**UNIT – I**

**Terminology**

|  |  |
| --- | --- |
| **Security** | The quality or state of being secure—to be free from danger   * **Computer Security**-generic name for the collection of tools designed to protect data and to thwart hackers * **Network Security**-measures to protect data during their transmission * **Internet Security**-measures to protect data during their transmission over a collection of interconnected networks |
| **Security Attack** | Any action that compromises the security of information owned by an organization  Generic types of attacks   * Passive attacks * Active attacks |
| **Passive Attacks** | A passive attack attempts to learn or make use of information from the system but does not affect system resources.  A passive attack, in computing security, is an attack characterized by the attacker listening in on communication. In such an attack, the intruder/hacker does not attempt to break into the system or otherwise change data  Goal: to obtain information that is being transmitted;  Passive attacks basically mean that the attacker is eavesdropping (listen secretly to or over-hear private conversation)  Two types of passive attacks are   * The release of message contents and * Traffic analysis. |
| **Active attack** | An active attack attempts to alter system resources or affect their operation. Active attacks involve some modification of the data stream or the creation of a false stream.  Active attacks can be subdivided into four categories:   * masquerade, * replay, * modification of messages, and * Denial of service. |
| **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. |
| **Modification** | **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 |
| **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; |
| **Authentication** | The authentication service is concerned with assuring that a communication is authentic. In the case of a single message, such as a warning or alarm signal, the function of the authentication service is to assure the recipient that the message is from the source that it claims to be from. In the case of an ongoing interaction, such as the connection of a terminal to a host, two aspects are involved.  First, at the time of connection initiation, the service assures that the two entities are authentic (that is, that each is the entity that it claims to be).  Second, the service must assure that the connection is not interfered with in such a way that a third party can masquerade as one of the two legitimate parties for the purposes of unauthorized transmission or reception. |
| **Access Control** | **Access control** is the ability to limit and control the access to host systems and applications via communications links. To achieve this, each entity trying to gain access must first be identified, or authenticated, so that access rights can be tailored to the individual. |
| **Data Confidentiality** | Confidentiality is the protection of transmitted data from passive attacks. With respect to the content of a data transmission, several levels of protection can be identified. The broadest service protects all user data transmitted between two users over a period of time. |
| **Data Integrity** | As with confidentiality, integrity can apply to a stream of messages, a single message, or selected fields within a message. Again, the most useful and straightforward approach is total stream protection. A connection-oriented integrity service deals with a stream of messages and assures that messages are received as sent with no duplication, insertion, modification, reordering, or replays. The destruction of data is also covered under this service. Thus, the connection-oriented integrity service addresses both message stream modification and denial of service. |

**Concepts**

|  |  |
| --- | --- |
| **Symmetric encryption** | **Symmetric encryption** is a form of cryptosystem in which encryption and decryption are performed using the same key. It is also known as conventional encryption. Symmetric encryption, also referred to as conventional encryption or single-key encryption.  ◆ Symmetric encryption transforms plaintext into ciphertext using a secret key and an encryption algorithm. Using the same key and a decryption algorithm, the plaintext is recovered from the ciphertext.  ◆ The two types of attack on an encryption algorithm are cryptanalysis, based on properties of the encryption algorithm, and brute-force, which involves trying all possible keys.  ◆ Traditional (precomputer) symmetric ciphers use substitution and/or transposition techniques. Substitution techniques map plaintext elements (characters, bits) into ciphertext elements. Transposition techniques systematically transpose the positions of plaintext elements. |
| **Feistel Cipher** | Feistel proposed [FEIS73] that we can approximate the ideal block cipher by utilizing the concept of a product cipher, which is the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers.The essence of the approach is to develop a block cipher with a key length of k bits and a block length of *n bits, allowing a total of 2k possible transformations, rather than the 2n! transformations available with the ideal block cipher.*  In particular, Feistel proposed the use of a cipher that alternates substitutions and permutations, where these terms are defined as follows:   * **Substitution:** Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements. * **Permutation:** A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed.   Feistel’s is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates *confusion* and *diffusion* functions  FEISTEL CIPHER STRUCTURE The left-hand side of Figure 3.3 depicts the structure proposed by Feistel. The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key . The plaintext block is divided into two halves, L0 and R0. The two halves of the data pass through n rounds of processing and then combine to produce the ciphertext block. Each round i has as inputs *Li*-1 and *Ri*-1 derived from the previous round, as well as a subkey *Ki* derived from the overall K. In general, the subkeys *Ki are different from K and from each othe.*  All rounds have the same structure. A **substitution** is performed on the left half of the data. This is done by applying a round function F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round subkey *Ki.*    **Feistel Cipher structures** |
| **Data Encryption Standard (DES)** | * DES is a Symmetric-key algorithm for the encryption of electronic data. * DES originated at IBM in 1977 & was adopted by the U.S Department of Defence. Now it is under the NIST (National Institute of Standard & Technology) * Data Encryption Standard (DES) is a widely-used method of data [encryption](http://searchsecurity.techtarget.com/definition/encryption) using a private (secret) key * DES applies a 56-bit key to each 64-bit block of data. The process can run in several modes and involves 16 rounds or operations.     **Inner workings of DES:**  DES (and most of the other major symmetric ciphers) is based on a cipher known as the Feistel block cipher. This was a block cipher developed by the IBM cryptography researcher Horst Feistel in the early 70’s. It consists of a number of rounds where each round contains bit-shuffling, non-linear substitutions (S-boxes) and exclusive OR operations. Most symmetric encryption schemes today are based on this structure (known as a feistel network).  **Overall structure**  DES (and most of the other major symmetric ciphers) is based on a cipher known as the Feistel block cipher.  Looking at the left-hand side of the figure, we can see that the processing of the plaintext proceeds in three phases.   * First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the *permuted input*. * This is followed by a phase consisting of sixteen rounds of the same function, which involves both permutation and substitution functions. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the **preoutput**. * Finally, the preoutput is passed through a permutation that is the inverse of the initial permutation function, to produce the 64-bit cipher text. With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher,   The right-hand portion of below shows the way in which the 56-bit key is used. Initially, the key is passed through a permutation function. Then, for each of the sixteen rounds, a *subkey* (*Ki* ) is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.    **Initial Permutation:** The initial permutation and its inverse are defined by tables, as shown in Tables 3.2a and 3.2b, respectively. 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 a numbered input bit in the output, which also consists of 64 bits.    To see that these two permutation functions are indeed the inverse of each other, consider the following 64-bit input M:    Where Mi is a binary digit. Then the permutation *X* = (IP(*M*)) is as follows:    **DETAILS OF SINGLE ROUND**  Below figure shows the internal structure of a single round. Again, begin by focusing on the left-hand side of the diagram. The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right). As in any classic Feistel cipher, the overall processing at each round can be summarized in the following formulas:      The round key Ki is 48 bits. The *R* input is 32 bits. This *R* input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the *R* bits (Table 3.2c).The resulting 48 bits are XORed with Ki . This 48-bit result passes through a substitution function that produces a 32-bit output, which is permuted as defined by Table 3.2d. The role of the S-boxes in the function F is illustrated in Figure 3.7.The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output. These transformations are defined in Table 3.3, which is interpreted as follows : The first and last bits of the input to box *Si* form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for . 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 representation to produce the output.    **KEY GENERATION** Returning to above all figures, we see that a 64-bit key is used as input to the algorithm.The bits of the key are numbered from 1 through 64; every eighth bit is ignored, as indicated by the lack of shading in Table 3.4a.The key is first subjected to a permutation governed by a table labeled Permuted Choice One (Table 3.4b)  The resulting 56-bit key is then treated as two 28-bit quantities, labelled C0 and D0. At each round, Ci-1 and Di-1 are separately subjected to a circular left shift or (rotation) of 1 or 2 bits, as governed by Table 3.4d.These shifted values serve as input to the next round. They also serve as input to the part labeled Permuted Choice Two (Table 3.4c), which produces a 48-bit output that serves as input to the Function F(Ri-1, Ki).  **DES DECRYPTION:**  Whatever process we following in the encryption that process is used for decryption also but the order of key is changed on input message (cipher text).  Reverse order of keys are K16, K15 ,……, K1. |
| **Triple DES** | Multiple encryption is a technique in which an encryption algorithm is used multiple times. In the first instance, plaintext is converted to ciphertext using the encryption algorithm. This ciphertext is then used as input and the algorithm is applied again. This process may be repeated through any number of stages. |
| **Double DES** | The simplest form of multiple encryption has two encryption stages and two keys (Figure 4.la). Given a plaintext P and two encryption keys K, and K,, ciphertext C is generated as    Decryption requires that the keys be applied in reverse order: |
| **MEET-IN-THE-MIDDLE ATTACK** | the use of double DES results in a mapping that is not equivalent to a single DES encryption. But there is a way to attack this scheme, one that does not depend on any particular property of DES but that will work against any block encryption cipher.  meet-in-the-middle attack is based on the observation that, if we have    Given a known pair, (*P*, *C*) the attack proceeds as follows. First, encrypt for all 256 possible values of K1. Store these results in a table and then sort the table by the values of .  Next, decrypt *C using all* 256 *possible values of* K2. As each decryption is produced, check the result against the table for a match. 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 P, there are 264 possible ciphertext values that could be produced by double DES. Double DES uses, in effect, a 112-bit key, so that there are 2112 possible keys. Therefore, on average, for a given plaintext , the number of different 112-bit keys that will produce a given ciphertext C is 2112/264=248 Thus, the foregoing procedure will produce about 248 false alarms on the first (*P*, *C*) |
| **Triple DES with Two Keys**: | An obvious counter to the meet-in-the-middle attack is to use three stages of encryption with three different keys. This raises the cost of the meet-in-the-middle attack to 2112 bits, which may be somewhat unwieldy.  As an alternative, Tuchman proposed a triple encryption method that uses only two keys [TUCH79]. The function follows an encrypt-decrypt-encrypt (EDE) sequence |
| **Triple DES with Three Keys :** | Although the attacks just described appear impractical, anyone using two-key 3DES may feel some concern. Thus, many researchers now feel that three-key 3DES is the preferred alternative (e.g., [KALI96a]).Three-key 3DES has an effective key length of 168 bits and is defined as    Backward compatibility with DES is provided by putting K3 = K2 or K1 = K2  A number of Internet-based applications have adopted three-key 3DES, including PGP and S/MIME. |
| **Electronic Codebook (ECB)** | Message is broken into independent blocks which are encrypted  ● Each block is encoded independently of the other blocks  Ci = DESK (Pi)    ● Applications |
| **Cipher Block Chaining Mode** | ● Message is broken into blocks  ● “Linked” together during encryption  ● each previous cipher block is chained with current plaintext block  ● Initial Vector (IV) used to start process  ● Applications: bulk data encryption, authentication    ● Each ciphertext block depends on all message blocks  ● A change in a message block affects all ciphertext blocks after the change (as well as the original block)  ● Need Initial Value (IV) known to sender & receiver  – however if IV is sent in the clear, an attacker can change bits of the first block, and change IV to compensate  – hence either IV must be a fixed value - or it must be sent encrypted in ECB mode before rest of message |
| **Cipher Feedback Mode** | ● Message is treated as a stream of bits  ● Added to the output of the block cipher  ● Result is feed back for next stage (hence name)  ● Standard allows any number of bit (1,8 or 64 or whatever) to be feed back  – denoted CFB-1, CFB-8, CFB-64 etc  ● CFB-64 is used most often (most efficient)  ● Applications: stream data encryption, authentication    ● Appropriate when data arrives in bits/bytes  ● Most common stream mode  ● Block cipher is used in encryption mode at both ends!  ● Errors propagate for several blocks after the error (depending on s) |
| **Output feedback mode** | Message treated as a stream of bits  ● Output of cipher is added to message  ● Output is then fed back  ● feedback is independent of message  ● Applications: stream encryption over noisy channels      ● Used when error feedback is a serious problem  ● Superficially similar to CFB  – but feedback is from the output of cipher and is independent of message  ● a variation of a Vernam cipher  – hence must never reuse the same sequence (key+IV)  ● Sender and receiver must remain in sync, and some recovery method is needed to ensure this occurs  ● Originally specified with s-bit feedback in the standards  ● Subsequent research has shown that only OFB-64 should be used |
| **ADVANCED ENCRYPTION STANDARD** | * The Advanced Encryption Standard (AES) was published by the National Institute of Standards and Technology (NIST) in 2001. * AES is a block cipher intended to replace DES for commercial applications. * It uses a 128-bit block size and a key size of 128, 192, or 256 bits. * AES does not use a Feistel structure. Instead, each full round consists of four separate functions: byte substitution, permutation, arithmetic operations over a finite field, and XOR with a key.   **AES parameters:**   |  |  |  |  | | --- | --- | --- | --- | | Key size(words/bytes/bits) | 4/16/128 | 6/24/192 | 8/32/256 | | Plaintext block Size (words/bytes/bits) | 4/16/128 | 4/16/128 | 4/16/128 | | Number of rounds | 10 | 12 | 14 | | Round Key size (words/bytes/bits) | 4/16/128 | 4/16/128 | 4/16/128 | | Expanded key size (words/bytes) | 44/176 | 52/208 | 60/240 |   **Inner Workings of a Round**  The algorithm begins with an Add round key stage followed by 9 rounds of four stages and a tenth round of three stages. This applies for both encryption and decryption with the exception that each stage of a round the decryption algorithm is the inverse of it’s counterpart in the encryption algorithm. The four stages are as follows:  1. Substitute bytes  2. Shift rows  3. Mix Columns  4. Add Round Key  The tenth round simply leaves out the Mix Columns stage. The first nine rounds of the decryption algorithm consist of the following:  1. Inverse Shift rows  2. Inverse Substitute bytes  3. Inverse Add Round Key  4. Inverse Mix Columns  Again, the tenth round simply leaves out the **Inverse Mix Columns** stage. Each of these stages will now be considered in more detail.      **Substitute Bytes**  This stage (known as SubBytes) is simply a table lookup using a 16×16 matrix of byte values called an s-box. This matrix consists of all the possible combinations of an 8 bit sequence (28 = 16 × 16 = 256). However, the s-box is not just a random permutation of these values and there is a well defined method for creating the s-box tables. The designers of Rijndael showed how this was done unlike the s-boxes in DES for which no rationale was given. We will not be too concerned here how the s-boxes are made up and can simply take them as table lookups.    Again the matrix that gets operated upon throughout the encryption is known as **state**. We will be concerned with how this matrix is effected in each round. For this particular round each byte is mapped into a new byte in the following way: the leftmost nibble of the byte is used to specify a particular row of the s-box and the rightmost nibble specifies a column. For example, the byte {95} (curly brackets represent hex values in FIPS PUB 197) selects row 9 column 5 which turns out to contain the value {2A}.  This is then used to update the **state** matrix. Figure 7.3 depicts this idea. |
| **SUBSTITUTION TECHNIQUES** | A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols. If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with cipher text bit patterns.  (i)**Caesar cipher (or) shift cipher**  The earliest known use of a substitution cipher and the simplest was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing 3 places further down the alphabet.  e.g., Plain text : pay more mone  Cipher text: SDB PRUH PRQHB  Note that the alphabet is wrapped around, so that letter following „z‟ is „a‟.  For each plaintext letter p, substitute the cipher text letter c such that C =  E(p) = (p+3) mod 26  A shift may be any amount, so that general Caesar algorithm is C = E (p) = (p+k) mod 26  Where k takes on a value in the range 1 to 25. The decryption algorithm is simply P = D(C) = (C-k) mod 26  **(ii)Playfair cipher**  The best known multiple letter encryption cipher is the playfair, which treats digrams in the plaintext as single units and translates these units into cipher text digrams. The playfair algorithm is based on the use of 5x5 matrix of letters constructed using a keyword. Let the keyword be „monarchy‟. The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetical order.  The letter „i‟ and „j‟ count as one letter. Plaintext is encrypted two letters at a time according to the following rules:   1. Repeating plaintext letters that would fall in the same pair are separated with a filler letter such as „x‟. 2. Plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row following the last. 3. Plaintext letters that fall in the same column are replaced by the letter beneath, with the top element of the column following the last. 4. Otherwise, each plaintext letter is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter.  |  |  |  |  |  | | --- | --- | --- | --- | --- | | 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 | |  |  |  |  |  | |  |  |  |  |  |   Plaintext = meet me at the school house  Splitting two letters as a unit => me et me at th es ch ox ol ho us ex Corresponding cipher text => CL KL CL RS PD IL HY AV MP HF XL IU  **Strength of playfair cipher**   1. Playfair cipher is a great advance over simple mono alphabetic ciphers. 2. Since there are 26 letters, 26x26 = 676 diagrams are possible, so identification of individual digram is more difficult. 3. Frequency analysis is much more difficult.   **(iii)Polyalphabetic ciphers**  Another way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message. The general name for this approach is polyalphabetic cipher. All the techniques have the following features in common.   1. A set of related monoalphabetic substitution rules are used 2. A key determines which particular rule is chosen for a given transformation.   **(iv)Vigenere cipher**  In this scheme, the set of related monoalphabetic substitution rules consisting of 26 caesar ciphers with shifts of 0 through 25. Each cipher is denoted by a key letter. e.g.,  Caesar cipher with a shift of 3 is denoted by the key value 'd‟ (since a=0, b=1, c=2 and so on).  To aid in understanding the scheme, a matrix known as vigenere tableau is constructed  Each of the 26 ciphers is laid out horizontally, with the key letter for each cipher to its left. A normal alphabet for the plaintext runs across the top. The process of encryption is simple: Given a key letter X and a plaintext letter y, the cipher text is at the intersection of the row labeled x and the column labeled y; in this case, the ciphertext is V.  To encrypt a message, a key is needed that is as long as the message. Usually, the key is a repeating keyword.   |  |  | | --- | --- | | e.g., key | = d e c e p t i v e d e c e p t i v e d e c e p t i v e | | PT | = w e a r e d i s c o v e r e d s a v e y o u r s e l f | | CT | = ZICVTWQNGRZGVTWAVZHCQYGLMGJ |   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.  **Strength of Vigenere cipher**   1. There are multiple ciphertext letters for each plaintext letter 2. Letter frequency inforamiton is obscured.   **One Time Pad Cipher**  It is an unbreakable cryptosystem. It represents the message as a sequence of 0s and 1s. this can be accomplished by writing all numbers in binary, for example, or by using ASCII. The key is a random sequence of 0‟s and 1‟s of same length as the message.  Once a key is used, it is discarded and never used again. The system can be expressed as follows:  Ci = Pi Ki  Ci - ith binary digit of cipher text Pi - ith binary digit of plaintext  Ki - ith binary digit of key  – exclusive OR opearaiton  Thus the cipher text is generated by performing the bitwise XOR of the plaintext and the key. Decryption uses the same key. Because of the properties of XOR, decryption simply involves the same bitwise operation:  Pi = Ci Ki   |  |  | | --- | --- | | e.g., plaintext | = 0 0 1 0 1 0 0 1 | | Key | = 1 0 1 0 1 1 0 0 | |  | ------------------- |   ciphertext = 1 0 0 0 0 1 0 1  **Advantage:**   1. Encryption method is completely unbreakable for a ciphertext only attack.   **Disadvantages**   1. It requires a very long key which is expensive to produce and expensive to transmit. 2. Once a key is used, it is dangerous to reuse it for a second message; any knowledge on the first message would give knowledge of the second.   **TRANSPOSITION TECHNIQUES**  All the techniques examined so far involve the substitution of a cipher text symbol for a plaintext symbol. A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.  **Rail fence** is simplest of such cipher, in which the plaintext is written down as a sequence ofdiagonals and then read off as a sequence of rows.  Plaintext = meet at the school house  To encipher this message with a rail fence of depth 2, we write the message as follows:  m e a t e c o l o s   * t t h s H o h u e   The encrypted message is  MEATECOLOSETTHSHOHUE  **Row Transposition Ciphers**-A more complex scheme is to write the message in a rectangle,row by row, and read the message off, column by column, but permute the order of the columns. The order of columns then becomes the key of the algorithm. e.g., plaintext = meet at the school house   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Key = 4 | 3 | 1 | 2 | 5 | 6 | 7 | | PT = m | e | e | t | a | t | t | | h | e | s | c | h | o | o | | l | h | o | u | s | e |  |   CT = ESOTCUEEHMHLAHSTOETO  A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext. The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is more complex permutation that is not easily reconstructed. |

**Questions**

1. Answer the following:

a. What is Non-repudiation

b. Distinguish between stream and block ciphers

c. List out the problems of one time pad

d. Define Diffusion and Replay attack

e. What is session key

f. Name any two security standards

g. What is masquerading

h. Differentiate passive attack from active attack example

i. Distinguish between Dos and DDoS

2. Using play fair cipher algorithm encrypt the message FACTIONALISM using key

MONARCHY and explain

3. Explain the ceaser cipher and Mono alphabetic cipher

4. A).what is the difference between a mono alphabetic and a poly alphabetic cipher

B). what you mean by cryptanalysis

5. Explain about substitution and transposition techniques with two examples for each

6. A).What is security mechanism? Briefly describe the relation between security services

and mechanisms.

B).What are the various components of symmetric cipher model? Explain or

Briefly describe the requirements for secure use of conventional encryption

7. A).What is security service? Describe various security services

B).Briefly describe TCP session hijacking

8. A).What are format string vulnerabilities? How they can be fixed and exploited?

B).What is cryptography? Briefly describe the requirements for secure use of

conventional encryption

9. Briefly describe a model for network security with the help of a neat diagram

10. A).What is encryption? Briefly describe the types of attacks on encrypted messages

B).What are the key principles of Security

C).Write a note on spoofing

11. Briefly explain about the SQL injection.

12. Briefly describe TCP session hijacking.

13. Explain the terms related to Buffer overflow: A). Stack dumping B).Execute Payload.

14. Explain the model of conventional crypto system

15. A) Explain rail fence transposition technique

B) Explain the symmetric key encryption model

**UNIT – II**

**Terminology**

|  |  |
| --- | --- |
| **Primality Testing** | * The first stage of key-generation for RSA involves finding two large primes p, q * Because of the size of numbers used, must find primes by trial and error * Modern primality tests utilize properties of primes eg:  1. an-1 = 1 mod n where GCD(a,n)=1 2. all primes numbers 'n' will satisfy this equation    * some composite numbers will also satisfy the equation, and are called pseudo-primes. |
| **Euler Totient Function [[phi]](n)** | 1. if consider arithmetic modulo n, then a **reduced set of residues** is a subset of the complete set of residues modulo n which are relatively prime to n   o eg for n=10,  o the complete set of residues is {0,1,2,3,4,5,6,7,8,9} o the reduced set of residues is {1,3,7,9}   1. the number of elements in the reduced set of residues is called the **Euler Totient** **function [[phi]](n)** 2. there is no single formula for [[phi]](n) but for various cases count how many elements are excluded:   p (p prime) [[phi]](p) =p-1  pr (p prime) [[phi]](p) =pr-1(p-1)  p.q (p,q prime) [[phi]](p.q) =(p-1)(q-1)  several important results based on [[phi]](n) are: |
| **Theorem (Euler's Generalization)** | * let gcd(a,n)=1 then   1. a[[phi]](n) mod n = 1 |
| Fermat's Theorem | * 1. let p be a prime and gcd(a,p)=1 then   2. ap-1 mod p = 1 |
| **Field GF(qn)** | its elements are polynomials of degree (n-1) or lower   * 1. a(x)=an-1xn-1+an-2xn-2+...+a1x+a0   have residues for polynomials just as for integers  p(x)=q(x)d(x)+r(x)  and this is unique if deg[r(x)]<deg[d(x)]  if r(x)=0, then d(x) **divides** p(x), or is a **factor** of p(x)  addition in GF(qn) just involves summing equivalent terms in the polynomial modulo q (XOR if q=2)  a(x)+b(x)=(an-1+bn-1)xn-1+...+(a1+b1)x+(a0+b0) |
| **Multiplication with Polynomials in GF(qn)** | multiplication in GF(qn) involves  multiplying the two polynomials together (cf longhand multiplication; here use  shifts & XORs if q=2)  then finding the residue modulo a given **irreducible polynomial** of degree n  an **irreducible polynomial** d(x) is a 'prime' polynomial, it has no polynomial divisors other than itself and 1  modulo reduction of p(x) consists of finding some r(x) st: p(x)=q(x)d(x)+r(x)  nb. in GF(2n) with d(x)=x3+x+1 can do simply by replacing x3 with x+1  eg in GF(23) there are 8 elements:  0, 1, x, x+1, x2, x2+1, x2+x, x2+x+1  with irreducible polynomial d(x)=x3+x+1\* arithmetic in this field can be summarised as:  can adapt GCD, Inverse, and CRT algorithms for GF(qn)  [[phi]](p(x)) = 2n-1 since every poly except 0 is relatively prime to p(x)  arithmetic in GF(qn) can be much faster than integer arithmetic, especially if the irreducible polynomial is carefully chosen  eg a fast implementation of GF(2127) exists  has both advantages and disadvantages for cryptography, calculations are faster, as are methods for breaking |
| **RSA and the Chinese Remainder Theorem** | a significant improvement in decryption speed for RSA can be obtained by using the Chinese Remainder theorem to work modulo p and q respectively  o since p,q are only half the size of R=p.q and thus the arithmetic is much faster  CRT is used in RSA by creating two equations from the decryption calculation:  M = Cd mod R  as follows:  M1 = M mod p = (C mod p)d mod (p-1)  M2 = M mod q = (C mod q)d mod (q-1)  then the pair of equations  M = M1 mod p M = M2 mod q  has a unique solution by the CRT, given by:  M = [((M2 +q - M1)u mod q] p + M1  where  p.u mod q = 1 |
| **FINITE FIELDS**  **Groups, Rings and Field** | **Group**: A set of elements that is closed with respect to some operation.  Closed-> The result of the operation is also in the set  The operation obeys:   * 1. Obeys associative law: **(a.b).c = a.(b.c)**   2. Has identity e: **e.a = a.e = a**   3. Has inverses **a-1: a.a-1 = e** |
| **Abelian Group** | The operation is commutative  **a.b = b.a**  Example: Z8, + modular addition, identity =0 |
| **Cyclic Group** | Exponentiation: Repeated application of operator   1. example: **a3 = a.a.a** 2. Cyclic Group: Every element is a power of some fixed element, i.e., **b = ak** for some a and every b in group a is said to be a generator of the group 3. Example: {1, 2, 4, 8} with mod 12 multiplication, the generator is 2. 4. 20=1, 21=2, 22=4, 23=8, 24=4, 25=8 |
| **Ring:** | 1. A group with two operations: addition and multiplication 2. The group is abelian with respect to addition: **a+b=b+a** 3. Multiplication and additions are both associative: **a+(b+c)=(a+b)+c**   **a.(b.c)=(a.b).c**   1. Multiplication distributes over addition, **a.(b+c)=a.b+a.c** 2. Commutative Ring: Multiplication is commutative, i.e., **a.b = b.a** 3. Integral Domain: Multiplication operation has an identity and no zero divisors |
| **Field** | An integral domain in which each element has a multiplicative inverse |
| **Modular Arithmetic** | a **congruence** a = b mod n says when divided by *n* that a and b have the same remainder o 100 = 34 mod 11  o usually have 0<=b<=n-1  o -12mod7 = -5mod7 = 2mod7 = 9mod7 o b is called the **residue** of a mod n  can do arithmetic with integers modulo n with all results between 0 and n |

**Concepts**

|  |  |  |
| --- | --- | --- |
| **Modular Arithmetic** | | modular arithmetic is 'clock arithmetic'  a **congruence** a = b mod n says when divided by *n* that a and b have the same remainder o 100 = 34 mod 11  o usually have 0<=b<=n-1  o -12mod7 = -5mod7 = 2mod7 = 9mod7 o b is called the **residue** of a mod n  can do arithmetic with integers modulo n with all results between 0 and n  **Addition**  **a+b mod n**  **Subtraction**  **a-b mod n = a+(-b) mod n**  **Multiplication**  **a.b mod n**  derived from repeated addition  can get a.b=0 where neither a,b=0 o eg 2.5 mod 10  **Division**  **a/b mod n**  is multiplication by inverse of b: a/b = a.b-1 mod n  if n is prime b-1 mod n exists s.t b.b-1 = 1 mod n   * 1. eg 2.3=1 mod 5 hence 4/2=4.3=2 mod 5   integers modulo n with addition and multiplication form a commutative ring with the laws of  **Associativity** : (a+b)+c = a+(b+c) mod n  **Commutativity** : a+b = b+a mod n  **Distributivity** : (a+b).c = (a.c)+(b.c) mod n  also can chose whether to do an operation and then reduce modulo n, or reduce then do the operation, since reduction is a homomorphism from the ring of integers to the ring of integers modulo n  o a+/-b mod n = [a mod n +/- b mod n] mod n o (the above laws also hold for multiplication)  if n is constrained to be a prime number *p* then this forms a **Galois Field modulo p** denoted **GF(p)** and all the normal laws associated with integer arithmetic work  **Greatest Common Divisor**  the greatest common divisor (a,b) of a and b is the largest number that divides evenly into both a and b    **Finite Fields or Galois Fields**  Finite Field: A field with finite number of elements  Also known as Galois Field  The number of elements is always a power of a prime number. Hence, denoted as GF(pn)  GF(p) is the set of integers {0,1, …, p-1} with arithmetic operations modulo prime p  Can do addition, subtraction, multiplication, and division without leaving the field GF(p)  GF(2) = Mod 2 arithmetic GF(8) = Mod 8 arithmetic  There is no GF(6) since 6 is not a power of a prime  **Polynomial Arithmetic**  ***f*(*x*) = an*x*n+ an-1*x*n-1+ …+ a1*x +* a0= Σ ai*x*i**   1. Ordinary polynomial arithmetic:    1. Add, subtract, multiply, divide polynomials,    2. Find remainders, quotient.    3. Some polynomials have no factors and are prime. 2. Polynomial arithmetic with mod p coefficients 3. Polynomial arithmetic with mod p coefficients and mod m(x) operations   **Polynomial Arithmetic with Mod 2 Coefficients**   1. All coefficients are 0 or 1, e.g.,   let *f*(*x*) = *x*3 + *x*2 and *g*(*x*) = *x*2 + *x* + 1  *f*(*x*) + *g*(*x*) = *x*3 + *x* + 1  *f*(*x*) x *g*(*x*) = *x*5 + *x*2   1. Polynomial Division: *f*(*x*) = *q*(*x*) *g*(*x*) + *r*(*x*) 2. can interpret *r*(*x*) as being a remainder 3. *r*(*x*) = *f*(*x*) mod *g*(*x*) 4. if no remainder, say *g*(*x*) divides *f*(*x*) 5. if *g*(*x*) has no divisors other than itself & 1 say it is irreducible (or prime) polynomial 6. Arithmetic modulo an irreducible polynomial forms a finite field 7. Can use Euclid‟s algorithm to find gcd and inverses. |
| **Public Key Cryptography** | ***Introduction to Public key Cryptography:***   * Public key cryptography also called as **asymmetric cryptography**. * It was invented by whitfield **Diffie** and Martin **Hellman** in 1976. Sometimes this cryptography also called as **Diffie-Helman Encryption**. * Public key algorithms are based on mathematical problems which admit no efficient solution that are inherent in certain integer factorization, discrete logarithm and Elliptic curve relations.   ***Public key Cryptosystem Principles:***   * The concept of public key cryptography in invented for two most difficult problems of Symmetric key encryption.  The Key Exchange ProblemThe Trust ProblemThe Key Exchange Problem: The key exchange problem arises from the fact that communicating parties must somehow share a secret key before any secure communication can be initiated, and both parties must then ensure that the key remains secret. Of course, direct key exchange is not always feasible due to risk, inconvenience, and cost factors.The Trust Problem: Ensuring the integrity of received data and verifying the identity of the source of that data can be very important. Means in the symmetric key cryptography system, receiver doesn’t know whether the message is coming for particular sender.  * This public key cryptosystem uses two keys as pair for encryption of plain text and Decryption of cipher text. * These two keys are names as “**Public key**” and “**Private key**”. The private key is kept secret where as public key is distributed widely. * A message or text data which is encrypted with the public key can be decrypted only with the corresponding private-key * This two key system very useful in the areas of confidentiality (secure) and authentication  |  |  |  | | --- | --- | --- | | A **public-key encryption** scheme has six ingredients | | | | 1 | **Plaintext** | This is the readable message or data that is fed into the algorithm as input. | | 2 | **Encryption algorithm** | The encryption algorithm performs various transformations on the plaintext. | | 3 | **Public 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. The exact transformations performed by the algorithm depend on the public or private key that is provided as input | | 4 | **Private key** | | 5 | **Ciphertext** | This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts. | | 6 | **Decryption algorithm** | This algorithm accepts the ciphertext and the matching key and produces the original plaintext. |   **Public key cryptography for providing confidentiality (secrecy)**    The essential steps are the following.   1. Each user generates a pair of keys to be used for the encryption and decryption of messages. 2. Each user places one of the two keys in a public register or other accessible file. This is the public key.The companion key is kept private.As Figure 9.1a suggests, each user maintains a collection of public keys obtained from others. 3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice’s public key. 4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice’s private key.     There is some source A that produces a message in plaintext ***X* = [*X*1, *X*2, . . . ,X*M*].**  The ***M***elements of ***X***are letters in some finite alphabet. The message is intended for destination **B**.  B generates a related 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 the encryption key ***PUb***as input, A forms the ciphertext *Y* = [*Y*1, *Y*2, . . . , *YN*]:    The intended receiver, in possession of the matching private key, is able to invert the transformation:    **Public key cryptography for proving Authentication:**      The above diagrams show the use of public-key encryption to provide authentication:     * In this case,A prepares a message to B and encrypts it using A’s private key before transmitting it. B can decrypt the message using A’s public key. 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.** * 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.   **Public key cryptography for both authentication and confidentiality (Secrecy)**  It is, however, possible to provide both the authentication function and confidentiality by a double use of the public-key scheme (above figure):    In this case, we begin as before by encrypting a message, using the sender’s private key. This provides the digital signature. Next, we encrypt again, using the receiver’s public key. The final ciphertext can be decrypted only by the intended receiver, who alone has the matching private key. Thus, confidentiality is provided.  **Applications for Public-Key Cryptosystems**  Public-key systems are characterized by the use of a cryptographic algorithm with two keys, one held private and one available publicly. Depending on the application, the sender uses either the sender’s private key or the receiver’s public key, or both, to perform some type of cryptographic    function. the use of **public-key cryptosystems** into three categories  • 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. Several different approaches are possible, involving the private key(s) of one or both parties. | |
| **RSA** | * It is the most common public key algorithm. * This RSA name is get from its inventors first letter (Rivest (R), Shamir (S) and Adleman (A)) in the year 1977. * The RSA scheme is a block cipher in which the plaintext & ciphertext are integers between 0 and n-1 for some ‘n’. * A typical size for ‘n’ is 1024 bits or 309 decimal digits. That is, n is less than 21024   **Description of the Algorithm:**   * RSA algorithm uses an expression with exponentials. * In RSA plaintext is encrypted in blocks, with each block having a binary value less than some number n. that is, the block size must be less than or equal to **log2(n)** * **RSA** uses two exponents ‘e’ and ‘d’ where e🡪public and d🡪private. * Encryption and decryption are of following form, for some PlainText ‘M’ and CipherText block ‘C’     M=Cd mod = (Me mod n) d mon n =(Me)d mod n= Med mod n  Both sender and receiver must know the value of n.  The sender knows the value of ‘e’ & only the reviver knows the value of ‘d’ thus this is a public key encryption algorithm with a  Public key PU={e, n}  Private key PR={d, n}  **Requirements:**  The RSA algorithm to be satisfactory for public key encryption, the following requirements must be met:   1. It is possible to find values of e, d n such that “ **Med mod n =M** ” for all M<n 2. It is relatively easy to calculate “ **Me mod n “** and “ **Cd mod n** “for M<n 3. It is infeasible to determine “d” given ‘e’ & ‘n’. The “ **Med mod n =M** ” relationship holds if ‘e’ & ‘d’ are multiplicative inverses modulo Ø(n).   Ø(n)🡪 Euler Totient function  For p,q primes where p\*q and p≠q.  Ø(n)= Ø(pq)=(p-1)(q-1)  Then the relation between ‘e’ & ‘d’ can be expressed as “ **“**  this is equivalent to saying    That is ‘e’ and ‘d’ are multiplicative inverses mod Ø(n).  Note: according to the rules of modular arithmetic, this is true only if ‘d’ (and ‘e’) is relatively prime to Ø(n).  Equivalently **gcd(Ø(n), d)=1.**  **Steps of RSA algorithm:**  Step 1🡪Select 2 prime numbers p & q  Step 2🡪Calculate n=pq  Step 3🡪Calculate Ø(n)=(p-1)(q-1)  Step 4🡪 Select or find integer e (public key) which is relatively prime to Ø(n).  ie., e with gcd (Ø(n), e)=1 where 1<e< Ø(n).  Step 5🡪 Calculate “d” (private key) by using following condition.  d< Ø(n).  Step 6🡪 Perform encryption by using  Step 7🡪 perform Decryption by using  **Example:**  **1.** Select two prime numbers, *p* = 17 and *q* = 11.  **2.** Calculate *n* = *pq* = 17 × 11 = 187.  **3.** Calculate Ø(*n*) = (*p* - 1)(*q* - 1) = 16 × 10 = 160.  **4.** Select *e* such that *e* is relatively prime to Ø(*n*) = 160 and less than Ø (*n*); we choose *e* = 7.  **5.** Determine *d* such that ***de ≡*1 (mod 160)** and *d* < 160.The correct value is *d* = 23, because 23 \* 7 = 161 = (1 × 160) + 1; *d* can be calculated using the extended Euclid’s algorithm  The resulting keys are public key *PU* = {7, 187} and private key *PR* = {23, 187}.  The example shows the use of these keys for a plaintext input of *M*= 88. For encryption,  we need to calculate *C* = 887 mod 187. Exploiting the properties of modular arithmetic, we can do this as follows.        **The Security of RSA**  Four possible approaches to attacking the RSA algorithm are  • **Brute force:** This involves trying all possible private keys.  • **Mathematical attacks:** There are several approaches, all equivalent in effort to factoring the product of two primes.  • **Timing attacks:** These depend on the running time of the decryption algorithm.  • **Chosen ciphertext attacks:** This type of attack exploits properties of the RSA algorithm. | |
| **Diffie-Hellman Key Exchange:** | * Diffie-Hellman key exchange is the first published public key algorithm * This Diffie-Hellman key exchange protocol is also known as exponential key agreement. And it is based on mathematical principles. * The purpose of the algorithm is to enable two users to exchange a key securely that can then be used for subsequent encryption of messages. * This algorithm itself is limited to exchange of the keys. * This algorithm depends for its effectiveness on the difficulty of computing discrete logarithms. * The discrete logarithms are defined in this algorithm in the way of define a primitive root of a prime number. * Primitive root: we define a primitive root of a prime number P as one whose power generate all the integers form 1 to P-1 that is if ‘a’ is a primitive root of the prime number P, then the numbers   are distinct and consist of the integers form 1 through P-1 in some permutation.  For any integer ‘b’ and ‘a’, here ‘a’ is a primitive root of prime number P, then  **b≡ ai mod P 0 ≤ i ≤ (P-1)**  The exponent i 🡪 is refer as discrete logarithm or index of b for the base a, mod P.  The value denoted as **ind a,p(b)**  **Algorithm for Diffie-Hellman Key Exchange:**  Step 1🡪 two public known numbers q, α  q🡪 Prime number  α🡪 primitive root of q and α< q.  Step 2 🡪 if A & B users wish to exchange a key   1. User A select a random integer XA<q and computes 2. User B independently select a random integer XB <q and computes 3. Each side keeps the X value private and Makes the Y value available publicly to the outer side.   Step 3🡪 User A Computes the key as  User B Computes the key as  Step 4🡪 two calculation produce identical results    (We know that )        (We know that)  The result is that the two sides have exchanged a secret key.    **Example:** | |
| **MAN-in the Middle Attack (MITM)** | **Definition:** A man in the middle attack is a form of eavesdropping where communication between two users is monitored and modified by an unauthorized party.  Generally the attacker actively eavesdrops by intercepting (stoping) a public key message exchange.  The Diffie- Hellman key exchange is insecure against a “Man in the middle attack”.  Suppose user ‘A’ & ‘B’ wish to exchange keys, and D is the adversary (opponent). The attack proceeds as follows.   1. ‘D’ prepares for the attack by generating two random private keys XD1 & XD2 and then computing the corresponding public keys YD1 and YD2. 2. ‘A’ transmits ‘YA’ to ‘B’ 3. ‘D’ intercepts YA and transmits YD1 to ‘B’. and D also calculates 4. ‘B’ receives YD1 & calculate 5. ‘B’ transmits ‘YB’ to ‘A” 6. ‘D’ intercepts ‘YB’ and transmits YD2 to ‘A’ and ‘D’ calculate K1 7. A receives YD2  and calculates   At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key *K*1 and Alice and Darth share secret key *K*2. All future communication between Bob and Alice is compromised in the following way.    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. | |
| **Elliptic Curve Cryptography** | * **Definition: Elliptic curve cryptography (ECC)** is an approach to [public-key cryptography](http://en.wikipedia.org/wiki/Public-key_cryptography) based on the algebraic structure of [elliptic curves](http://en.wikipedia.org/wiki/Elliptic_curve) over [finite fields](http://en.wikipedia.org/wiki/Finite_field). These are analogy of existing public key cryptosystem in which modular arithmetic is replaced by operations defined over elliptic curve. * The use of elliptic curves in cryptography was suggested independently by [Neal Koblitz](http://en.wikipedia.org/wiki/Neal_Koblitz)and [Victor S. Miller](http://en.wikipedia.org/wiki/Victor_S._Miller) in **1985.** * Elliptic curve cryptography (ECC) is one of the most powerful but least understood types of cryptography in wide use today. An increasing number of websites make extensive use of ECC to secure everything from customers' HTTPS connections to how they pass data between data centers.   An elliptic curve is defined by an equation in two variables with coefficients. For  cryptography, the variables and coefficients are restricted to elements in a finite field,  which results in the definition of a finite abelian group.  **Elliptic Curves over Real Numbers**  Elliptic curves are not ellipses. They are so named because they are described by cubic equations,  is similar to equation of calculating circumference of an ellipse.  Where  a,b,c,d and e 🡪 real numbers.  X and Y are🡪taken on values in the real numbers.  For utilization of this in cryptography  🡪 EQ1, is sufficient.  Such equations are said to be cubic, or of degree 3, because the highest exponent they contain is a 3. Also included in the definition of an elliptic curve is a single element denoted *O* and called the *point at infinity* or the *zero point. To plot such a curve, we need to compute*  For given values of and, the plot consists of positive and negative values of for  each value of . Thus, each curve is symmetric about *y* = 0.  Two families of elliptic curves are used in cryptographic applications:   * Prime curves over **Z*p [it is*** *Best for software application****]*** * Binary curves over **GF(2m)  *[it is*** *Best for software application****]***   **Prime curves over Z*p***  In Prime curves over **Z*p , p🡪*** referred to as a modulus.  we use a cubic equation in which the variables and coefficients all take on values in the set of integers from **0** through ***p* - 1** and in which calculations are performed modulo *p*.  from EQ1, in this case coefficients and variables limited to **Z*p.***  🡪 eq2  Now consider the set E*p*(*a*, *b*) consisting of all pairs of integers (*x*, *y*) that satisfy  Equation eq2 together with a point at infinity .The coefficients a and b and the variables x and y are all elements of Zp.    For example, let *p* = 23 and consider the elliptic curve *y*2 = *x*3 + *x* + 1 In this case, *a* = *b* = 1  For the set E23(1, 1), we are only interested in the nonnegative integers in the quadrant from (0, 0) through (*p* - 1, *p* - 1) that satisfy the equation mod *p.*  **Elliptic Curves over GF(2m):**  A finite field GF(2m) consists of 2m elements, together with addition & multiplication operations that can be defined over polynomials.  For elliptic Curves over GF(2m), we use a cubic equation in which the variables and coefficients all take on values in GF(2m), for some number m.  By this, the form of cubic equation appropriate for cryptographic application.  The form is  🡪 EQ3.  To form a cryptographic system using elliptic curves, we need to find a “hard problem” corresponding to factoring the product of two primes or taking the discrete logarithm.  Consider the equation  It is relatively easy to calculate *Q* given *k and P*  *But it is relatively hard to determine given Q* and P.  This is called the discrete logarithm problem for elliptic curves. | |
| **ECC Diffie-Hellman Key Exchange** | ECC can do key exchange, that is analogous to Diffie Hellman.  Key exchange using elliptic curves can be done in the following manner.  First pick a large integer q , which is either a prime number P or an integer of the form 2m  and elliptic curve parameters a & b for equation  or .  This define elliptic group of point Eq(a,b).  Pick a base point G=(x1,y1) in Ep(a,b) whose order is a very large value n.  The order n of a point G on an elliptic curve is the smallest +ve integer n such that nG=0.Eq(a,b) | |
| **Elliptic Curve Encryption/Decryption:** |  | |

**Questions**

1. Explain public key encryption scheme
2. Perform encryption and decryption using RSA algorithm for p=3,q=11,e=7 and M=5.
3. Explain public key cryptosystem for secrecy and authentication.
4. Explain Deffie-Hellman key exchange.
5. What are the principal elements of a public key cryptosystem? Explain.
6. In a public key system using RSA, you intercept the cipher text C=10 sent to a user whose public key is e=5, n= 35. What is the plain text M?
7. Explain RSA algorithm in detail.
8. Perform encryption and decryption using RSA algorithm for p=5,q=11,e=3 and M=9.
9. Describe in general terms an efficient procedure for picking a prime number
10. What requirements must a public key cryptosystems fulfill to be a secure algorithm?
11. Explain RSA algorithm in detail with an example.
12. Compare and contrast different secure hash functions.

**UNIT – III**

**Terminology**

|  |  |
| --- | --- |
| **Disclosure** | Release of message contents to any person or process not possessing the appropriate cryptographic key. |
| **Traffic analysis** | Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined. |
| **Masquerade** | Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent acknowledgments of message receipt or nonreceipt by someone other than the message recipient. |
| **Content modification** | Changes to the contents of a message, including insertion, deletion, transposition, and modification. |
| **Sequence modification** | Any modification to a sequence of messages between parties, including insertion, deletion, and reordering. |
| **Timing modification** | Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., datagram) could be delayed or replayed. |

**Concepts**

|  |  |
| --- | --- |
| **MESSAGE AUTHENTICA-**  **TION**  **FUNCTIONS** | Any message authentication or digital signature mechanism has two levels of functionality. At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message. This lower-level function is then used as a primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.  Authentication Functions that may be used for produce an authenticator. These may be grouped into three classes.  • **Message encryption:** The ciphertext of the entire message serves as its authenticator  • **Message authentication code (MAC):** A function of the message and a secret key that produces a fixed-length value that serves as the authenticator  • **Hash function:** A function that maps a message of any length into a fixed length hash value, which serves as the authenticator.  **Message Encryption**  Message encryption by itself can provide a measure of authentication. The analysis differs for symmetric and public-key encryption schemes.    ***SYMMETRIC ENCRYPTION*** Consider the straightforward use of symmetric encryption (Figure 12.1a). A message M transmitted from source A to destination B is encrypted using a secret key K shared by A and B. If no other party knows the key, then confidentiality is provided: No other party can recover the plaintext of the message.  B is assured that the message was generated by A. Why? The message must have come from A, because A is the only other party that possesses K and therefore the only other party with the information necessary to construct ciphertext that can be decrypted with K. 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 ciphertext to produce the desired changes in the plaintext. So we may say that symmetric encryption provides authentication as well as confidentiality  Given a decryption function D and a secret key K, the destination will accept any input X and produce output Y=D(X, K). If X is the ciphertext of a legitimate message produced by the corresponding encryption function, then Y is some plaintext message. Otherwise, will likely be a  meaningless sequence of bits. There may need to be some automated means of determining at B whether is legitimate plaintext and therefore must have come from A.  If incoming cipher text decrypts to intelligible plaintext. If the plaintext is, say, a binary object file or digitized X-rays, determination of properly formed and therefore authentic plaintext may be difficult.  Thus, an opponent could achieve a certain level of disruption simply by issuing messages with random content purporting to come from a legitimate user.  One solution to this problem is to force the plaintext to have some structure that is easily recognized but that cannot be replicated without recourse to the encryption function.We could, for example, append an error-detecting code, also known as a frame check sequence (FCS) or checksum, to each message before encryption, as illustrated in Figure 12.2a.    **Internal error control**  A prepares a plaintext message and then provides this as input to a function F that produces an FCS. The FCS is appended to M and the entire block is then encrypted. At the destination, B decrypts the incoming block and treats the results as a message with an appended FCS.  B applies the same function F to attempt to reproduce the FCS. If the calculated FCS is equal to the incoming FCS, then the message is considered authentic. It is unlikely that any random sequence of bits would exhibit the desired relationship. The sequence illustrated in Figure 12.2a is referred as **internal error control.**  **External error control**  Figure 12.2 b, with internal error control, authentication is provided because an opponent would have difficulty generating ciphertext that, when decrypted, would have valid error control bits. If instead the FCS is the outer code, an opponent can construct messages with valid error-control codes. Although the opponent cannot know what the decrypted plaintext will be, he or she can still hope to create confusion and disrupt operations. |
| **MESSAGE AUTHENTICA-**  **TION CODE**  **(MAC)** | **Definition:** In cryptography, a message authentication Code (MAC) is a short piece of information used to authenticate a message and to provide integrity and authenticity (valid) assurances (free form doubt) on the message.  Integrity assurance detects accidental and internal message changes.  Authenticity assurances affirm (swear) the messages origin   * MAC, also known as a cryptographic checksum. * It is an alternative authentication technique involves the use of a secret key to generate a small fixed size block of data, that is appended to the message. * A MAC or cryptographic checksum, is generated by a function C of the form   **MAC=C (K, M)**  M🡪Input Message  C🡪MAC function  K🡪 Shared Secret Key of Communication parties (A & B).  MAC🡪 Message Authentication Code.  When A has a message to send to B, The “message + MAC” are transmitted to the intended (aimed) recipient (B).  The recipient perform 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.  NOTE: A MAC function is similar to encryption; one difference is that the MAC algorithm need not be reversible, as it must for decryption.  MAC function is a many-to-one, potentially many messages have same MAC. That is, if message M is 100 bit message, and 10 bit MAC then there are 2100 messages and 210 MAC are available.  There four 2100/210 = 290 different message, 5 bit key used then 25=32 different mapping form the set of messages to the set of MAC values.  In general n-bit MAC is used, then there are 2n possible MAC’s  N🡪possible messages with N>>2n With K-bits key, there 2K possible keys.  **Situations in which a message authentication code is used**   1. There are a number of applications in which the same message is broadcast 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. It is cheaper and more reliable to have only one destination responsible for monitoring authenticity. Thus, the message must be broadcast 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.     1. Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages. Authentication is carried out on a selective basis, messages being chosen at random for checking. 2. Authentication of a computer program in plaintext is an attractive service. The computer program can be executed without having to decrypt it every time, which would be wasteful of processor resources. However, if a message authentication code were attached to the program, it could be checked whenever assurance was required of the integrity of the program. 3. For some applications, it may not be of concern to keep messages secret, but it is important to authenticate messages.   An **example** is the Simple Network Management Protocol Version 3 (SNMPv3), which separates the functions of confidentiality and authentication. For this application, it is usually important for a managed system to authenticate incoming SNMP messages, particularly if the message contains a command to change parameters at the managed system. On the other hand, it may not be necessary to conceal the SNMP traffic.   1. Separation of authentication and confidentiality functions affords architectural flexibility. For example, it may be desired to perform authentication at the application level but to provide confidentiality at a lower level, such as the transport layer. 2. A user may wish to prolong the period of protection beyond the time of reception and yet allow processing of message contents. With message encryption, the protection is lost when the message is decrypted, so the message is protected against fraudulent modifications only in transit but not within the target system.   MAC does not provide a digital signature, because both sender and receiver share the same key.  **HASH FUNCTION:**  It is a one of the authentication function; it accepts a variable size message M as input and produce a fixed size output.  A hash value ‘h’ is generated by a function H of the form  **h=H (M)**  M🡪 variable length message  H(M)🡪 fixed length hash value.  The hash code is also referred as Message Digest (MD) or hash value.  The main difference between Hash Function and MAC is , a hash code does not use a key but is a function only of the input message.  The hash value 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 hash value.  that message by re-computing the hash value.   1. 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. Because encryption is applied to the entire message plus hash code, confidentiality is also provided. 2. Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality. 3. It is possible to use a hash function but no encryption for message authentication. The technique assumes that the 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. Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.      1. It is possible to use a hash function but no encryption for message authentication. The technique assumes that the 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. Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message. 2. Confidentiality can be added to the approach of method (c) by encrypting the entire message plus the hash code.   When confidentiality is not required, method (b) has an advantage over methods (a) and (d), which encrypts the entire message, in that less computation is required. Nevertheless, there has been growing interest in techniques that avoid encryption (Figure 11.2c). Because   * Encryption software is relatively slow. Even though the amount of data to be encrypted per message is small, there may be a steady stream of messages into and out of a system. * Encryption hardware costs are not negligible. Low-cost chip implementations of DES are available, but the cost adds up if all nodes in a network must have this capability. * Encryption hardware is optimized toward large data sizes. For small blocks of data, a high proportion of the time is spent in initialization/invocation overhead. * Encryption algorithms may be covered by patents, and there is a cost associated with licensing their use.   **Requirements for a hash function**:  The purpose of a hash function is to produce a “fingerprint” of a file, message or other block of data. To be useful for message authentication, a hash function H must have the following properties:   1. H van 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. 4. One-way property:- for any given value h, it is computationally infeasible to find x such that H(x)=h. this sometimes referred to in the literature as the one way property. 5. Weak collision resistance:- for any given block x. it is computationally infeasible to find y≠x with H(y)=H(x). this is referred as weak collision resistance. 6. Strong collision resistance:- it is computationally infeasible to find any pair (X,Y) such that H(x)=H(y). this referred as strong collision resistance.     **Simple Hash functions:**  All hash functions operate using the following general principles. The input (message, file, etc.) is viewed as a sequence of -bit blocks. The input is processed one block at a time in an iterative fashion to produce an -bit hash function. One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block.  This can be expressed as    **Birthday attacks:**  Birthday attacks are a class of brute force techniques used in an attempt to solve a class of cryptographic hash function problem. These methods takes advantage of functions which, when supplied with a random input, return one of k equally likely values.  Suppose that a 64 bit hash code is use, if an encryption hash code C is transmitted with the corresponding unencrypted message M then an opponent would need to find an M1 such that H(M1)=H(M) to substitute another message and fool the receiver. On average, the opponent would have to try about 263 messages to find one that matches the hash code of the intercepted message.  **Hash Algorithms :**   1. Message Digest:MD5 2. Secure Hash Algorithm: SHA-1 (from MD4) 3. RIPEMD-160 4. HMAC   **MD5 Message Digest Algorithm:**  Introduction:-  The MD% message digest algorithm was developed by “Ron Rivest” at MIT (Massachusetts Institute of Technology 1861). It was developed to avoid brute force & crypt-analytic attacks. MD5 was the most widely used secure hash algorithm.  MD5 logic:  Input:-This algorithm takes as input a message of arbitrary length.  Output:- produce a 128 bit message digest.  The input is processed in 512 bit blocks.  **Algorithm processing Steps:**  Step1: Append Padding Bits  Step 2: Append Length  Step 3: Initialize MD Buffer  Step 4: Process Message in 512 bit (16-Word) Blocks  Step 5: Output    Step-1: **Appending Padding Bits.** The original message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. The padding rules are:   * The original message is always padded with one bit "1" first. * Then zero or more bits "0" are padded to bring the length of the message up to 64 bits fewer than a multiple of 512.   Step-2: **Appending Length**. 64 bits are appended to the end of the padded message to indicate the length of the original message in bytes. The rules of appending length are:   * The length of the original message in bytes is converted to its binary format of 64 bits. If overflow happens, only the low-order 64 bits are used. * Break the 64-bit length into 2 words (32 bits each). * The low-order word is appended first and followed by the high-order word.   Step-3: **Initializing MD Buffer**. A 128 bit buffer is used to hold intermediate and final results of the ash function. The buffer can be represented as four 32 bit registers (A, B, C, D). these registers are initialize to the following 32 bit integers (hexadecimal values)   * Word A is initialized to: 0x67452301. * Word B is initialized to: 0xEFCDAB89. * Word C is initialized to: 0x98BADCFE. * Word D is initialized to: 0x10325476.   Step-4: Processing Message in 512-bit Blocks. This is the main step of MD 5 algorithm, which loops through the padded and appended message in blocks of 512 bits each. For each input block, 4 rounds of operations are performed with 16 operations in each round. The four rounds have a similar structure, but each uses a different primitive logical function, referred to as F,G,H and I in the specification.  Step-5: **output:** After all L 512 bit blocks have been processed, the output form the Lth stage is the 128 bit message digest. |
| **Digital Signature** | **Definition:** A digital signature or digital signature scheme is a mathematical scheme for demonstration the authenticity of digital message or document.  Means, a digital signature is an authentication mechanism that enables the creator of a message to attach a code that act as a signature.  This 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.  The digital signature standard (DSS) is an NIST standard that uses the secure hash algorithm (SHA).  **Where it used:**  Message authentication protects two parties who exchange message from any third party. But it does not protect the two parties against each other.  Example: Suppose that john sends an authenticated message to marry, using any authentication scheme (symmetric or public key cryptography). There are two disputes (clash or fight or arguments) that could arise.   1. Mary may forge a different message & claim that it came from John. Means, Mary would simply have to create a message & append an authentication code using the key, which John and Mary share. 2. John can deny sending the message Because it is possible for mary to forge that john did n fact send the message.   **Properties of Digital Signature:**  • It must verify the author and the date and time of the signature.  • It must authenticate the contents at the time of the signature.  • It must be verifiable by third parties, to resolve disputes.  **Digital Signature Requirements**   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.   **Approaches for Digital Signature:**   * Direct Digital Signature * Arbitrated Digital Signature   **Direct Digital Signature**:  The term **direct digital signature** refers to a digital signature scheme that involves only the communicating parties (source, destination).  The validly of scheme depends on the security of the sender’s private key.  The sender later wishes to deny sending a particular message by claiming the private key was lost or stolen or some other reason.  There is chance in stole the private key of a sender at some time T.  **Arbitrated Digital Signature:**  In this every signed message from a sender X to a receiver Y goes first to an arbiter A, who subjects the message and its signature to a number of tests to check it origin and content. The message is then dated and sent to Y.  This process is an indication that has been verified to the satisfaction of the arbiter.  By this process, it solves the direct Digital signature problem.    Sender X,  Arbiter A,  Receiver Y,   * X🡪 construct message M and compute hash value H(M) then X transmitted “M+ Digital Signature” to A.   Signature consists🡪 identity “IDx of X +hash value” of all encrypted using KXA (it is common shared key between Sender X and Arbiter A).   * A🡪 A decrypts the signature & checks the hash value to validate the message. Then transmit it to Y by encryption it with KAY (it is common shared key between Arbiter A and Receiver Y). the message include IDx and M & time Stam. * Y🡪 Decrypt it by using KAY   X🡪A: M||EKXA[IDx||H(M)]  A🡪Y: EKAY[IDx||M|| EKXA[IDx||H(M)]||T] |
| **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 Secure Hash Algorithm (SHA) presents a new digital signature technique, the Digital Signature Algorithm (DSA).  **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.  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 generated for this particular signature. The signature function also depends on the sender’s private key (*PRa*) and a set of parameters known to a group of communicating principals. We can consider this set to constitute a global public key (PUG). |

**Questions**

1. What are the requirements of a Hash function
2. Explain SHA-1 algorithm.
3. Explain MD5 in detail
4. Explain Simple Hash Function.
5. What is a MAC? Explain message authentication using MAC
6. Explain RIPEMD-160 in detail
7. Compare the principal characteristics of MD5,SHA-1,and RIPEMD-160
8. What is a digital signature? Explain this using public key algorithm
9. Explain the HMAC algorithm in detail.
10. Explain the Digital Signature Algorithm.
11. Write about MAC functions?
12. Compare and contrast SHA-1 and HMAC functions
13. Define digital signature? Explain its role in network security.
14. Explain about SHA-512.
15. What is one-way hash? With an example, describe one-way hash function.
16. Explain the procedure involved in SHA-1 algorithm.

**UNIT – IV**

**Terminology**

|  |  |
| --- | --- |
| **Kerberos** | Def: Introduction: Kerberos is a computer network authentication protocol which works on the basis of “tickets” to allow nodes communication over a non secure network.  (Or) It is a secure method (service) for authenticate a request for a service in a computer network.  It was developed in “Athena Project” at MIT.  It provide authentication by using Secret-Key cryptography. |
| **Use of Kerberos** | Kerberos is used for decreasing the burden for server, means; Kerberos will takes responsibility of authentication.  It is designed for providing for strong authentication for client/server applications by using secret-key.   * **Versions:**   Kerberos Version4  Kerberos Version5 |
| **Characteristics of KERBEROS** | * It is secure: it never sends a password unless it is encrypted. * Only a single login is required per session. Credentials defined at login are then passed between resources without the need for additional logins. * The concept depends on a trusted third party – a Key Distribution Center (KDC). The KDC is aware of all systems in the network and is trusted by all of them. * It performs mutual authentication, where a client proves its identity to a server and a server proves its identity to the client. |
| **Requirements Kerberos** | * **Secure**: Kerberos should be strong enough that a potential opponent does not find it to be the weak link. * **Reliable:** 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. |

**Concepts**

|  |  |
| --- | --- |
| **Key Management and Distribution** | 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.  Key distribution often involves the use of master keys, which are infrequently used and are long lasting, and session keys, which are generated and distributed for temporary use between two parties.  **SYMMETRIC KEY DISTRIBUTION USING SYMMETRIC ENCRYPTION**  In symmetric encryption two parties to an exchange must share the same key, and that key must be protected from access by others.  The *key distribution technique* is 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.   Physical delivery (1 & 2) is simplest - but only applicable when there is personal contact between recipient and key issuer. This is fine for link encryption where devices & keys occur in pairs, but does not scale as number of parties who wish to communicate grows. 3 is mostly based on 1 or 2 occurring first.  A third party, whom all parties trust, can be used as a trusted intermediary to mediate the establishment of secure communications between them (4) Must trust intermediary not to abuse the knowledge of all session keys. As number of parties grows, some variant of 4 is only practical solution to the huge growth in number of keys potentially needed.  **Key distribution centre:**   * 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 and then discarded * **Master key** is shared by the key distribution center and an end system or user and used to encrypt the session key.   **Key Distribution Scenario:**  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 shares the 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, a counter, or a random number; the minimum requirement is that it differs with each request. Also, to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a 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 it originated at the KDC. The message includes two items intended for A:    1. The one-time session key, Ks, to be used for the session    2. 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:   * The one-time session key, Ks to be used for the session * An identifier of A (e.g., its network address), IDA   These last two items are encrypted with Kb (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.   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, namely, E(Kb, [Ks || IDA]). Because this information is encrypted with Kb, it is protected from eavesdropping. B now knows the session key (Ks), knows that the other party is A (from IDA), 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:   1. Using the newly minted session key for encryption, B sends a nonce, N2, to A. 2. Also using Ks, A responds with f(N2), where f is a function that performs some transformation on N2 (e.g., adding one).   **SYMMETRIC KEY DISTRIBUTION USING ASYMMETRIC**  **ENCRYPTION**   * Once public keys have been distributed or have become accessible, secure communication that thwarts eavesdropping, tampering, or both, is possible. * Public-key encryption provides for the distribution of secret keys to be used for conventional encryption.   **Simple Secret Key Distribution**   * A generates a public/private key pair {PUa, PRa} and transmits a message to B consisting of PUa and an identifier of A, IDA * B generates a secret key, Ks, and transmits it to A, encrypted with A's public key. * A computes D(PRa, E(PUa, Ks)) to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of Ks. * A discards PUa and PRa and B discards PUa.     Here third party can intercept messages and then either relay the intercepted message or substitute another message Such an attack is known as a **man-in-the-middle attack.**  **Secret Key Distribution with Confidentiality and Authentication:**     * A uses B's public key to encrypt a message to B containing an identifier of A (IDA) and a nonce (N1), which is used to identify this transaction uniquely * B sends a message to A encrypted with PUa and containing A's nonce (N1) as well as a new nonce generated by B (N2) Because only B could have decrypted message (1), the presence of N1 in message (2) assures A that the correspondent is B * A returns N2 encrypted using B's public key, to assure B that its correspondent is A. * A selects a secret key Ks and sends M = E(PUb, E(PRa, Ks)) 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. * B computes D(PUa, D(PRb, M)) to recover the secret key.   **Publicly Available Directory**   * can obtain greater security by registering keys with a public directory * directory must be trusted with properties:   + The authority maintains a directory with a {name, public key} entry for each participant.   + Each participant registers a public key with the directory authority.   + A participant may replace the existing key with a new one at any time because the corresponding private key has been compromised in some way.   + 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 an adversary succeeds in obtaining or computing the private key of the directory authority, the adversary could authoritatively pass out counterfeit public keys and subsequently impersonate any participant and eavesdrop on messages sent to any participant.  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. * It requires users to know the public key for the directory, and that they interact with directory in real-time to obtain any desired public key securely. * Totally seven messages are required.      1. A sends a time stamped 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, PRauth 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:    1. B's public key, PUb which A can use to encrypt messages destined for B    2. The original request, 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.    3. The original timestamp, 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 (IDA) and a nonce (N1), which is used to identify this transaction uniquely. 4. B retrieves A's public key from the authority in the same manner as A retrieved B's public key. 5. At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable: 6. B sends a message to A encrypted with PUa and containing A's nonce (N1) as well as a new nonce generated by B (N2) Because only B could have decrypted message (3), the presence of N1 in message (6) assures A that the correspondent is B. 7. A returns N2, encrypted using B's public key, to assure B that its correspondent is A. |
| **X.509 CERTIFICATES** | **Introduction:**   * In cryptography, X.509 is a TU-T (International Telecommunication Union-Telecommunication) standard for proving authentication directory service. * The directory is a server or distributed set of servers that maintains a database of information about users. * This information includes a mapping from users name to a network address, as well as other attributes and information about the users. * X.509 defines alternative authentication protocols based on the use of public key certificates. * This was initial used in 1988.   **User of X.509:**  X.509 is an important standard because the certificate structure and authentication protocols defined in X.509 are used in a variety context.  Example: this format is used in   * S/MIME * IP Security * SSL (Secure Socket Layer) * TLS (Transport Layer Security) * SET (Secure Electronic Transaction)   X.509 is based on the use of public key cryptography & Digital Signature.  **Certificates:**  In X.509 Scheme, the public key certificate associated with each user.  The user’s certificates created by some trusted Certificate Authority (CA) and placed in the directory by the CA or by the user.  The directory server itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.  The standard uses the notation for a certificate of:  CA<<A>> where the CA signs the certificate for user A with its private key. In more detail CA<<A>> = CA {V, SN, AI, CA, UCA, A, UA, Ap, TA}.  If the corresponding public key is known to a user, then that user can verify that a certificate signed by the CA is valid.   * **Version:** Differentiates among successive versions of the certificate format; the default is version 1. If the issuer unique identifier or subject unique identifier are present, the value must be version 2. If one or more extensions are present, the version must be version 3. * **Serial number:** An integer value unique within the issuing CA that is unambiguously associated with this certificate. * **Signature algorithm identifier**: The algorithm used to sign the certificate together with any associated parameters. Because this information is repeated in the signature field at the end of the certificate, this field has little, if any, utility. * **Issuer name:** X.500 is the 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. That is, this certificate certifies the public key of the subject who holds the corresponding private key. * **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. Extensions were added in version 3 and are discussed later in this section. * **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.   **Certificate Revocation:**  A certificate includes a period of validity. Typically a new certificate is issued just before the expiration of the old one. In addition, it may be desirable on occasion to revoke a certificate before it expires, for one of a range of following reasons:   1. The user’s private key is assumed to be compromised. 2. The user is no longer certified by this CA. Reasons for this include that the subject’s name has changed, the certificate is superseded, or the certificate was not issued in conformance with the CA’s policies. 3. The CA’s certificate is assumed to be compromised.   To support this, each CA must maintain a list consisting of all revoked but not expired certificates issued by that CA, known as the certificate revocation list (CRL). 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. Each entry consists of the serial number of a certificate and revocation date for that certificate. Because serial numbers are unique within a CA, the serial number is sufficient to identify the certificate.  When a user receives a certificate in a message, the user must determine whether the certificate has been revoked, by checking the directory CRL each time a certificate is received, this often does not happen in practice.  **X.509 Version 3**  The X.509 version 2 format does not convey all of the information. Rather than continue to add fields to a fixed format, standards developers felt that a more flexible approach was needed. X.509 version 3 includes a number of optional extensions that may be added to the version 2 format. Each extension consists of an extension identifier, a criticality indicator, and an extension value. The criticality indicator indicates whether an extension can be safely ignored or not.  **Certificate Extensions**  The certificate extensions fall into three main categories:   * **Key and policy information** - convey additional information about the subject and issuer keys, plus indicators of certificate policy. A certificate policy is a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements. * **Subject and issuer attributes** - support alternative names, in alternative formats, for a certificate subject or certificate issuer and can convey additional information about the certificate subject; eg. postal address, email address, or picture image * **Certification path constraints** - allow constraint specifications to be included in certificates issued for CA’s by other CA’s that may restrict the types of certificates that can be issued by the subject CA or that may occur subsequently in a certification chain. |
| **Electronic Mail Security** | The protection of [email](http://en.wikipedia.org/wiki/Email) from unauthorized access and inspection is known as **electronic privacy**. There are mainly two methods for proving security for electronic mails   * Pretty Good Privacy * S/MIME   **Pretty Good Privacy:**  In virtually all distributed environments, electronic mail is the most heavily used network based application.  Introduction:  PGP is data encryption and decryption computer program that provides privacy (Confidentiality) and authentication for data communication.  It was created by Phil Zimmermann in 1991  Use of PGP:  It is used in Electronic mail  File storage applications.  PGP is an open-source, freely available software package for e-mail security. It provides authentication through the use of digital signature, confidentiality through the use of symmetric block encryption, compression using the ZIP algorithm, and e-mail compatibility using the radix-64 encoding scheme.  PGP incorporates tools for developing a public-key trust model and public-key certificate management  PGP has grown explosively and is now widely used, because of following reasons:   1. It is available free worldwide in versions that run on a variety of platforms, including Windows, UNIX, Macintosh, and many more. In addition, the commercial version satisfies users who want a product that comes with vendor support. 2. It is based on algorithms that have survived extensive public review and are considered extremely secure. Specifically, the package includes RSA, DSS, and Diffie-Hellman for public-key encryption; CAST-128, IDEA, and 3DES for symmetric encryption; and SHA-1 for hash coding. 3. It has a wide range of applicability, from corporations that wish to select and enforce a standardized scheme for encrypting files and messages to individuals who wish to communicate securely with others worldwide over the Internet and other networks. 4. It was not developed by, nor is it controlled by, any governmental or standards organization. For those with an instinctive distrust of “the establishment,” this makes PGP attractive. 5. PGP is now on an Internet standards track (RFC 3156; MIME Security with Open PGP). Nevertheless, PGP still has an aura of an antiestablishment endeavour.   Providing authentication by using PGP:  The sequence steps for providing authentication by using PGP   1. The sender creates a message. 2. SHA-1 is used to generate a 160-bit hash code of the message. 3. The hash code is encrypted with RSA using the sender’s private key, and the result is prepended to the message. 4. The receiver uses RSA with the sender’s public key to decrypt and recover the hash code. 5. The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic.         Steps for providing confidentiality:   1. The sender generates a message and a random 128-bit number to be used as a session key for this message only. 2. The message is encrypted using CAST-128 (or IDEA or 3DES) with the session key. 3. The session key is encrypted with RSA using the recipient’s public key and is prepended to the message. 4. The receiver uses RSA with its private key to decrypt and recover the session key. 5. The session key is used to decrypt the message. |

**Questions**

1. With the help of neat diagram, explain Kerberos action
2. Explain the following PGP services.
3. i) Authentication ii) Confidentiality
4. Explain X.509 Authentication Services
5. Describe S/MIME.
6. Explain Kerberos version 4 message exchange
7. Briefly explain the PGP services
8. What is Kerberos? What are the requirements of a Kerberos
9. Explain the limitations of SMTP/822 scheme.
10. Explain briefly the services provided by PGP?
11. Write a short note on the functionality of S/MIME.
12. Distinguish between Kerberos version 4 and version 5?
13. Describe the X.509 version 3 in details?
14. Briefly describe public key management in PGP
15. What are the requirements for the use of a public-key certificate scheme?
16. What problem was Kerberos designed to address? Briefly describe Kerberos system.
17. What is CA? Briefly describe Public-key certificates.

**UNIT – V**

**Terminology**

|  |  |
| --- | --- |
| **IPSECURITY** | **:** IPSec is a protocol suit for securing internet protocol (IP) communications by authenticating and encrypting each IP packet by authenticating and encrypting each packet of communication session.  It added to either current version of the IP (i.e., IPV4 or IPV6), by means of additional header. |
| **Firewall** | Firewall is software or hardware based network security system that controls the incoming and outgoing network traffic, based on applied rule set.  A fire wall establishes a barrier between a trusted secure internal network and another network. |
| **Encryption**  **Algorithm** | A set of documents that describe how various encryption algorithms are used for ESP. |
| **Domain of Interpretation**  **(DOI)** | Contains values needed for the other documents to relate to each other. These include identifiers for approved encryption and authentication algorithms, as well as operational parameters such as key lifetime. |

**Concepts**

|  |  |
| --- | --- |
| **Applications of IPsec** | * **Secure branch office connectivity over the Internet:** A company can build a secure virtual private network over the Internet or over a public WAN. This enables a business to rely heavily on the Internet and reduce its need for private networks, saving costs and network management overhead. * **Secure remote access over the Internet:** An end user whose system is equipped with IP security protocols can make a local call to an Internet Service Provider (ISP) and gain secure access to a company network. This reduces the cost of toll charges for travelling employees and telecommuters. * **Establishing extranet and intranet connectivity with partners:** IPsec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism. * **Enhancing electronic commerce security:** Even though some Web and electronic commerce applications have built-in security protocols, the use of IPsec enhances that security. IPsec guarantees that all traffic designated by the network administrator is both encrypted and authenticated, adding an additional layer of security to whatever is provided at the application layer. |
| **Benefits of IPsec** | * When IPsec is implemented in a firewall or router, it provides strong security that can be applied to all traffic crossing the perimeter. Traffic within a company or workgroup does not incur the overhead of security-related processing. * IPsec in a firewall is resistant to bypass if all traffic from the outside must use IP and the firewall is the only means of entrance from the Internet into the organization. * IPsec is below the transport layer (TCP, UDP) and so is transparent to applications. There is no need to change software on a user or server system when IPsec is implemented in the firewall or router. Even if IPsec is implemented in end systems, upper-layer software, including applications, is not affected. * IPsec can be transparent to end users. There is no need to train users on security mechanisms, issue keying material on a per-user basis, or revoke keying material when users leave the organization. * IPsec can provide security for individual users if needed. This is useful for offsite workers and for setting up a secure virtual sub-network within an organization for sensitive applications. |
| **IP Sec OverView** | * **Architecture**: Covers the general concepts, security requirements, definitions, and mechanisms defining IPSec technology * **Encapsulating Security Payload (ESP):** Covers the packet format and general issues related to the use of the ESP for packet encryption and, optionally, authentication. * **Authentication Header (AH):** Covers the packet format and general issues related to the use of AH for packet authentication. * **Encryption Algorithm**: A set of documents that describe how various encryption algorithms are used for ESP. * **Authentication Algorithm:** A set of documents that describe how various authentication algorithms are used for AH and for the authentication option of ESP. * **Key Management**: Documents that describe key management schemes. * **Domain of Interpretation (DOI):** Contains values needed for the other documents to relate to each other. These include identifiers for approved encryption and authentication algorithms, as well as operational parameters such as key lifetime. |
| **IPSec Services** | IPSec architecture makes use of two major protocols (i.e., Authentication Header and ESP protocols) for providing security at IP level. This facilitates the system to beforehand choose an algorithm to be implemented, security protocols needed and any cryptographic keys required to provide requested services. The IPSec services are as follows:   * **Connectionless Integrity:-** Data integrity service is provided by IPSec via AH which prevents the data from being altered during transmission. * **Data Origin Authentication**:- This IPSec service prevents the occurrence of replay attacks, address spoofing etc., which can be fatal * **Access Control**:- The cryptographic keys are distributed and the traffic flow is controlled in both AH and ESP protocols, which is done to accomplish access control over the data transmission. * **Confidentiality**:- Confidentiality on the data packet is obtained by using an encryption technique in which all the data packets are transformed into cipher-text packets which are unreadable and difficult to understand. * **Limited Traffic Flow Confidentiality**:- This facility or service provided by IPSec ensures that the confidentiality is maintained on the number of packets transferred or received. This can be done using padding in ESP. * **Replay packets Rejection**:- The duplicate or replay packets are identified and discarded using the sequence number field in both AH and ESP. |
| **Security Associations** | Since IPSEC is designed to be able to use various security protocols, it uses Security Associations (SA) to specify the protocols to be used. SA is a database record which specifies security parameters controlling security operations. They are referenced by the sending host and established by the receiving host. An index parameter called the Security Parameters Index (SPI) is used. SAs are in one direction only and a second SA must be established for the transmission to be bi-directional. A security association is uniquely identified by three parameters:   1. **Security Parameters Index (SPI):** A bit string assigned to this SA and having local significance only. The SPI is carried in AH and ESP headers to enable the receiving system to select the SA under which a received packet will be processed. 2. **IP Destination Address**: Currently, only unicast addresses are allowed; this is the address of the destination endpoint of the SA, which may be an end user system or a network system such as a firewall or router. 3. **Security Protocol Identifier**: This indicates whether the association is an AH or ESP security association. |
| **SA Parameters** | In each IPSec implementation, there is a nominal Security Association Database that defines the parameters associated with each SA. A security association is normally defined by the following parameters:   * **Sequence Number Counter:** A 32-bit value used to generate the Sequence Number field in AH or ESP headers * **Sequence Counter Overflow:** A flag indicating whether overflow of the Sequence Number Counter should generate an auditable event and prevent further transmission of packets on this SA (required for all implementations). * **Anti-Replay Window:** Used to determine whether an inbound AH or ESP packet is a replay * **AH Information:** Authentication algorithm, keys, key lifetimes, and related parameters being used with AH (required for AH implementations). * ESP Information: Encryption and authentication algorithm, keys, initialization values, key lifetimes, and related parameters being used with ESP (required for ESP implementations). * **Lifetime of This Security Association:** A time interval or byte count after which an SA must be replaced with a new SA (and new SPI) or terminated, plus an indication of which of these actions should occur (required for all implementations). * **IPSec Protocol Mode:** Tunnel, transport, or wildcard (required for all implementations). * **Path MTU:** Any observed path maximum transmission unit (maximum size of a packet that can be transmitted without fragmentation) and aging variables (required for all implementations). |
| **Transport and Tunnel Modes** | **Tunnel mode** is most commonly used between gateways, or at an end-station to a gateway, the gateway acting as a proxy for the hosts behind it.  Tunnel mode protects the internal routing information by encrypting the IP header of the original packet. The original packet is encapsulated by another set of IP headers.  **Transport mode** is used between end-stations or between an end-station and a gateway, if the gateway is being treated as a host—for example, an encrypted Telnet session from a workstation to a router, in which the router is the actual destination. The transport mode encrypts only the payload and ESP trailer; so the IP header of the original packet is not encrypted.   |  |  |  | | --- | --- | --- | |  | **Transport Mode SA** | **Tunnel Mode SA** | | **AH** | **Authenticates** IP payload and selected portions of IP header and IPv6 extension headers | **Authenticates** entire inner IP packet plus selected portions of outer IP header | | **ESP** | **Encrypts** IP payload and any IPv6 extension header | **Encrypts** inner IP packet | | **ESP with authentication** | **Encrypts** IP payload and any IPv6 extension header. **Authenticates** IP payload but no IP header | **Encrypts** inner IP packet.  **Authenticates** inner IP packet. |   IP sec can be used (both AH packets and ESP packets) in two modes   * **Transport mode:** the IP sec header is inserted just after the IP header –this contains the security information, such as SA identifier, encryption, authentication   + Typically used in end-to-end communication   + IP header not protected * Tunnel mode: the entire IP packet, header and all, is encapsulated in the body of a new IP packet with a completely new IP header.   + Typically used in firewall-to-firewall communication   + Provides protection for the whole IP packet   + No routers along the way will be able (and will not need) to check the content of the packets |
| **Authentication Header** | The Authentication Header provides support for data integrity and authentication of IP packets. The data integrity feature ensures that undetected modification to a packet's content in transit is not possible. The authentication feature enables an end system or network device to authenticate the user or application and filter traffic accordingly; it also prevents the address spoofing attacks observed in today's Internet. The AH also guards against the replay attack. Authentication is based on the use of a message authentication code (MAC), hence the two parties must share a secret key. The Authentication Header consists of the following fields:    IPSec Authentication Header   * **Next Header (8 bits):** Identifies the type of header immediately following this header. * **Payload Length (8 bits):** Length of Authentication Header in 32-bit words, minus 2. For example, the default length of the authentication data field is 96 bits, or three 32-bit words. With a three-word fixed header, there are a total of six words in the header, and the Payload Length field has a value of 4. * **Reserved (16 bits):** For future use. * **Security Parameters Index (32 bits):** Identifies a security association. * **Sequence Number (32 bits):** A monotonically increasing counter value, discussed later. * **Authentication Data (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value (ICV), or MAC, for this packet.   **Encapsulating Security Payload:**  The Encapsulating Security Payload provides confidentiality services, including confidentiality of message contents and limited traffic flow confidentiality. As an optional feature, ESP can also provide an authentication service.  **ESP Format**  The following figure shows the format of an ESP packet. It contains the following fields:   * **Security Parameters Index** (32 bits): Identifies a security association. * **Sequence Number** (32 bits): A monotonically increasing counter value; this provides an anti-replay function, as discussed for AH. * **Payload Data (variable):** This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption. * **Padding (0-255 bytes):** This field is used to make the length of the plaintext to be a multiple of some desired number of bytes. It is also added to provide confidentiality. * **Pad Length (8 bits):** Indicates the number of pad bytes immediately preceding this field. * **Next Header (8 bits):** Identifies the type of data contained in the payload data field by identifying the first header in that payload (for example, an extension header in IPv6, or an upper-layer protocol such as TCP).     IPSec ESP format   * **Authentication Data (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value computed over the ESP packet minus the Authentication Data field. |
| **Web Security Threats** |  |
| **Secure Socket Layer/Transport Layer Security** | Secure Socket Layer (SSL) provides security services between TCP and applications that use TCP. The Internet standard version is called Transport Layer Service (TLS).  SSL/TLS provides confidentiality using symmetric encryption and message integrity using a message authentication code.  SSL/TLS includes protocol mechanisms to enable two TCP users to determine the security mechanisms and services they will use.  Netscape originated SSL. Version 3 of the protocol was designed with public review and input from industry and was published as an Internet draft document. Subsequently, when a consensus was reached to submit the protocol for Internet standardization, the TLS working group was formed within IETF to develop a common standard. This first published version of TLS can be viewed as essentially an SSLv3.1 and is very close to and backward compatible with SSLv3. |
| **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 16.2.  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. These SSL-specific protocols are used in the management of SSL exchanges and are examined later in this section.  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 (in the OSI layering model definition) 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. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.     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:** An arbitrary byte sequence chosen by the server to identify an active or resumable session state. * **Peer certificate:** An X509.v3 certificate of the peer. This element of the state may be null. * **Compression method:** The algorithm used to compress data prior to encryption. * **Cipher spec:** Specifies the bulk data encryption algorithm (such as null,AES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation. It also defines cryptographic attributes such as the hash\_size. * **Master secret:** 48-byte secret shared between the client and server. * **Is resumable:** A flag indicating whether 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 secret encryption key for data encrypted by the server and decrypted by the client. * Client write key: The symmetric 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 first initialized by the SSL Handshake Protocol. Thereafter, the final cipher-text block from each record is preserved for use as the IV with the following record. * 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. Sequence numbers may not exceed 264 – 1. |
| **SSL Record Protocol** | The SSL Record Protocol provides two services for SSL connections:   * Confidentiality: The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads. * Message Integrity: The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).   Figure 16.3 indicates the overall operation of the SSL Record Protocol. The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled before being delivered to higher-level users.    The first step is **fragmentation**. Each upper-layer message is fragmented into blocks of 214 bytes (16384 bytes) or less. Next, **compression** is optionally applied. Compression must be lossless and may not increase the content length by more than 1024 bytes.1In SSLv3 (as well as the current version of TLS), no compression algorithm is specified, so the default compression algorithm is null.  The next step in processing is to compute a **message authentication code** over the compressed data. For this purpose, a shared secret key is used. |
| **Firewalls** | Firewalls can be an effective means of protecting a local system or network of systems from network-based security threats while at the same time affording access to the outside world via wide area networks and the Internet.    **THE NEED FOR FIREWALLS**   * Centralized data processing system, with a central mainframe supporting a number of directly connected terminals * Local area networks (LANs) interconnecting PCs and terminals to each other and the mainframe * Premises network, consisting of a number of LANs, interconnecting PCs, servers, and perhaps a mainframe or two * Enterprise-wide network, consisting of multiple, geographically distributed premises networks interconnected by a private wide area network (WAN) * Internet connectivity, in which the various premises networks all hook into the Internet and may or may not also be connected by a private WAN   **FIREWALL CHARACTERISTICS**  The following design goals for a firewall:   1. All traffic from inside to outside, and vice versa, must pass through the firewall. This is achieved by physically blocking all access to the local network except via the firewall. Various configurations are possible available in fire wall. 2. Only authorized traffic, as defined by the local security policy, will be allowed to pass. Various types of firewalls are used, which implement various types of security policies, as explained later in this chapter. 3. The firewall itself is immune to penetration. This implies the use of a hardened system with a secured operating system. Trusted computer systems are suitable for hosting a firewall and often required in government applications   **Firewall controls:**  Firewalls use to control access and enforce the site’s security policy. Originally, firewalls focused primarily on service control, but they have since evolved to provide all four:   1. **Service control:** Determines the types of Internet services that can be accessed, inbound or outbound. The firewall may filter traffic on the basis of IP address, protocol, or port number; may provide proxy software that receives and interprets each service request before passing it on; or may host the server software itself, such as a Web or mail service. 2. **Direction control**: Determines the direction in which particular service requests may be initiated and allowed to flow through the firewall. 3. **User control:** Controls access to a service according to which user is attempting to access it. This feature is typically applied to users inside the firewall perimeter (local users). It may also be applied to incoming traffic from external users; the latter requires some form of secure authentication technology, such as is provided in IPsec. 4. **Behavior control:** Controls how particular services are used. For example, the firewall may filter e-mail to eliminate spam, or it may enable external access to only a portion of the information on a local Web server.   **Capabilities of firewall:**  The following capabilities are within the scope of a firewall:   1. A firewall defines a single choke point that keeps unauthorized users out of the protected network, prohibits potentially vulnerable services from entering or leaving the network, and provides protection from various kinds of IP spoofing and routing attacks. The use of a single choke point simplifies security management because security capabilities are consolidated on a single system or set of systems. 2. A firewall provides a location for monitoring security-related events. Audits and alarms can be implemented on the firewall system. 3. A firewall is a convenient platform for several Internet functions that are not security related. These include a network address translator, which maps local addresses to Internet addresses, and a network management function that audits or logs Internet usage. 4. A firewall can serve as the platform for IPsec.   **Limitations of Firewalls:**  Firewalls have their limitations   1. The firewall cannot protect against attacks that bypass the firewall. Internal systems may have dial-out capability to connect to an ISP. An internal LAN may support a modem pool that provides dial-in capability for travelling employees and telecommuters. 2. The firewall may not protect fully against internal threats, such as a disgruntled employee or an employee who unwittingly cooperates with an external attacker. 3. An improperly secured wireless LAN may be accessed from outside the organization. An internal firewall that separates portions of an enterprise network cannot guard against wireless communications between local systems on different sides of the internal firewall. 4. A laptop, PDA, or portable storage device may be used and infected outside the corporate network, and then attached and used internally.   **TYPES OF FIREWALLS:**   * Packet Filtering Firewall * Application Level Gateway * Circuit Level Gateway   **Packet Filtering Firewall:**  Packet filtering firewall applies a set of rules to each incoming and outgoing IP packet and then forwards or discards the packet .The firewall is typically configured to filter packets going in both directions (from and to the internal network). Filtering rules are based on information contained in a network packet:   * **Source IP address:** The IP address of the system that originated the IP packet (e.g., 192.178.1.1) * **Destination IP address:** The IP address of the system the IP packet is trying to reach (e.g., 192.168.1.2) * **Source and destination transport-level address:** The transport-level (e.g., TCP or UDP) port number, which defines applications such as SNMP or TELNET * **IP protocol field:** Defines the transport protocol * **Interface:** For a firewall with three or more ports, which interface of the firewall the packet came from or which interface of the firewall the packet is destined for. * possible default policies   Default = discard: That which is not expressly permitted is prohibited.  Default = forward: That which is not expressly prohibited is permitted.    **Application Level Gateway (or Proxy)**  •have application specific gateway / proxy  •has full access to Protocol  User requests service from proxy  Proxy validates request as legal  Then actions request and returns result to user  Can log / audit traffic at application level.  •Need separate proxies for each service  Some services naturally support proxying  Others are more problematic.  **Circuit Level Gateway:**   * It is a stand a-lone system or it can be a specialized function performed by an application level gate way for certain applications. * It does not permit end to end TCP connection; this relays two TCP connections, one between itself and a TCP user on an inner host, and one between itself and TCP user on outside host. * Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the content.       **Firewall Configurations**  Ch20. Firewall Configs.pdf                                     002F6F4DMacintosh HD                   B83AE914:  Ch20. Firewall Configs.pdf                                     002F6F4DMacintosh HD                   B83AE914:  Ch20. Firewall Configs.pdf                                     002F6F4DMacintosh HD                   B83AE914:  **Bastion Host:**  A bastion host is a system identified by the firewall administrator as a critical strong point in the network's security. Typically, the bastion host serves as a platform for an application-level or circuit-level gateway. Common characteristics of a bastion host include the following:   * The bastion host hardware platform executes a secure version of its operating system, making it a trusted system. * Only the services that the network administrator considers essential are installed on the bastion host. These include proxy applications such as Telnet, DNS, FTP, SMTP, and user authentication. * The bastion host may require additional authentication before a user is allowed access to the proxy services. In addition, each proxy service may require its own authentication before granting user access. * Each proxy is configured to support only a subset of the standard application's command set.   **Screened host firewall:**  In the screened host firewall, single-homed bastion configuration (Figure 20.2a), the firewall consists of two systems: a packet-filtering router and a bastion host. Typically, the router is configured so that  1. For traffic from the Internet, only IP packets destined for the bastion host are allowed in.  2. For traffic from the internal network, only IP packets from the bastion host are allowed out.  The bastion host performs authentication and proxy functions. This configuration has greater security than simply a packet-filtering router or an application-level gateway alone, for two reasons. First, this configuration implements both packet-level and application-level filtering, allowing for considerable flexibility in defining security policy. Second, an intruder must generally penetrate two separate systems before the security of the internal network is compromised.  This configuration also affords flexibility in providing direct Internet access. For example, the internal network may include a public information server, such as a Web server, for which a high level of security is not required. In that case, the router can be configured to allow direct traffic between the information server and the Internet**.**  **Screened host firewall, dual-homed bastion:**  In the single-homed configuration just described, if the packet-filtering router is completely compromised, traffic could flow directly through the router between the Internet and other hosts on the private network. The **screened host firewall, dual-homed bastion** configuration physically prevents such a security breach (Figure 20.2b). The advantages of dual layers of security that were present in the previous configuration are present here as well. Again, an information server or other hosts can be allowed direct communication with the router if this is in accord with the security policy.  **Screened subnet firewall configuration:**  The screened subnet firewall configuration of Figure 20.2c is the most secure of those we have considered. In this configuration, two packet-filtering routers are used, one between the bastion host and the Internet and one between the bastion host and the internal network. This configuration creates an isolated sub-network, which may consist of simply the bastion host but may also include one or more information servers and modems for dial-in capability. Typically, both the Internet and the internal network have access to hosts on the screened subnet, but traffic across the screened subnet is blocked.  **This configuration offers several advantages**:   * There are now three levels of defense to thwart intruders. * The outside router advertises only the existence of the screened subnet to the Internet; therefore, the internal network is invisible to the Internet. * Similarly, the inside router advertises only the existence of the screened subnet to the internal network; therefore, the systems on the inside network cannot construct direct routes to the Internet. |

**Questions**

1. Explain secure electronic commerce components in detail
2. Explain benefits of IPSec
3. Explain the SSL Record Protocol
4. Explain Web Security threats
5. Describe IP Security
6. Explain IPSec ESP format
7. Explain the IPSec Authentication Header fields with diagram
8. Explain the payment process supported in SET
9. Explain briefly the format of ISAKMP header and generic payload types
10. Briefly describe encapsulating a security payload
11. Briefly describe the features of Oakley’s key determination protocol.
12. What is WWW? What are the challenges web presents?
13. Briefly describe the dual signature in SET.
14. Briefly describe anti-replay service and integrity check value
15. Briefly describe about ISAKMP exchanges
16. Briefly explain the following:
17. i) Trapdoors ii) logic bomb iii) Trojan horse iv) Viruses
18. Explain the concept of trusted systems
19. Explain password selection procedure in detail.
20. Explain the capabilities and limitations of firewalls
21. Explain various approaches to Intrusion Detection.
22. What is a firewall? Explain packet filter router.
23. Describe different classes of Intruders.
24. Explain malicious programs
25. Describe trusted system in detail.
26. What is a firewall? Explain different types of firewalls. Explain the characteristics and

capabilities of firewall?