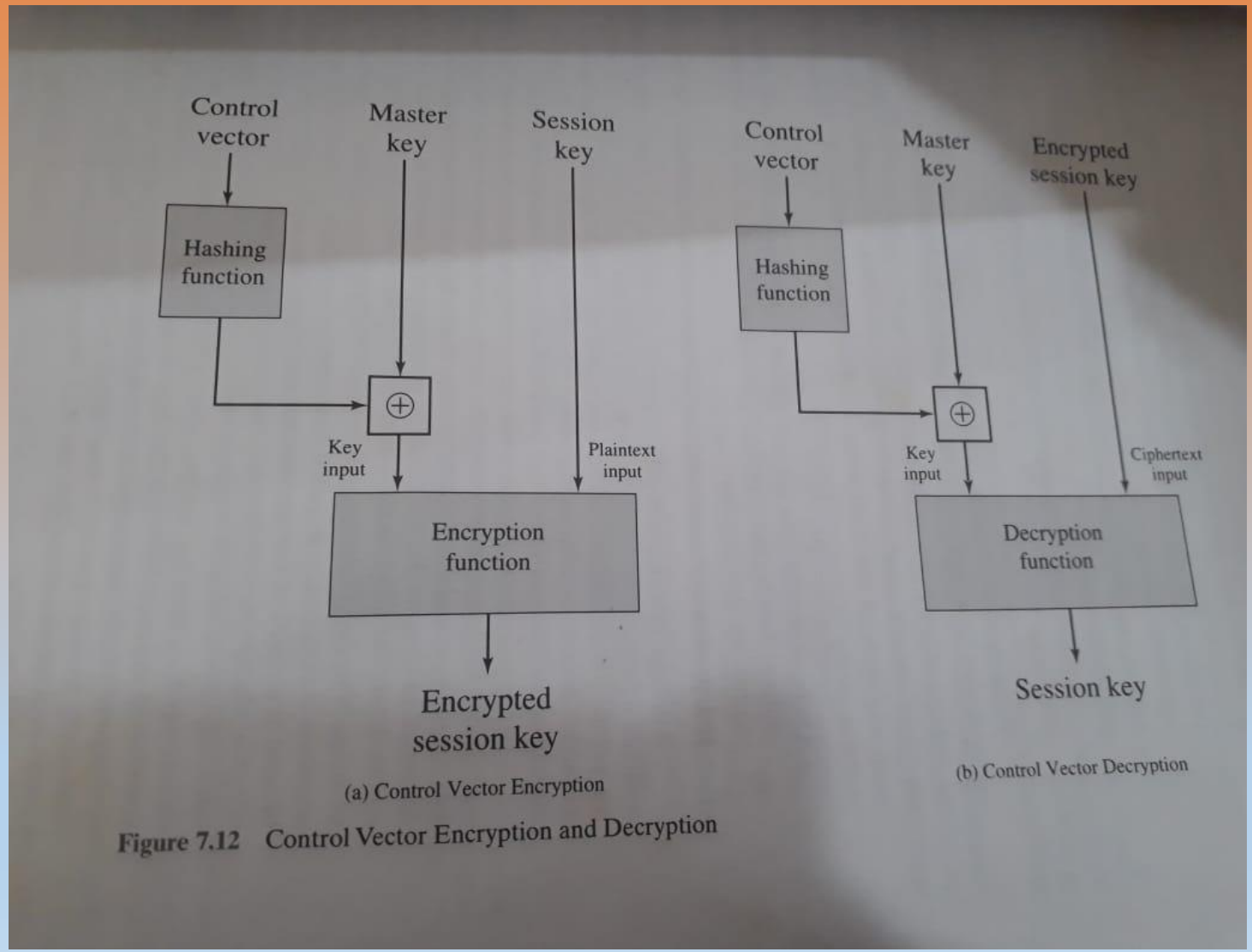


Figure 7.12 Control Vector Encryption and Decryption

Usage of random number

- Random number plays important role in encryption.
- A number of network security algorithms based on cryptography make use of random numbers
- Session key generation, whether done by a key distribution centre or by one of the principals.
- Generation of keys for the public-key encryption algorithm.

These applications give rise to two distinct and not necessarily compatible requirements for a sequence of random numbers: randomness and unpredictability.

The following two criteria are used to that a sequence of numbers is random

- **Uniform distribution:** The distribution of numbers in the sequence should be uniform, that is, the frequency of occurrence of each of the numbers should approximately the same.

Independence: No one value in the sequence can be inferred from the other.

Although there are well-defined tests for determining that a sequence of numbers matches a particular distribution, such as the uniform distribution, there is no such s to "prove" independence. Rather, a number of tests can be applied to demonstrate if a sequence does not exhibit independence. The general strategy is to apply a number of such tests until the confidence that independence exists is sufficiently strong.

Unpredictability: In applications such as reciprocal authentication and session key generation, the requirement is not so much that the sequence of numbers be statistically random but that the successive members of the sequence are unpredictable. With "true" random sequences, each number is statistically independent of other numbers in the sequence and therefore unpredictable. However, true random numbers are seldom used, rather, sequences of numbers that appear to he random are generated by some algorithm. In this latter case, care must be taken that an opponent not be able to predict future elements of the sequence on the basis of earlier elements.

Puedorandom number

- **Pseudo Random Number Generator(PRNG)**

Refers to an algorithm that uses mathematical formulas to produce sequences of random numbers. PRNGs generate a sequence of numbers approximating the properties of random numbers. A PRNG starts from an arbitrary starting state using a **seed state**. Many numbers are generated in a short time and can also be reproduced later, if the starting point in the sequence is known. Hence, the numbers are **deterministic and efficient**.

- **Why do we need PRNG?**

With the advent of computers, programmers recognized the need for a means of introducing randomness into a computer program. However, surprising as it may seem, it is difficult to get a computer to do something by chance as computer follows the given instructions blindly and is therefore completely predictable. It is not possible to generate truly random numbers from deterministic thing like computers so PRNG is a technique developed to generate random numbers using a computer.

Characteristics of PRNG

- **Efficient:** PRNG can produce many numbers in a short time and is advantageous for applications that need many numbers
- **Deterministic:** A given sequence of numbers can be reproduced at a later date if the starting point in the sequence is known.Determinism is handy if you need to replay the same sequence of numbers again at a later stage.
- **Periodic:** PRNGs are periodic, which means that the sequence will eventually repeat itself. While periodicity is hardly ever a desirable characteristic, modern PRNGs have a period that is so long that it can be ignored for most practical purposes

Applications of PRNG

- PRNGs are suitable for applications where many random numbers are required and where it is useful that the same sequence can be replayed easily. Popular examples of such applications are **simulation and modeling applications**. PRNGs are not suitable for applications where it is important that the numbers are really unpredictable, such as **data encryption and gambling.**

Linear Congruential Generator

- **Linear Congruential Method** is given by Lehmer. It is a class of Pseudo Random Number Generator (PRNG) algorithms used for generating sequences of random-like numbers in a specific range. This method can be defined as:
- *where,*
- ***X,*** *is the sequence of pseudo-random numbers*
  m, *(> 0) the modulus*
  a, *(0, m) the multiplier*
  c, *[0, m) the increment*
  X_0 *[0, m) Initial value of sequence known as seed*

- If *m, a, c, and X_0 are integers, then this technique will produce a sequence of integers with each integer in the range 0<=Xn<m.*
- *The selection of values for a,c and m is critical in developing a good random number generator.*
- *For example consider a= c=1. The sequence produced I not satisfactory.*
- *Now consider values a=7, c= 0, m = 32 and X0=1. this generates the sequence (7,17,23,1, 7,17..) which is also clearly unsatisfactory. Out of 32 possible values only 4 are used.*
- *If we change a to 5 the period will be 8.*

- Period, We would like to be very large, so that there is the potential for producing long series of distinct random numbers.
- A common criterion is that m be nearly equal to the maximum representable nonnegative integer for a given computer. Thus, a value of m near to or equal to 2^31 is typically chosen.
- **PARK proposed three tests to be used in evaluating a random generator:**
- **T1:** The function should be a full-period generating function. That is the function should generate all the numbers between 0 and m.
- **T2:** The generated sequence should appear random. Because it is generated deterministically, the sequence is not random. There is a variety of statistical test that can be used to assess the degree to which a sequence exhibits randomness.
- **T3:** The function should implement efficiently with 32 bit arithmetic.

- With appropriate values of a c and m, these three tests can be passed With respect to T1, it can be shown that if m is prime and c=0, then for certain values of a, the period of the generating function is m-1, with only the value 0 missing. For 32-bit arithmetic, a convenient prime value of m is (2^31)-1.
- **The strength of the linear congruential algorithm is that if the multiplier and modulus are properly chosen**, the resulting sequence of numbers will be statistically indistinguishable from a sequence drawn at random. **But there is nothing random at all about the algorithm, apart from the choice of the initial value Xo**. Once that value in chosen, the remaining numbers in the sequence follow deterministically.

- This has implications for cryptanalyst, If an opponent knows that the linear congruential algorithm is being used and if the parameters are known (eg.a=7, c=0, m=2^13. then once a single number is discovered, all subsequent numbers are known.
- Suppose the opponent is able to determine the values of X0, X1, X2, and X3. Then

$$X1 = (aX0 + c) \bmod m$$

$$X2 = (aX1 + c) \bmod m$$

$$X3 = (aX2 + c) \bmod m$$

Above equations can be solved for a, c and m.

- *m, a, c, and X_0 should be chosen appropriately to get a period almost equal to m.*
- For a = 1, it will be the additive congruence method. For c = 0, it will be the multiplicative congruence method.
- **Approach:**
- Choose the seed value $X_{0,}$ Modulus parameter m, Multiplier term a, and increment term c.
- Initialize the required amount of random numbers to generate (say, an integer variable *no Of RandomNums*).
- Define a storage to keep the generated random numbers (here, *vector* is considered) of size *no Of RandomNums*.
- Initialize the 0^{th} index of the vector with the seed value.
- For rest of the indexes follow the Linear Congruential Method to generate the random numbers.

  randomNums[i] = ((randomNums[i – 1] * a) + c) % m

Finally, return the random numbers.

Blum Blum sub generator

- A popular approach for generating secure puedorandom numbers is known as the Blum, Blum, Shub generator, named for its developer.
- It it cryptrogaraphically very strong.

Procedure:

- Choose two large prime numbers p and q, such that both have a remainder of 3when divided by 4.

$$P \bmod 4 = 3 = q \bmod 4$$

- Next choose random number s such that s is relatively prime to n, means neither p nor q shold be a factor of s.

- Then BBS produces the sequence according to the following algorithm

$$X_0 = s^2 \bmod n$$

For i = 1 to inf

$$X_i = (x_{i-1})^2 \bmod n$$

$$B_i = X_i \bmod 2$$

- Thus the Least significant bit is taken at each iteration.

- The BBS is referred to **as a cryptographically secure pseudorandom bit generator (CSPRBG**). A **CSPRBG is defined as one that passes the next bit test turn, is defined** as follows [MENE97) **A pseudorandom number generator is said to pass the next-bit test: if there is not a polynomial time algorithm that input on input of the the first k bits of an output sequence, can predict the (k+1) bit with probability significantly greater than ½.**