Cryptography (18CS744)

Module-1

* Classical Encryption Techniques

→ An original message is known as the plaintext while the coded message is called the ciphertext.

→ The process of converting from plaintext to ciphertext is known as enciphering or encryption; restoring the plaintext from the ciphertext is deciphering or decryption. The many schemes used for encryption constitute the area of study known as Cryptography.

→ Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of Cryptanalysis.

Cryptanalysis - breaking the code

Cryptology - Cryptography + Cryptanalysis

* Symmetric cipher model

A Symmetric encryption scheme has five ingredients:

Plaintext - This is the original intelligible message or data that is fed into the algorithm as input.

Encryption algorithm - The encryption algorithm performs various substitutions & transformations on the plaintext.

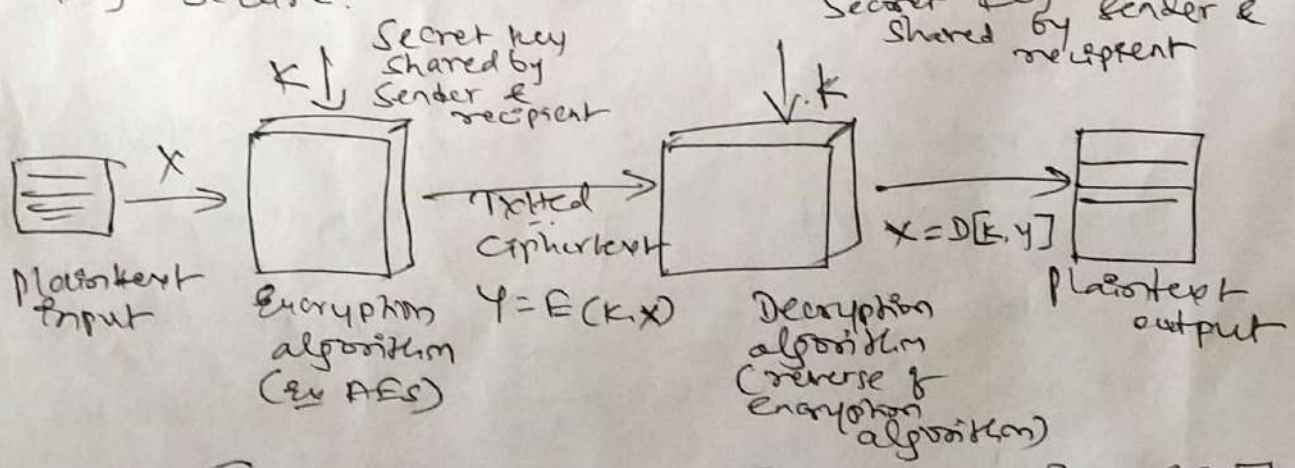
Secret key- It also input to the encryption algorithm. The key is a value independent of the plaintext & of the algorithm.

Ciphertext - This is the scrambled message produced as output. It depends on the plaintext & Secret key.

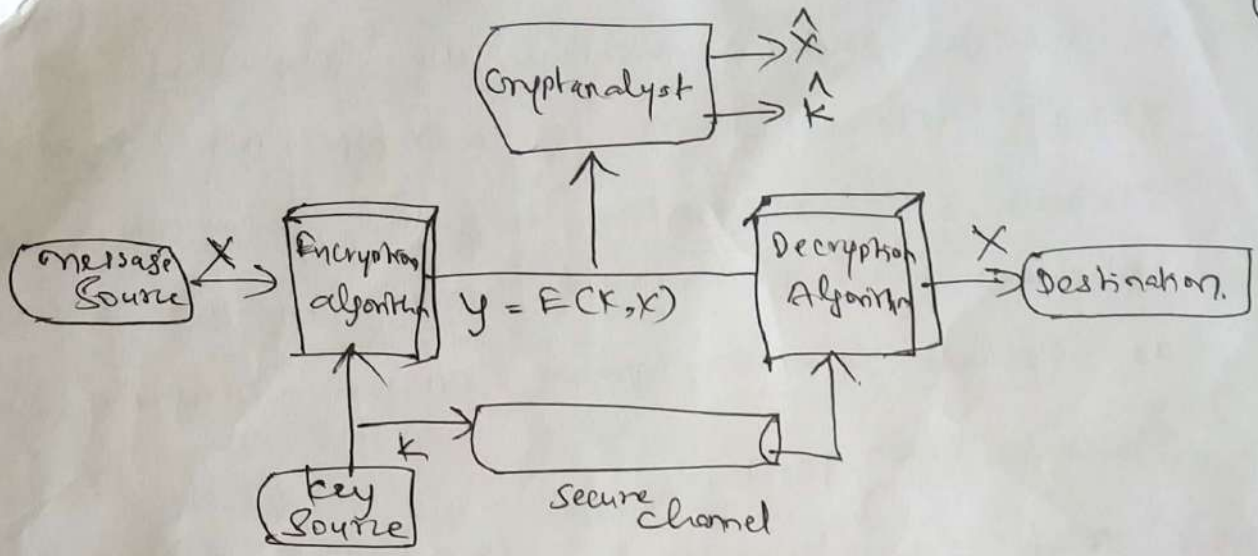
Decryption algorithm This is essentially the encryption algorithm run in reverse.

There are two requirements for secure use of Conventional encryption:

1. We need a strong encryption algorithm. This requirement is usually stated in a stronger form: The opponent should be unable to decrypt Ciphertext or discover the key even if he or she is in possession of a no. of Ciphertexts together with the plaintext that produced each Ciphertext.
2. Sender & receiver must have obtained copies of the secret key in a secure fashion & must keep the key secure.



[Simplified model of Symmetric Encryption]



(Model of Symmetric cryptosystem)

A source produces a message in plaintext $X = [X_1, X_2, \ldots X_M]$. For encryption, a key of the form $K = [K_1, K_2 \ldots K_J]$ is generated.

With the message X and the encryption key K as input the encryption algorithm forms the ciphertext $Y = [Y_1, Y_2 \ldots Y_N]$. We can write this as

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

The receiver, in possession of the key is able to invert the transformation:

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

$\hat{X}$ - estimate plaintext

$\hat{K}$ - estimate key

* Cryptography - Cryptographic systems are characterized along three independent dimensions:

1. The type of operations used for transforming plaintext to ciphertext. All encryption algorithms are based on two general principles: substitution,

in which each element in the plaintext (bit, letter, group of bits or letters) is mapped into another element & transposition in which elements in the plaintext are rearranged. most systems, referred to as product systems, involve multiple stages of substitutions & transpositions.

2. The no. of keys used If both sender & receiver use the same key the system is referred to as symmetric, secret key or conventional encryption. If the sender & receiver use different keys, the system is referred to as asymmetric, two-key or public key encryption.

3. The way in which the plaintext is processed A block cipher processes the input one block of elements at a time, producing an o/p block for each input block. A stream cipher processes the input elements continuously, producing output one element at a time as it goes along.

* Cryptanalysis and Brute-force Attack There are two general approaches to attacking a conventional encryption scheme:

↳ Cryptanalysis - This type of attack exploits the characteristics of the algorithm to attempt to deduce a

a Specific plaintext or to deduce the key being used.

↳ Brute-force attack The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained.

The following summarizes the various types of cryptanalytic attacks based on the amount of information known to the cryptanalyst.

Type of Attack	Known to Cryptanalyst
Ciphertext only	→ Encryption algorithm → Ciphertext
Known plaintext	→ Encryption algorithm → Ciphertext → one or more plaintext-ciphertext pairs formed with secret key
Chosen-plaintext	→ Encryption algorithm → Ciphertext → plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen-ciphertext	→ Encryption algorithm → Ciphertext → Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with secret key

Chosen text → Encryption algorithm

→ Ciphertext

→ plaintext message chosen by cryptanalyst, together with its corresponding Ciphertext generated with the secret key

→ Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

---

→ An encryption scheme is <u>unconditionally secure</u> if the ciphertext generated by the scheme does not contain enough information to determine uniquely the corresponding plaintext, no matter how much ciphertext is available. [No matter how much time an opponent has it is impossible for him or her to decrypt the ciphertext simply because the required information is not there.]

→ The users of an encryption algorithm can strive for is an algorithm that meets one or both of the following criteria:

↳ The cost of breaking cipher exceeds the value of the encrypted information.

↳ The time required to break the cipher exceeds the useful lifetime of the information.

An encryption scheme is said to be Computationally Secure if either of the foregoing two criteria are met. Unfortunately, it is very difficult to estimate the amount of effort required to cryptanalyze ciphertext successfully.

* 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 ciphertext bit patterns.

Ceaser Cipher - involves replacing each letter of the alphabet with the letter standing three places further down the alphabet. For Example

Plain : meet me after the toga party

Cipher : PHHW PH DIWHU WKH WRJD SDUWB

Note that the alphabet is wrapped around so that the letter following Z is A.

Let us assign a numerical equivalent to each ~~other~~ letter:

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

Then the algorithm can be expressed as follows. For each plaintext letter P, substitute the ciphertext letter C:

$$C = E(3, P) = (P + 3) \bmod 26$$

A shift may be of any amount, so that the general Ceaser algorithm is

$$C = E(K, P) = (P + K) \bmod 26$$

where K takes on a value in the range 1 to 25. The decryption algorithm is simply

$$P = D(K, C) = (C - K) \bmod 26$$

If it is known that a given ciphertext is a Ceaser cipher, then a brute-force cryptanalysis is easily performed: simply try all the 25 possible keys.

→ Three important characteristics of this problem enabled us to use a brute force analysis:

1. The encryption & decryption algorithm are known.
2. There are only 25 keys to try.
3. The language of the plaintext is known and easily recognizable.

monoalphabetic ciphers

With only 25 possible keys the ceaser cipher is far from secure. A dramatic increase in the key space can be achieved by allowing an arbitrary substitution. Before proceeding, we define the term permutation. A permutation of a finite set of elements is an ordered sequence of

all the elements of S, with element appearing exactly once. For example, if S = {a, b, c} there are six permutations of S:

abc, acb, bac, bca, cab, cba

In general there are n! permutations of a set of n elements, because the first element can be chosen in one of n ways, the second in n-1 ways, the third in n-2 ways & so on.

If instead, the "cipher" line can be any permutation of the 26 alphabetic characters, then there are 26! possible keys. Such an approach is referred to as a monoalphabetic substitution cipher, because a single cipher alphabet is used per message.

→ Monoalphabetic cipher is a substitution cipher in which for a given key, the cipher alphabet for each plain alphabet is fixed throughout the encryption process.

* Playfair cipher → The best known multiple-letter encryption cipher is the Playfair cipher, which treats diagrams in the plaintext as single units & translates these into ciphertext diagrams.

→ The Playfair algorithm is based on the use of 5×5 matrix of letters constructed using a keyword.

Eg If sender & receiver decide on a particular key "tutorials" the first characters (going left to right) in the table is the phrase excluding duplicate letters.

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

Suppose the message 'Hide money' need to be encrypted. Firstly it is split into pairs of two letters. If there is an odd no. of letters a Z is added to the last letter.

HI DE MO NE YZ

The rules of Encryption are -

→ If both the letters are in the same column take the letter below each one (going back to the top if at the bottom.

So, HI → QC

→ If both the letters are in the same row, take the letter to the right of each one (going back to the left if at the farthest right)

So, DE → EF

→ If neither of the preceding two rules are true, form a rectangle with two letters & take the letters on the horizontal opposite corner of the rectangle.

So, MO → NU

NE → MF & YZ → ZV

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

AR → RM

MU → CM

HS → BP

EA → IM or JM

* **Hill Cipher** Another multiletter cipher is the Hill cipher developed by Lester Hill in 1929.

Concepts from linear algebra (Matrix arithmetic modulo 26)

we define M^{-1} as inverse of a matrix M such that $M(M^{-1}) = M^{-1}M = I$ where I is identity matrix.

For Eg. $A = \begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix}$ $A^{-1} \bmod 26 = \begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix}$

$$A A^{-1} = \begin{pmatrix} 5\times9 + 8\times1 & 5\times2 + 8\times15 \\ 17\times9 + 3\times1 & 17\times2 + 3\times15 \end{pmatrix}$$

$$= \begin{pmatrix} 53 & 130 \\ 156 & 79 \end{pmatrix} \bmod 26 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

⇒ To explain how the inverse of matrix is computed, we begin by with the concept of determinant. For any square matrix (m×m) the determinant equals the sum of all the products that can be formed by taking exactly one element from each row & exactly one element from each column, with certain of the product terms preceded

by minus sign. For 2×2 matrix

$$\begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix}$$

the determinant is $k_{11}k_{22} - k_{12}k_{21}$.

for 3×3 matrix

the determinant is $k_{11}k_{22}k_{33} + k_{21}k_{32}k_{13} + k_{31}k_{12}k_{23} - k_{31}k_{22}k_{13} - k_{21}k_{12}k_{33} - k_{11}k_{32}k_{23}$

So determinant of $\begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix} = (5\times3) - (8\times17)$

$= -121 \bmod 26 = 9.$

$-121 \bmod 26 = -121 - 26 * \text{floor}(-121/26)$

$= -121 - 26*(-5)$

$= -121 + 130$

$= 9$

we can show that $9^{-1} \bmod 26 = 3$ because $9\times3 = 27 \bmod 26 = 1$. Therefore we can compute the inverse of A as

$$A = \begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix}$$

$$A^{-1} \bmod 26 = 3\begin{pmatrix} 3 & -8 \\ -17 & 5 \end{pmatrix} = 3\begin{pmatrix} 3 & 18 \\ 9 & 5 \end{pmatrix} = \begin{pmatrix} 9 & 54 \\ 27 & 15 \end{pmatrix} = \begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix}$$

Assignment $K = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$ what is K^{-1}?

The Hill Algorithm This encryption algorithm takes m successive plaintext letters and substitutes for them m ciphertext letters. The substitution is determined by m linear equations in which each character is assigned a numerical value (a=0, b=1, ... z=25). for m=3, the system can be described as

$$C_1 = (K_{11}P_1 + K_{12}P_2 + K_{13}P_3) \bmod 26$$

$$C_2 = (K_{21}P_1 + K_{22}P_2 + K_{23}P_3) \bmod 26$$

$$C_3 = (K_{31}P_1 + K_{32}P_2 + K_{33}P_3) \bmod 26$$

This can be expressed in terms of row vectors and matrices:

$$(C_1\ C_2\ C_3) = (P_1\ P_2\ P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \bmod 26$$

or

$$C = PK \bmod 26$$

Eg plaintext "paymoremoney"

& key, $K = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$

The first three letters of the plaintext are represented by the vector (15 0 24). Then (15 0 24) K = (303 303 531) mod 26 = (17 17 11) = RRL. Continuing this fashion, the ciphertext for the entire plaintext is RRLMWBKASPDH.

→ Decryption requires using the inverse of the matrix K.

$$K^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 3 & 17 \end{pmatrix} \quad [\text{determinant } K = 23]$$

$$\Rightarrow P = CK^{-1} \bmod 26 = P$$

* 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 substitution cipher.

→ These techniques have the following features in common:

1. A set of related monoalphabetic substitution rules is used.
2. A key determines which particular rule is chosen for a given transformation.

Vigenere Cipher – In this scheme a set of monoalphabetic substitution rules consists of the 26 ceaser ciphers with shifts of 0 through 25.

→ Assume a sequence of plaintext letters $P = P_0, P_1, P_2 \dots P_{n-1}$ & a key consisting of the sequence of letters $K = K_0, K_1, \dots K_{m-1}$, where typically $m \leq n$. The sequence of ciphertext letters $C = C_0, C_1, C_2 \dots C_{n-1}$ is calculated as follows:

$$C = C_0, C_1, C_2 \dots C_{n-1} = E(K, P)$$
$$= E[(K_0, K_1, K_2 \dots K_{m-1}), (P_0, P_1, P_2, \dots P_{n-1})]$$

$$= (P_0 + K_0) \bmod 26, (P_1 + K_1) \bmod 26, \dots (P_{m-1} + K_{m-1}) \bmod 26,$$
$$(P_m + K_0) \bmod 26, (P_{m+1} + K_1) \bmod 26, \dots$$
$$(P_{2m-1} + K_{m-1}) \bmod 26, \dots$$

A general equation of the encryption process is

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

Similarly, decryption is a generalization of equation:

$$p_i = (C_i - k_i \bmod m) \bmod 26$$

→ To encrypt a message, a key is needed that is as long as the message. Usually the key is a repeating keyword.

For eg if the keyword is deceptive the message "we are discovered save yourself" is encrypted as

Key : deceptivedeceptivedeceptive

Plaintext : wearediscoveredsaveyourself

Ciphertext : ZICVTWQBNGRZGVTWAVZHCQYGLMGJ

Vernam Cipher

→ Introduced by an AT&T engineer named Gilbert Vernam in 1918.

→ Here System works on binary data (bits) rather than letters. The System can be expressed as follows

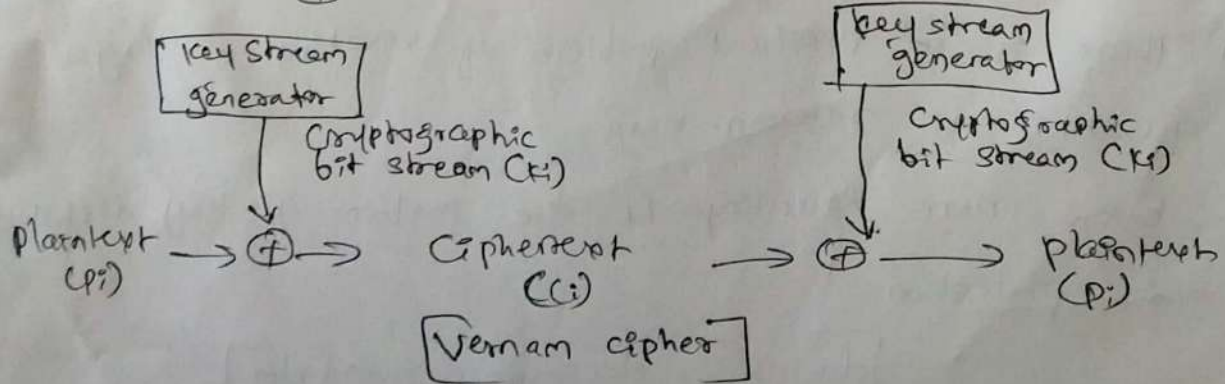
$$C_i = P_i \oplus k_i$$

where P_i = ith binary digit of plaintext

k_i = ith binary digit of key

C_i = ith binary digit of ciphertext

$\oplus$ = exclusive-or (XOR) operation



$P_i = C_i \oplus K_i$

Eg $P_i = 0011$ $K_i = 1011$

$C_i = P_i \oplus K_i$

$= 0011$
$\quad 1011$
$\quad 1000$

$P_i = C_i \oplus K_i = 1000$
$\quad 1011$
$\quad 0011$

One Time pad

Mauborgne Suggested using a random key that is as long as the message, so that the key need not be repeated.

→ Each new message requires a new ~~message~~ key of the same length as the new message. Such a scheme is known as a <u>one-time pad</u>, is unbreakable

Eg mr mustard (PT)
PXLMVMSYDO (key)
ANKYODKYURE (CT)
(Vigenere cipher)

→ The one time pad offers Completely Security but in practice has two fundamental difficulties:

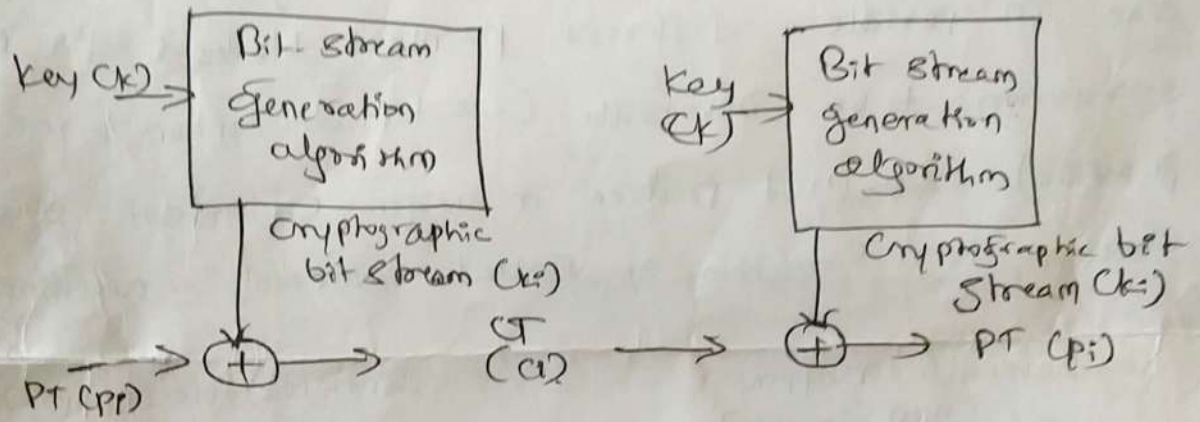
1. There is the practical problem of making a large quantity of random keys.
2. Even more daunting is the problem of key distribution and protection.

[daunt – discourage, overhelm]

* Block ciphers & data Encryption standard
* Block Cipher principles

Stream ciphers & Block ciphers A stream cipher is one that encrypts a digital data stream one bit or one byte at a time.

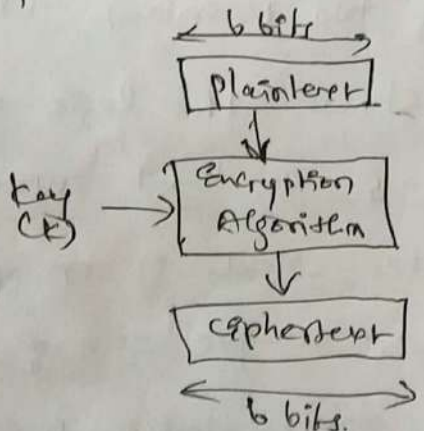
→ In the following figure, the bit stream generator is a key controlled algorithm & must produce a bit stream that is cryptographically strong.



(Stream cipher using algorithmic bit-stream generator)

→ A block cipher is one in which a block of plaintext is treated as a whole & used to produce ciphertext block of equal length

→ Typically a block size of 64 or 128 bits is used.



(Block cipher)

→ many block ciphers have a feistel structure. Such a structure consists of a no. of identical rounds of processing. In each round a substitution is performed followed by permutation. The original key is expanded so that a different key is used for each round.

motivation for the Feistel Cipher structure

A block cipher operates on a plaintext block of n bits to produce a ciphertext block of n bits. There are 2^n possible different plaintext blocks & for the encryption to be reversible (i.e for decryption to be possible) each must produce a unique ciphertext block. Such a transformation is called reversible or non singular.

Reversible mapping (non Singular)

PT	CT
00	11
01	10
10	00
11	01

Irreversible mapping (Singular)

PT	CT
00	11
01	10
10	01
11	01

In the latter case, a ciphertext of 01 could have been produced by one of two plaintext blocks.

→ The following figure shows the logic of general substitution cipher for n=4. A 4-bit input produces one of 16 possible input states, which is mapped by the substitution cipher into a unique one of 16 possible o/p states, each of which is represented by 4 ciphertext bits.

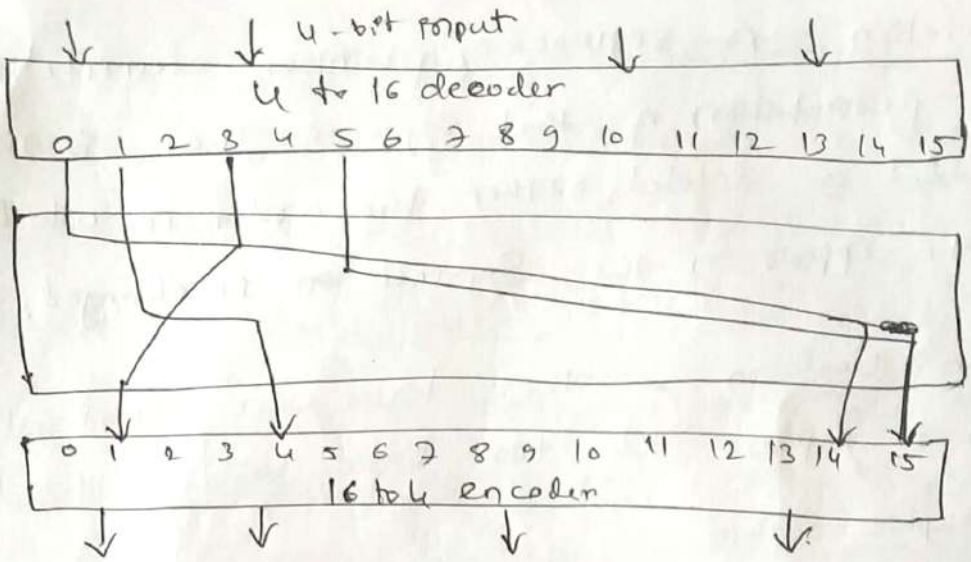


Figure: General n-bit-n bit Block Substitution (Shown with n=4)

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

(Encryption & Decryption Tables for Substitution Cipher)

The Feistel Cipher - Feistel proposed the use of Cipher that alternates Substitutions & permutations:

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, rather the order in which the elements appear in the sequence is changed.

Diffusion & Confusion - These terms were introduced by claude shannon to capture the two basic building blocks for any cryptographic system.

→ In diffusion, the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext. This is achieved by having each plaintext digit affect the value of many ciphertext digits; generally this is equivalent to having each ciphertext digit be affected by many plaintext digits.

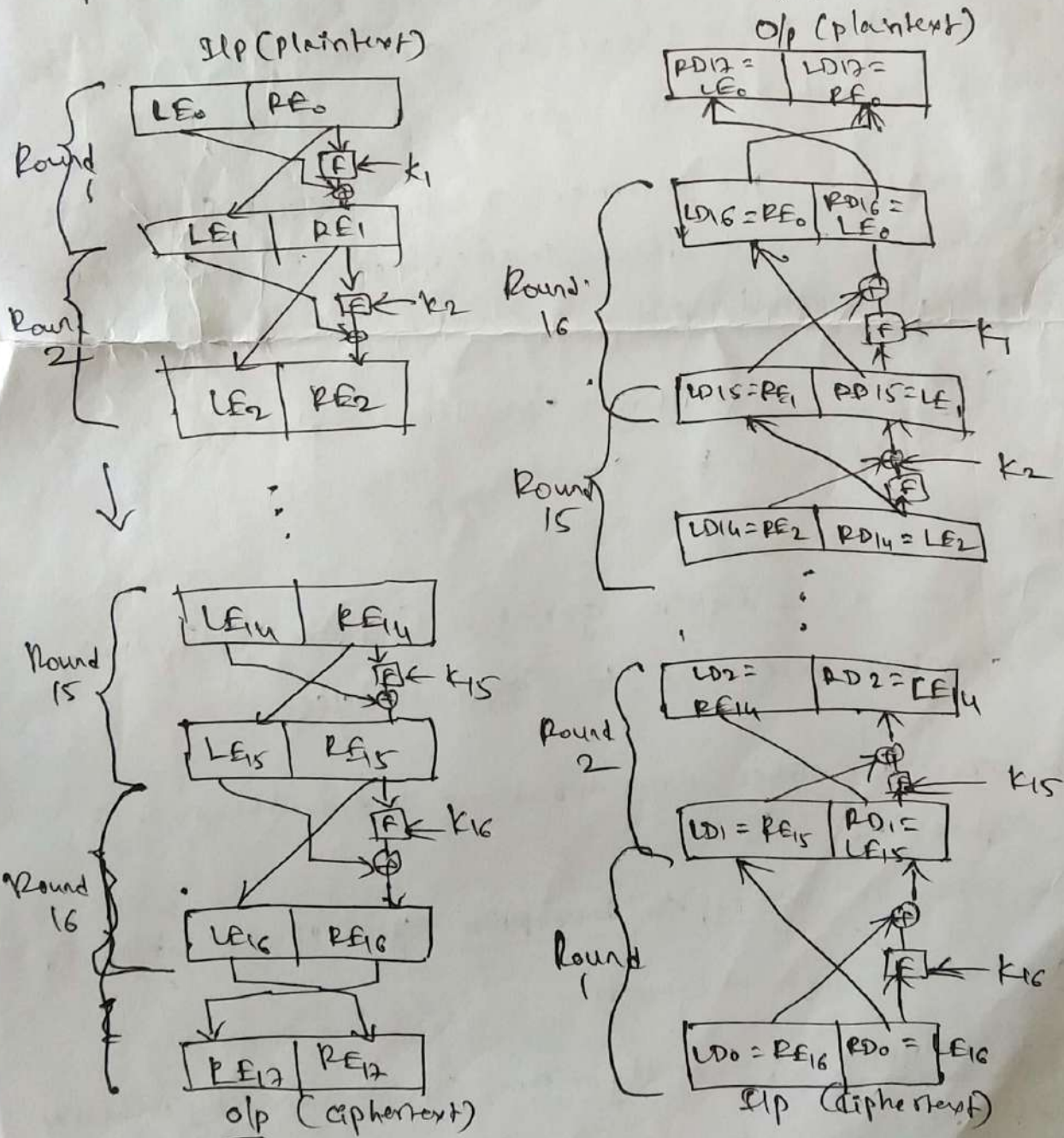
→ The mechanism of diffusion seeks to make the statistical relationship between the plaintext and ciphertext as complex as possible in order to thwart attempts to deduce key.

On the other hand, confusion seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart attempts to discover the key.

Feistel Cipher structure - The plaintext block is divided into two halves, L_0 and R_0. 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 L_{i-1} & R_{i-1} derived from the previous round as well

as a subkey K_i derived from the overall K.

→ 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 & then taking the exclusive-OR of the o/p of that function & the left half of that data. The round fun. has the same general structure for each round but is parameterized by the round subkey K_i.



Feistel Encryption and Decryption (16 rounds)

the exact realization of a Feistel n/w depends on the choice of the following parameters and design features:

Block Size - Larger the block size mean greater security being but reduced encryption/decryption speed for a given algorithm. The greater security is achieved by greater diffusion.

Key Size - Larger key size means greater security but may decrease encryption/decryption speed. The greater security is achieved by greater resistance to brute-force attacks. and greater confusion.

Number of rounds - The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security.

Subkey generation algorithm - Greater complexity in this algorithm should lead to greater difficulty of crypt-analysis.

Round function F - Again greater complexity generally means greater resistance to cryptanalysis.

⇒ There are two other considerations in the design of a Feistel cipher:

↳ Fast software encryption/decryption - Speed of execution of the algorithm becomes a concern.

↳ Ease of analysis - If the algorithm can be concisely & clearly explained, it is easier to analyze that algorithm for cryptographic vulnerabilities to develop higher level of security strength

Consider the encryption process, we see that

$$L_{16} = RE_{15}$$

$$RE_{16} = LE_{15} \oplus f(RE_{15}, K_{16})$$

On the decryption side,

$$LD_1 = RD_0 = LE_{16} = RE_{15}$$

$$RD_1 = LD_0 \oplus F(RD_0, K_{16})$$

$$= RE_{16} \oplus F(RE_{15}, K_{16})$$

$$= [LE_{15} \oplus F(LE_{15}, K_{16})] \oplus F(RE_{15}, K_{16})$$

The XOR has the following properties:

$$[A \oplus B] \oplus C = A \oplus [B \oplus C]$$

$$D \oplus D = 0$$

$$E \oplus 0 = E$$

Thus we have $LD_i = RE_{15}$ & $RD_1 = LE_{15}$

for the ith iteration of the encryption algorithm

$$LE_i = RE_{i-1}$$

$$RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i)$$

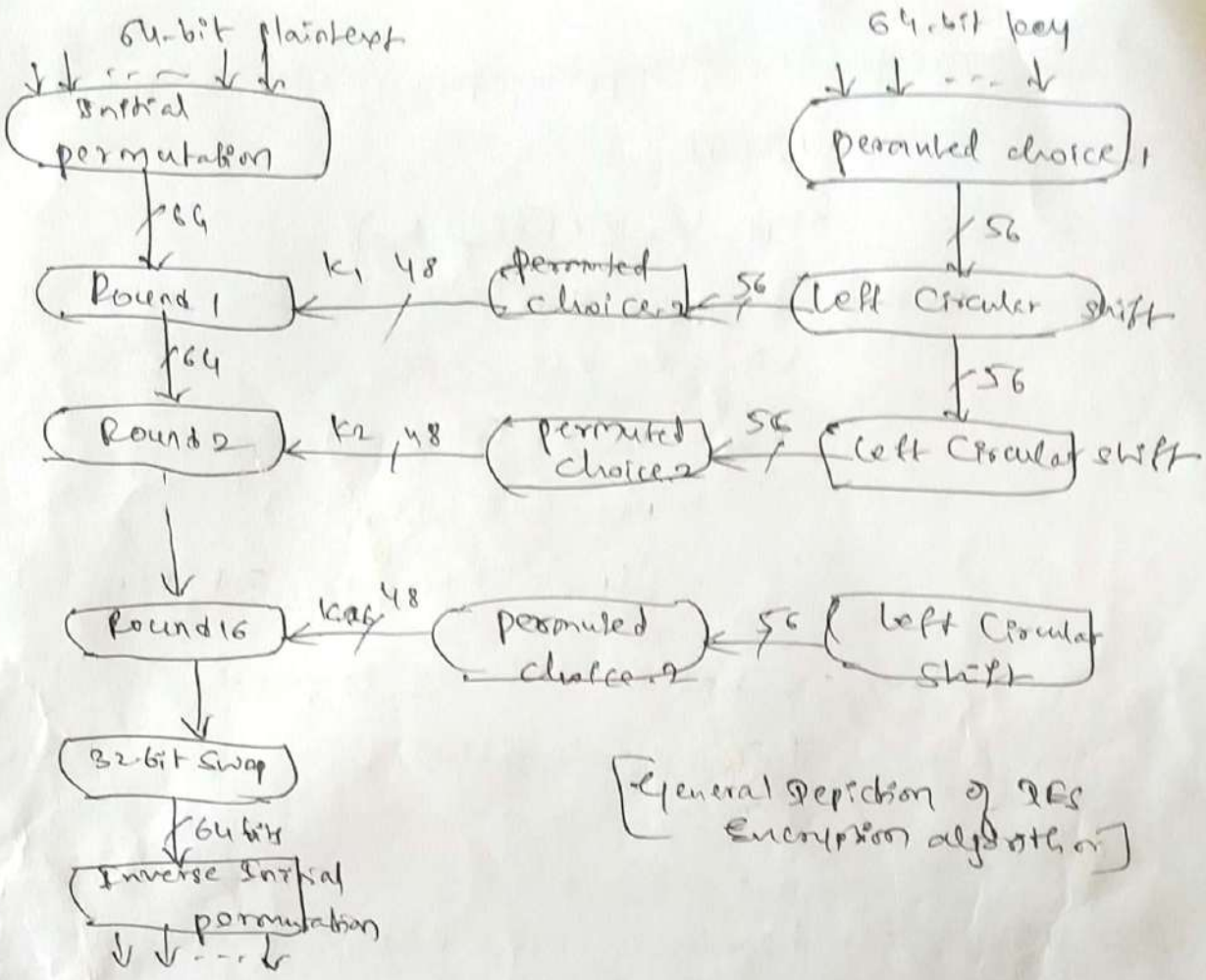
* The Data Encryption Standard

→ Adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards & Technology (NIST) as Federal Information processing standard.

DES Encryption

→ plaintext must be 64 bits in length & the key is 56 bits in length.

→ As with any encryption scheme there are two inputs to the encryption function: the plaintext to be encrypted and the key.



[General Depiction of DES Encryption algorithm]

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

[Initial permutation]

* The Strength of DES - There are two concerns by & large fall into two areas: key size & the nature of the DES algorithm.

The use of 56-bit keys - With a key length of 56 bits, there are 2^{56} possible keys, which is approximately 7.2×10^{16} keys. Thus on the face of it, a brute force attack appears impractical.

→ Assuming that, on average, half of the key space has to be searched a single machine performing one DES encryption per microsecond would take more than a thousand years to break the cipher.

→ A single pc can break DES in about a year; if multiple pcs work in parallel, the time is drastically shortened.

Fortunately there are a no. of alternatives to DES, the most important of which are AES & triple DES.

The Nature of DES algorithm The focus of concern has been on the eight substitution tables or S-boxes, that are used in each iteration. Because the design criteria for these boxes & indeed for the entire algorithm were not made public, there is a suspicion that the boxes were constructed in such a way that cryptanalysis is possible, for an opponent who knowns the weaknesses in the S-boxes.

Timing attacks - A Timing attack is one in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various ciphertexts. A timing attack exploits the fact that an encryption & decryption algorithm often takes slightly different amounts of time on different inputs. The authors conclude that DES appears to be fairly resistance to a successful timing attack but suggest some avenues to explore.

* Block cipher Design principles - There are three critical

aspects of block cipher design : the no. of rounds, design of the function F and key scheduling.

No. of rounds - The greater the no. of rounds, the more difficult it is to perform cryptanalysis, even for a relatively weak F.

→ In general, the criteria should be that the no. of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple brute-force attack.

Design of function F - This provids the element of confusion in a feistel cipher. Thus it must be difficult to unscramble the substitution performed by F. one obvious criteria is that F be nonlinear. The more non-linear F, the more difficulty any type of cryptanalysis.

→ we would like the algorithm to have good avalanche properties. This means a change in one bit of the i/p should produce a change in many bits of the output. (strict avalanche criteria) & Bit independent criteria (Two o/p bits should change independently when any single bit is inverted for all.

key schedule algorithm with any feistel block cipher the key is used to generate the subkey for each round. In general we would like to select subkeys to maximize the difficulty of deducing individual subkeys and difficulty of working back to the main key.

—ò—

Module. 2

Public key Cryptography & RSA

Principles of public-key cryptosystems - The concept of the public key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption. The first problem is that of key distribution.

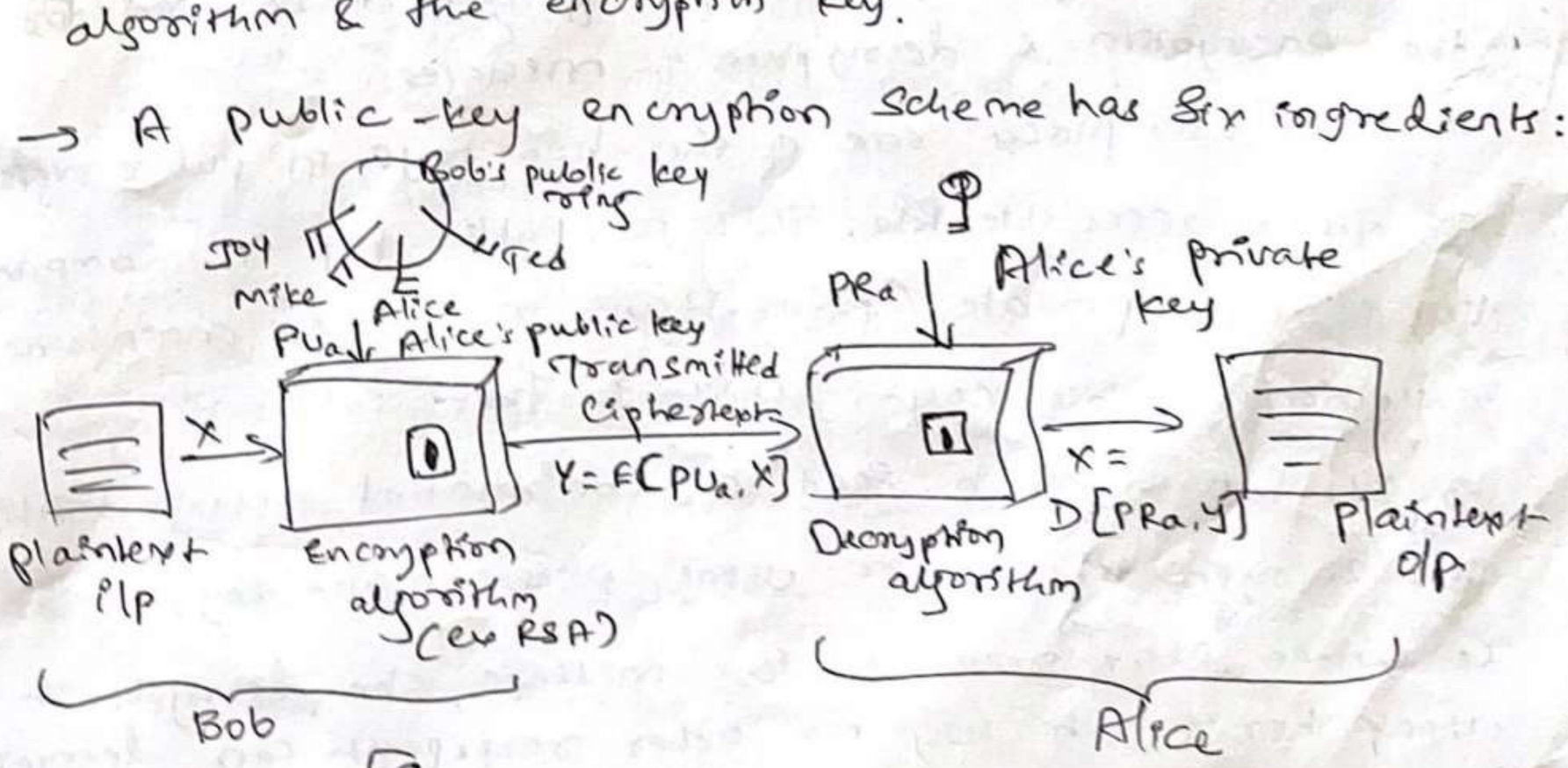
→ **key distribution** under symmetric encryption requires either 1) that two communicants already share a key which somehow has been distributed to them or 2) the use of a key distribution center.

→ The second problem that Diffie pondered, and one that was apparently unrelated to the first was that of ~~digital signatures~~.

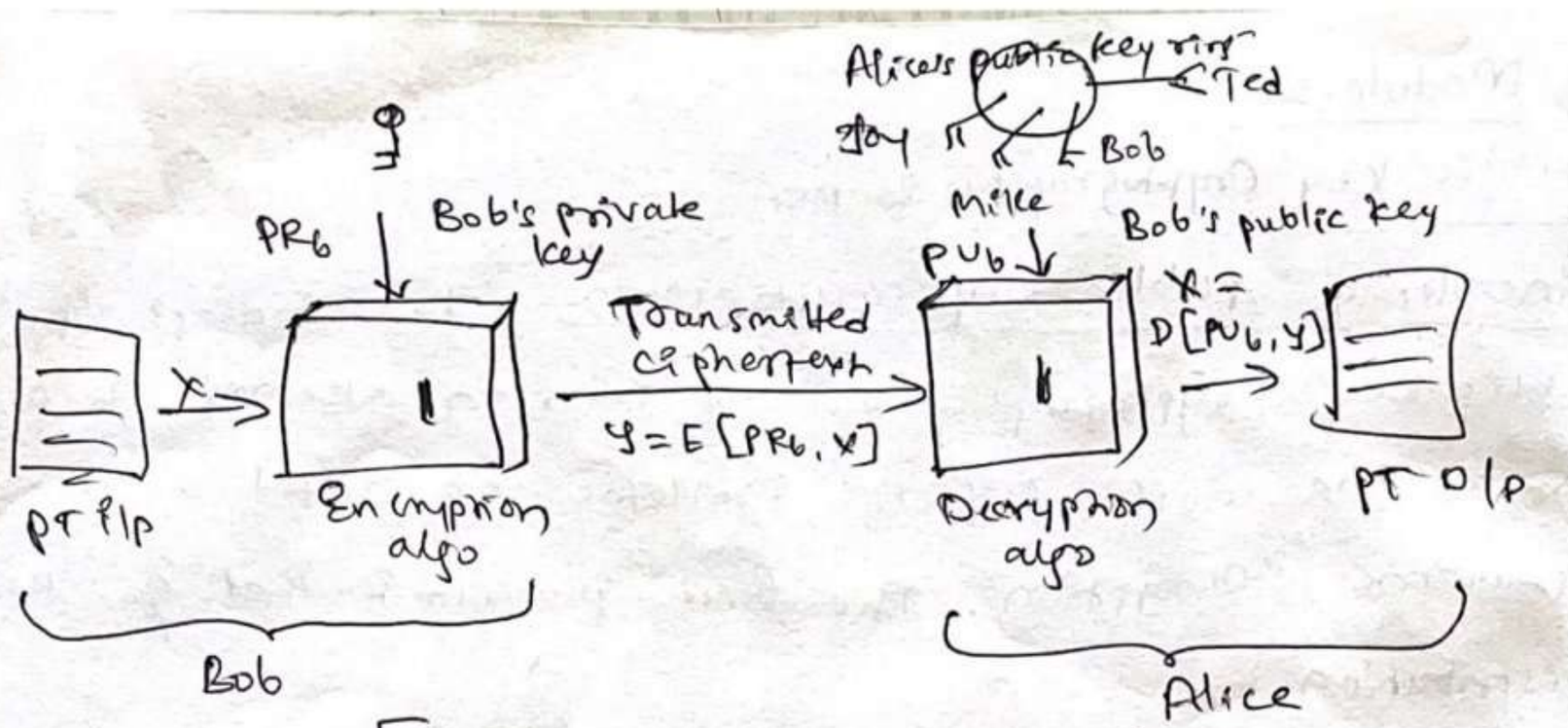
public key cryptosystems - These algorithms have the following important characteristic:

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

→ A public-key encryption scheme has six ingredients:



[Encryption with public key]



[Encryption with private key]

Plaintext - This is readable message or data that is fed into the algorithm as input.

Encryption algorithm - performs various transformations on the plaintext.

Public & private keys - This is a pair of keys that have been selected.

Ciphertext - This is scrambled message produced as o/p.

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

The essential steps are the following:

1. Each user generates a pair of keys to be used for the encryption & decryption of messages.
2. Each user places one of the two keys in public register or other accessible file. This is public key. The companion key is kept private. (From figure, 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 because Alice knows Alice's private key.

Conventional encryption

Needed to work:

1. The same algorithm with the same key is used for encryption & decryption.

2. The sender & receiver must share the algorithm & key.

Needed for Security

1. The key must be kept secret.

2. It must be impossible or atleast impractical to decipher a message if the key is kept secret.

3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.

Public key encryption

Needed to work:

1. one algorithm is used for encryption & a related algorithm for decryption with a pair of keys

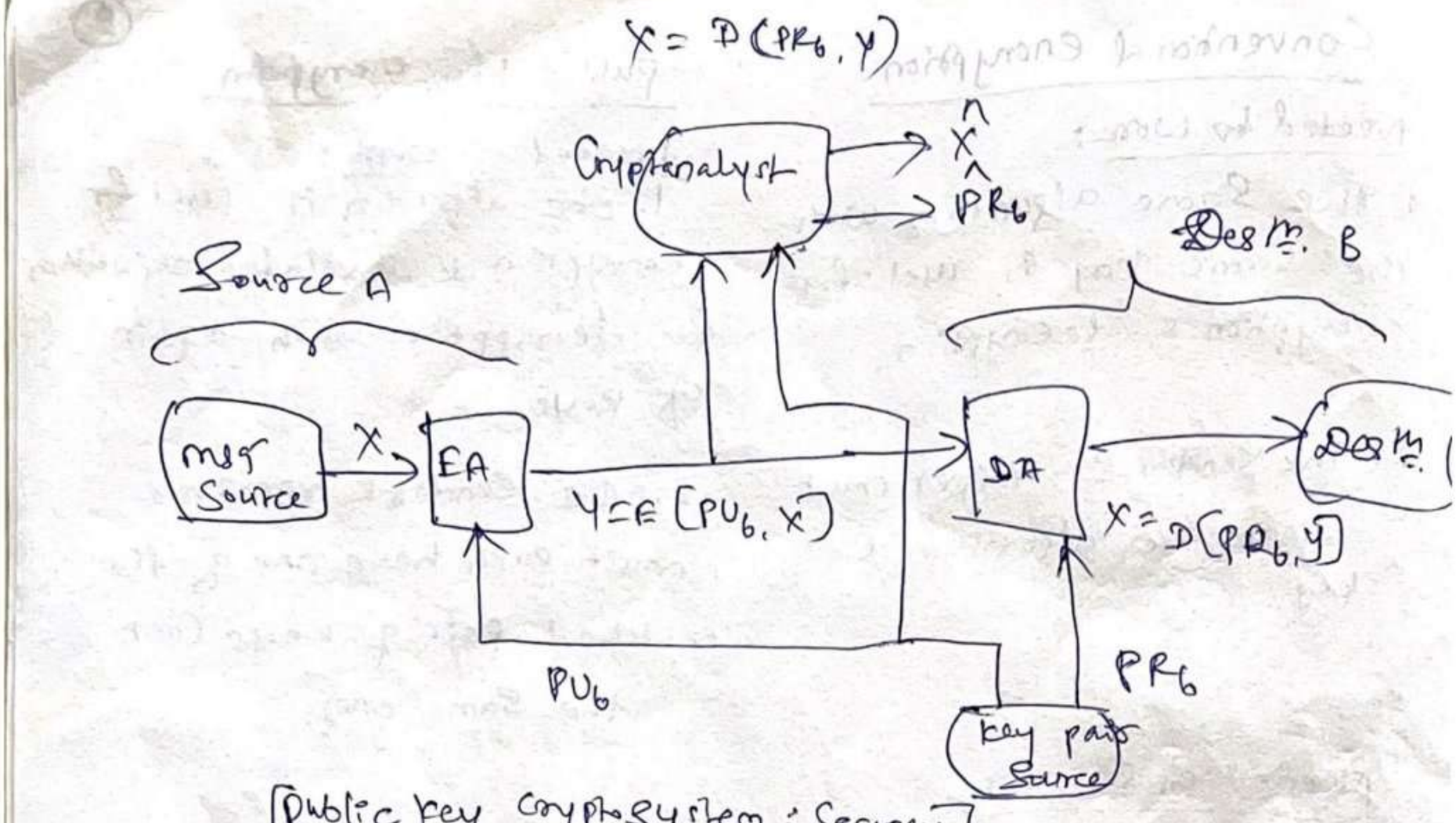
2. The sender & receiver must each have one of the matched pair of keys (not the same one).

1. one of the two keys must be kept secret.

2. It must be impossible or atleast impractical to decipher a message if one of the keys is kept secret.

3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

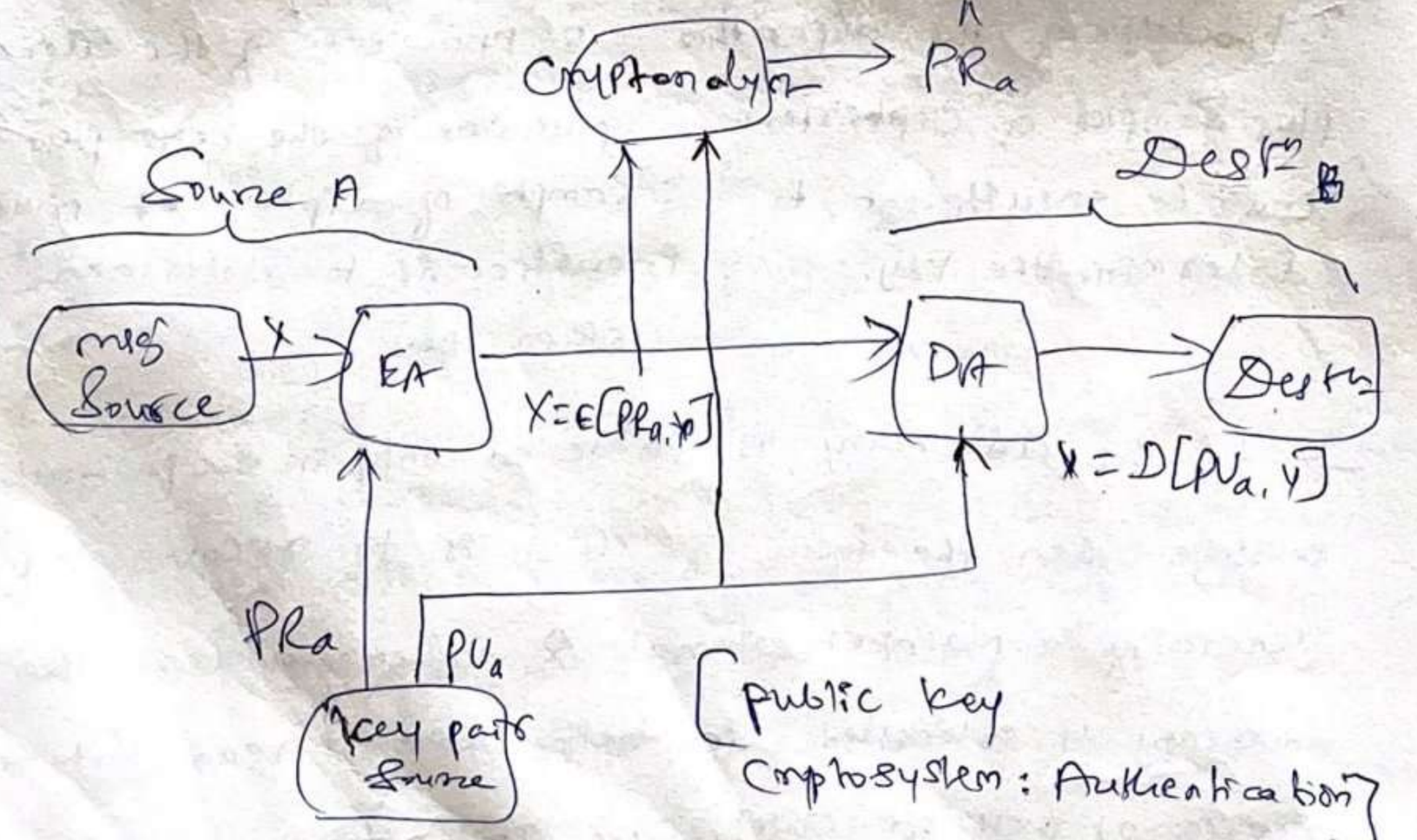
→ If the adversary is interested only in the particular message, then the focus of effort is to recover X by generating a plaintext estimate $\hat{X}$. often however the adversary is interested in being able to read future messages as well, in which case an attempt is made to recover PR_b by generating an estimate $\hat{PR}_b$.



[Public key cryptosystem : Secrecy]

The above figure provides Confidentiality. The figure showing Encryption with private key & below figure show the use of public key encryption to provide authentication: $Y = E(PR_a, X)$

$X = D(PU_a, Y)$



[Public key Cryptosystem : Authentication]

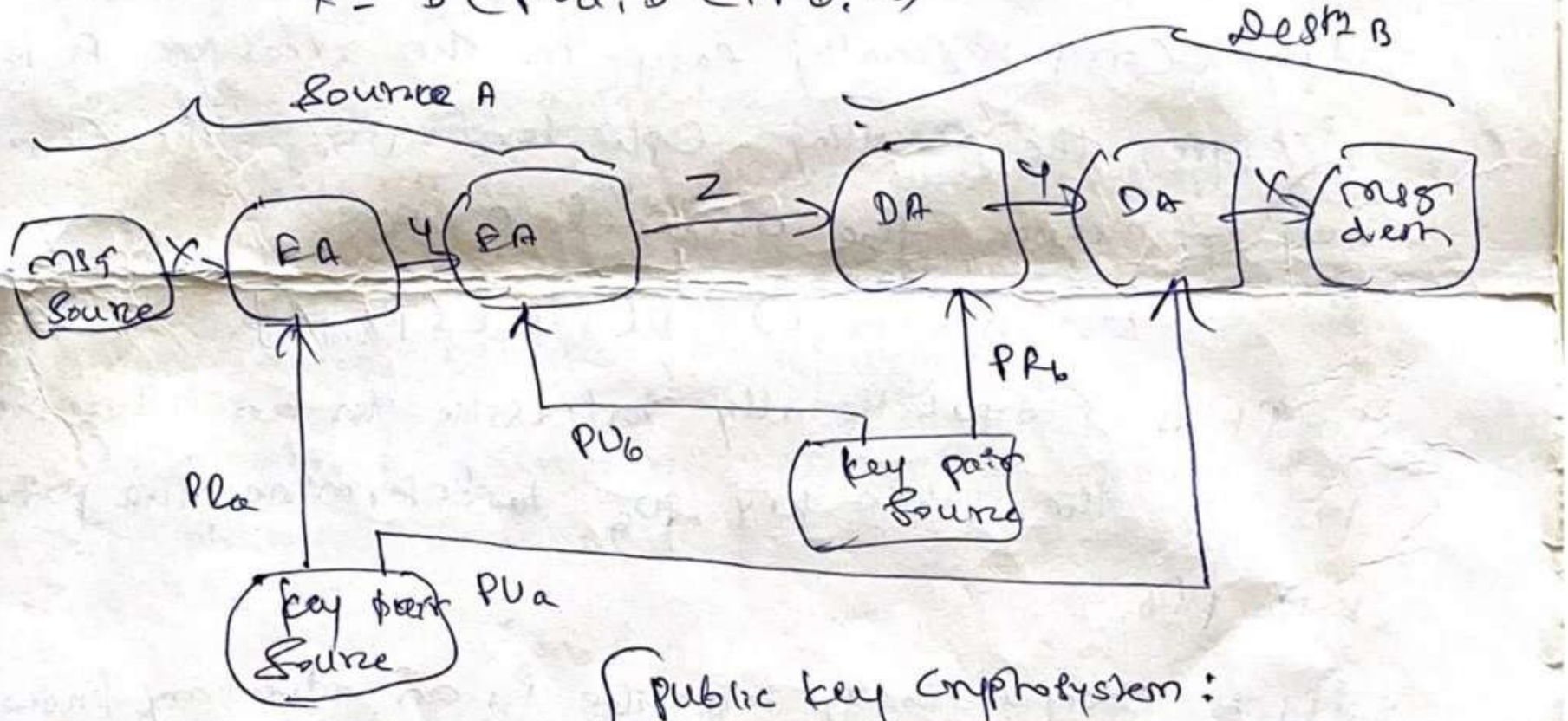
→ In above case, A prepares a message to B and encrypts it using A's private key before transmitting it.

B can decrypt the message using A's public key. Only A could have prepared the message. Therefore the entire encrypted message serves as a digital signature. So its impossible to alter message without access to A's private key. So message is authenticated both in terms of source & data integrity.

→ However it is possible to provide both the authentication function & confidentiality by a double use of the public key scheme.

$$Z = E(PU_b, E(PR_a, X))$$

$$X = D(PU_a, D(PR_b, Z))$$



[Public key Cryptosystem: Authentication & Secrecy]

* Applications for Public-key Cryptosystems - We can classify public key cryptosystems into three categories:

↳ Encryption/Decryption The sender encrypts a msg with the recipient's public key.

↳ Digital Signature - The sender signs a message with its private key. Signing is achieved by a cryptographic algo ~~achieved~~ applied to msg or to a small block of data

Key exchange - two sides cooperate to exchange a session key.

* Requirements for public-key cryptography: Diffie/Hellman did layout the conditions that algorithms must fulfill

1. It is computationally easy for party B to generate a pair (public key PU_b, private key PR_b).

2. It is computationally easy for a sender A, knowing the public key & the message to be encrypted, M to generate corresponding ciphertext:

$$C = E(PU_b, M)$$

3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message

$$M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$$

4. It is computationally infeasible for an adversary knowing the public key, PU_b to determine the private key PR_b.

5. It is computationally infeasible for an adversary, knowing the public key, PU_b & ciphertext C, to recover the original message, M.

We can add a sixth requirement that, although useful, is not necessary for all public-key applications:

6. The two keys can be applied in either order:

$$M = D[PU_b, E(PR_b, M)]$$
$$= D[PR_b, E(PU_b, M)]$$

* Public key cryptanalysis - As with symmetric encryption, a public key encryption scheme is vulnerable to a brute-force attack. The countermeasure is same ; use large keys. However there is a tradeoff to be considered. The key size must be large enough to make brute-force attack impractical but small enough for practical encry^n & Decryp^n.

→ Another form of attack is to find some way to compute the private key given the public key.

* The RSA algorithm

→ Was developed by Ron. Rivest, Adi Shamir & Adleman (Len) at MIT in 1977.

Description of the algorithm

→ RSA makes use of an expression with exponentials. ~~PT is encrypted in blocks with each block is having a~~ binary value less than some number n.

→ Encryption & decryption are of the following form, for some plaintext block M & ciphertext block C

$$C = M^e \bmod n$$

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

Both sender & receiver must know the value of n.

The sender knows the value of e and only the receiver knows the value of d.

public key, $PU = \{e, n\}$ &

private key, $PR = \{d, n\}$

→ For this algorithm to be satisfactory for public-key encryption following requirements must be met.

1. It is possible to find values of e, d & n such that $M^{ed} \bmod n = M$ for all $M < n$.
2. It is relatively easy to calculate $M^e \bmod n$ & $C^d \bmod n$ for all values of $M < n$.
3. It is infeasible to determine d given e & n.

→ for p, q Prime $\phi(pq) = (p-1)(q-1)$

The relationship between e and d can be expressed as

$$ed \bmod \phi(n) = 1$$

This is equivalent to saying

$$ed = 1 \bmod \phi(n)$$

$$d = e^{-1} \bmod \phi(n)$$

That is e and d are multiplicative inverses mod $\phi(n)$.

Example: Euler Totient function

$\phi(0) = 1$

gcd(1,0) = 1

$\phi(2) = 1$

gcd(1,2) = 1, gcd(2,2) = 2

$\phi(3) = 2$

gcd(1,3) = 1 gcd(2,3) = 1, gcd(3,3) = 3

$\phi(5) = 4$

⇒ We are now ready to state the RSA scheme. The ingredients are the following:

p, q, two prime numbers (private, chosen)

$n = pq$ (Public, calculated)

e, with $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$ (public, chosen)

$d = e^{-1} \pmod{\phi(n)}$ (private, calculated)

The private key consists of [d, n] and the public key consists of [e, n]. Suppose that user A has published its public key & that user B wishes to send the message M to A. Then B calculates $C = M^e \bmod n$ & transmits C. On receipt of this ciphertext, user A decrypts by calculating $M = C^d \bmod n$. (5)

Ex 1. Select two prime numbers $p = 17$ & $q = 11$

2. Calculate $n = pq = 17 \times 11 = 187$.

3. Calculate $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$.

4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$; we choose $e = 7$

5. Determine d such that $de = 1 \bmod 160$ & $d < 160$. The correct value is $d = 23$, because $23 \times 7 = 161 = (1 \times 160) + 1$

The resulting keys are

Public key, $PU = \{7, 187\}$

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 $C = 88^7 \bmod 187$

For decryption, we calculate $M = 11^{23} \bmod 187$.

$88^7 \bmod 187 = (88^4 \bmod 187 \times 88^2 \bmod 187 \times 88^1 \bmod 187) \bmod 187$

$= 894432 \bmod 187 = 11$

Computational Aspects: There are actually two issues to consider: encryption/decryption and key generation.

Exponentiation in modular Arithmetic: Both encryption & decryption in RSA involve raising an integer to an

integer power mod n. If the exponentiation is done ov integers and then reduced modulo n, the intermediate va could be gargantum. Fortunately we can make use of a pro of modular arithmetic.

$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$

Thus we can reduce intermediate results modu This makes calculation practical.

→ Another consideration for the efficiency of exponentiation, because with RSA, we are dealing with potentially large expone Suppose we wish to Compute x^{16}. A straightforward approad requires 15 multiplications:

$$x^{16} = x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x$$

However, we can achieve the same final result with only four multiplications if we repeatedly take the square of each partial result, successively forming (x^2, x^4, x^8, x^{16})

→ As another example suppose we wish to calculate $x^{11} \bmod n$ for some integers x & n. Observe that $x^{11} = x^{1+2+8}$ In this case, we compute $x \bmod n$, $x^2 \bmod n$, $x^4 \bmod n$ & $x^8 \bmod n$ & then calculate $[(x \bmod n) \times (x^2 \bmod n) \times (x^8 \bmod n)] \bmod n$.

Efficient operation using the public key — To speed up the operation of the RSA algorithm using the public key a specific choice of e is usually made. The most common choice is 65537 ($2^{16}+1$); two other popular choices are 3 and 17.

However with a very small public key, such as $e=3$, RSA becomes vulnerable to a simple attack. Suppose we have three different RSA users who all use the value $e=3$ but have unique values of n, namely (n_1, n_2, n_3). If user A sends message m to all three users then three ciphertexts are $c_1 = m^3 \bmod n_1$; $c_2 = m^3 \bmod n_2$ & $c_3 = m^3 \bmod n_3$. It is likely that n_1, n_2, n_3 are pairwise relatively prime. Therfore one can use Chinese Remainder Theorem (CRT) to compute $m^3 \bmod (n_1 n_2 n_3)$. By the RSA scale m is less than $n \Rightarrow m^3 < n_1 n_2 n_3$. Accordingly the attacker need only compute the cube root of m^3.

Efficient operation using the private key we cannot choose a small constant value of d for efficient operation. we wish to compute the value $m = c^d \bmod n$

$$V_p = c^d \bmod p \qquad V_q = c^d \bmod q$$

$$\left.\begin{aligned} x_p &= q \times (q^{-1} \bmod p) \\ x_q &= p \times (p^{-1} \bmod q) \end{aligned}\right\} \text{using CRT}$$

The CRT then shows that

$$m = (V_p x_p + V_q x_q) \bmod n$$

Furthermore we can simplify the calculation of V_p & V_q using fermat's theorem, which states that $a^{p-1} = 1 \pmod p$ if p & a are relatively prime. Some thought should convince you that the following are valid.

$$V_p = c^d \bmod p = c^{d \bmod (p-1)} \bmod p$$

$$V_q = c^d \bmod q = c^{d \bmod (q-1)} \bmod q$$

The quantities $d \bmod (p-1)$ & $d \bmod (q-1)$ can be precalculated.

* Key Generation Before the appl^{n} of the public key cryptosystem, each participant must generate a pair of keys. This involves following tasks:

↳ Determining two prime numbers p & q.

↳ Selecting either e or d & calculating each other.

First Consider the selection of p & q. Because the value of n = pq will be known to any potential adversary in order to prevent the discovery of p & q by exhaustive methods these primes must be chosen from sufficiently large set. On the other hand, the method used for finding large primes must be reasonably efficient.

→ A variety of tests for primality have been developed. Almost invariably, the tests are probabilistic. With Miller-Rabin algorithm & most such algorithms the procedures for testing whether a given integer n is prime is to perform some calculation that involves n & a randomly chosen integer a. If n fails the test, then n is not prime. If n passes the test then n may be prime or non prime. If n passes many such tests with many different randomly chosen values for a, then we can have high confidence that n is in fact prime.

In Summary the procedure for picking a prime no. is as follows:

1. Pick an odd integer n at random (eg using PRNG)
2. Pick an integer $a < n$ at random.

3. perform the probabilistic primality test such as Miller-Rabin with a as a parameter. If n fails the test, reject the value n & go to step 1.

4. If n has passed sufficient number of tests, accept n; Otherwise go to step 2.

This is a somewhat tedious procedure. However remember that this process is performed relatively infrequently: only when a new pair (pu, PR) is needed.

The Security of RSA The 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.

↳ Hardware fault-based attack - This involves hardware faults in the processor that is generating digital signatures.

↳ Chosen ciphertext attacks - this type of attack exploits properties of the RSA algorithm.

The factoring problem we can identify three approaches to attacking RSA mathematically

1. Factor n into its two prime factors. This enables calculation of $\phi(n) = (p-1) \times (q-1)$ which in turn enables determination of $d = e^{-1} \pmod{\phi(n)}$.

2. Determine $\phi(n)$ directly. Without first determining p and q. Again this enables determination of $d = e^{-1} \pmod{\phi(n)}$.

3. Determine d directly without first determining $\phi(n)$.

[Refer table 5, page no. 261]

→ Factoring a 1024-bit RSA modulus would be about a thousand times harder than factoring a 256-bit modulus.

→ In addition to specifying the size of n a no. of other constraints have been suggested by researchers. To avoid values of n that may be factored more easily the algorithm inventors suggest the following constraints on p & q.

1. p and q should differ in length by only a few digits. Thus for a 1024-bit key (309 decimal digits) both p & q should be on the order of magnitude of 10^{75} to 10^{100}.
2. Both (p-1) & (q-1) should contain a large prime factor.
3. gcd (p-1) (q-1) should be small.

It has been demonstrated that if $e < n$ & $d < n^{1/4}$ then d can be easily determined.

* Timing attacks Paul Kocher demonstrated that a Snooper can determine a private key by keeping track of how long a Computer takes to decipher messages. These attacks are applicable not just to RSA but to other public key systems.

→ This attack is alarming for two reasons: It comes from a completely unexpected direction & it is a ciphertext-only attack.

→ Suppose the target system uses a modular multiplication fn that is very fast in almost cases but in a few cases takes much more time than an entire average modular exponentiation. The attack proceeds bit-by-bit starting left most bits, bk. Suppose that first j bits are known. For a given ciphertext, the attacker can complete the first j iterations of the for loop.

→ If the bit is set to d ← (d*a) mod n. For a few values of a & d the modular multiplication will be extremely slow, and the attacker knows which these are. Therfore if the observed time to execute the decryption algo is always slow when this particular iteration is slow with a bit 1, then this bit is assumed to be 1, otherwise 0.

Although a the timing attak is a serious threat there are simple countermeasures that can be used, including the following:

↳ Constant exponentiation time - Ensure all exponentiations takes the same amount of time before returning result.

↳ Random delays = Better performance could be achieved by adding a random delay to the exponentiation alg to confuse the timing attack. Some noise need to be add

Blinding - multiply ciphertext by a random no. before performing Exponentiation. This process prevents the attacker from knowing what ciphertext bits are being processed inside computer & therefore prevents bit by bit analysis. essential to the timing attack.

→ RSA Data Security incorporates a blinding feature into some of its products. The private key operation $M = C^d \bmod n$ is implemented as follows:

1. Generate a secret random no. r between 0 & n-1.
2. Compute $C' = C(r^e) \bmod n$ where e is public exponent.
3. Compute $M' = (C')^d \bmod n$
4. Compute $M = M' r^{-1} \bmod n$, where r^{-1} is multiplicative inverse of $r \bmod n$

It can be demonstrated that this is correct result by observing that $r^{ed} \bmod n = r \bmod n$

Fault based Attack - The attack induces faults in Signature Computation by reducing the power to the processor. The faults cause the RSA to produce invalid Signatures, which can be then analyzed to receive private key.

→ It requires that the attacker have physical access to the target machine.

chosen ciphertext attack & optimal asymmetric encryption padding :- CCA is defined as an attack in which the adversary chooses a number of CTs & is then given corresponding plaintexts, decrypted with the target's private key. Thus an adversary could select a PT encrypt it with the targets public key. & decrypt with private key.

→ A simple example of CCA against RSA takes advantage of the following property of RSA:

$E(PU, M_1) \times E(PU, M_2) = E(PU, [M_1 \times M_2])$

we can decrypt $C = M^e \bmod n$ using CCA as:

1. Compute $X = C \times 2^e \bmod n$
2. Submit X as a chosen CT & receive back $Y = X^d \bmod n$

But now note that

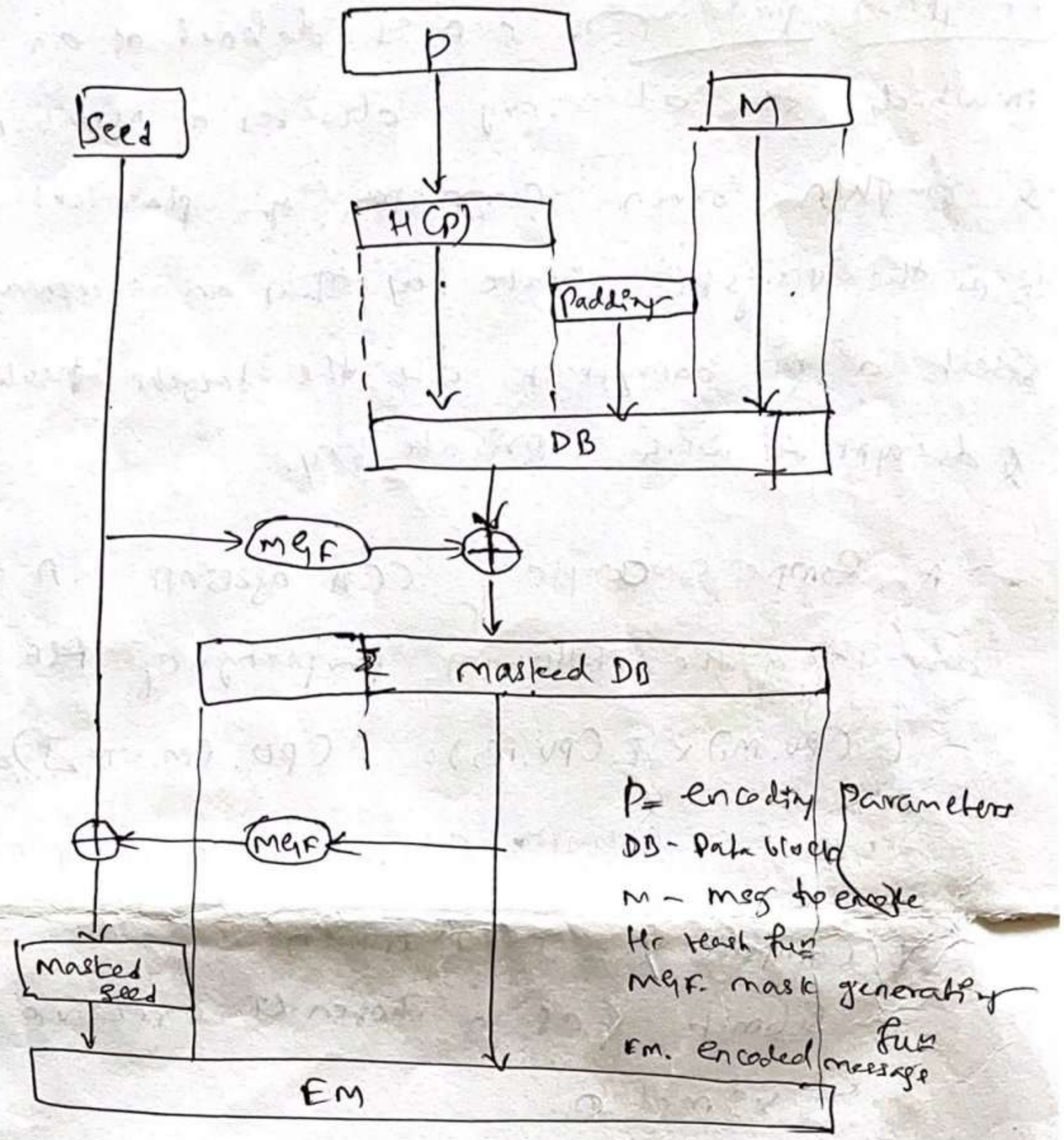
$$X = (C \bmod n) \times (2^e \bmod n)$$
$$= (M^e \bmod n) \times (2^e \bmod n)$$
$$= (2M)^e \bmod n$$

Therefore $Y = (2M) \bmod n$. From this we can deduce M. To overcome this, practical RSA based cryptosystems randomly pad the PT prior to encryption

→ RSA patents, recommends modifying PT using a procedure known as optimal asymmetric encryption padding (OAEP).

(b)

→ The following figure depicts OAEP encryption.



[Encryption using Optimal Asymmetric Encryption padding]

* Other public key cryptosystems

Diffie-Hellman key exchange - The purpose of the algorithm is to enable two users to securely exchange a key that can then be used for subsequent symmetric encryption of messages. The algo. itself is limited to the exchange of secret values.

→ This algorithm depends on discrete logarithm.

→ Discrete logarithm - A primitive root of a prime no. p is one whose powers modulo p generate all the integers from 1 to p-1. That is if a is a primitive root of the prime no. p then the numbers

$a \bmod p, a^2 \bmod p, \dots a^{p-1} \bmod p$

are distinct. & consists of the integers from 1 through p-1 in some permutation.

Eg: 1) P=7, α=3
2) P=11, α=2

For any integer b & a primitive root á of prime number p, we can find a unique exponent i such that

$b \equiv a^i \pmod p$ where $0 \le i \le (p-1)$

The exponent i is referred to as discrete logarithm of b for the base a. mod p. We express this value as $\text{dlog}_{a,p}(b)$.

The algorithm

→ For this scheme, there are two publicly known numbers: a prime number q & an integer α that is a primitive root of q.

Alice

Bob

Alice & Bob share a prime no. q & an integer α such that $\alpha < q$ & α is a primitive root of q

Alice generates a private key X_A such that $X_A < q$

Bob generates a private key X_B such that $X_B < q$

Alice Calculates a public key $Y_A = \alpha^{X_A} \bmod q$

Bob Calculates a public key $Y_B = \alpha^{X_B} \bmod q$

Alice receives Bob's public key Y_B in plaintext

Bob receives Alice's public key Y_A in plaintext

Alice Calculates shared secret key $K = (Y_B)^{X_A} \bmod q$

Bob Calculates shared secret key $K = (Y_A)^{X_B} \bmod q$

[The Diffie Hellman key exchange]

Eg: $q = 71$, $\alpha = 7$, $X_A = 5$, $X_B = 12$, $K = ?$

By the rules of modular Arithmetic

$$K = (Y_B)^{X_A} \bmod q$$
$$= (\alpha^{X_B} \bmod q)^{X_A} \bmod q$$
$$= (\alpha^{X_B})^{X_A} \bmod q$$
$$= \alpha^{X_B X_A} \bmod q$$
$$= (\alpha^{X_A})^{X_B} \bmod q$$
$$= (\alpha^{X_A} \bmod q)^{X_B} \bmod q$$
$$= (Y_A)^{X_B} \bmod q$$

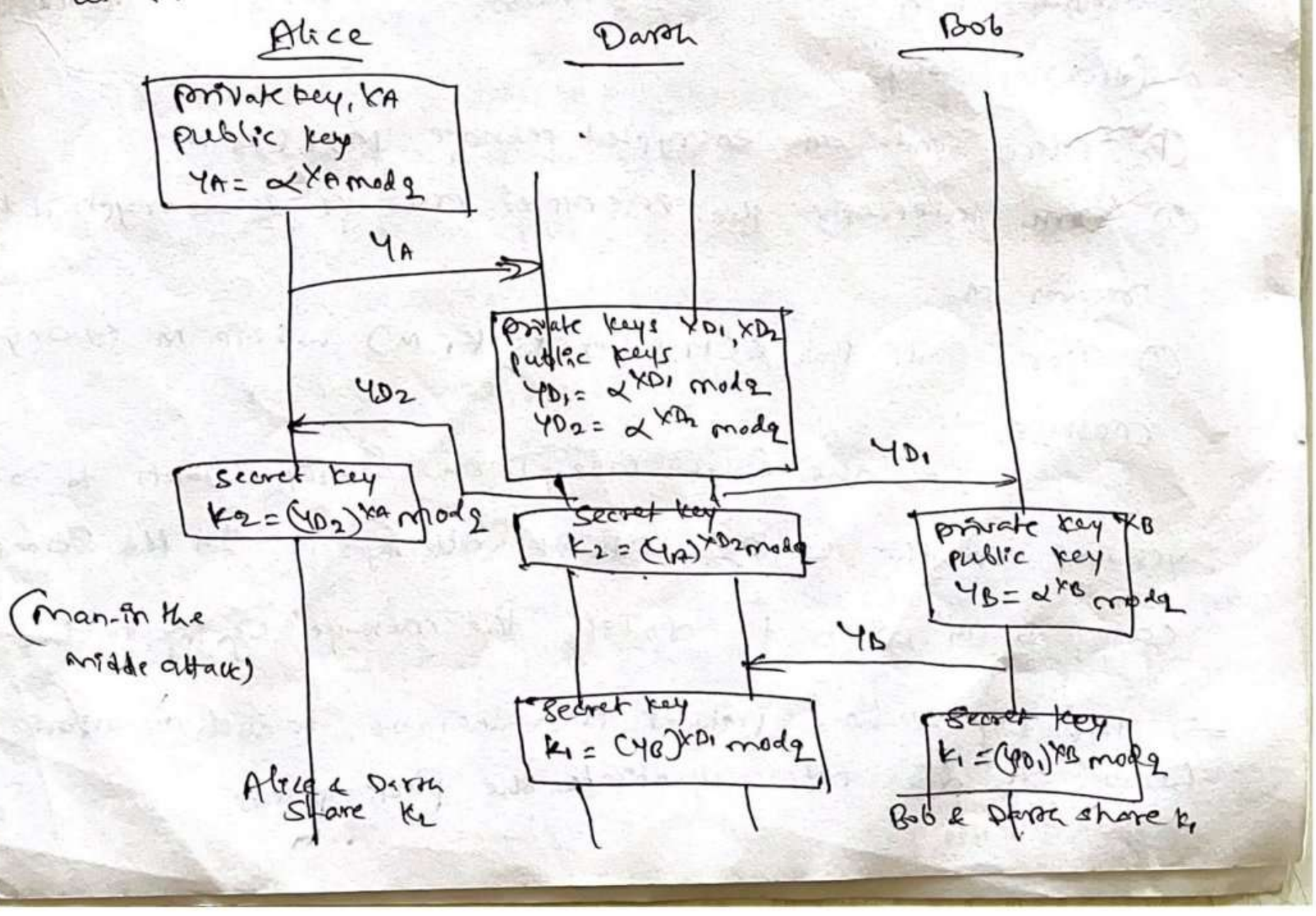
The result is that the two sides have exchanged a secret value.

Key exchange protocol The previous figure shows a simple protocol that makes use of the Diffie-Hellman Calculation. Suppose that user A wishes to set up a connection with user B and use a secret key to encrypt message on that conn.

⇒ User A can generate a one time private key X_A, Calculate Y_A and send that to user B. User B responds by generating a private value X_B, Calculating Y_B & sending Y_B to user A. Both users can now calculate the key.

The necessary public values q and α would need to be known ahead of time. Alternatively, user A could pick values for q & α and include those in the first message.

* Man-in the Middle Attack Suppose Alice & Bob wish to exchange keys and Darth is the adversary. The attack proceeds as follows:



(21)

$k = (r_A)^{X_A}$

(11)

① Darth prepares for the attack by generating two random private keys X_{D_1} and X_{D_2} and then computing the corresponding public keys Y_{D_1} & Y_{D_2}.

② Alice transmits Y_A to Bob.

③ Darth intercepts Y_A & transmits Y_{D_1} to Bob. Darth also calculates $K_2 = (Y_A)^{X_{D_2}} \bmod q$.

④ Bob receives Y_{D_1} and calculates $K_1 = (Y_{D_1})^{X_B} \bmod q$.

⑤ Bob transmits Y_B to Alice.

⑥ Darth intercepts Y_B & transmits Y_{D_2} to alice. Darth calculates $K_1 = (Y_B)^{X_{D_1}} \bmod q$.

⑦ Alice receives Y_{D_2} and calculates $K_2 = (Y_{D_2})^{X_A} \bmod q$.

At this point, Bob & Alice think that they share a Secret key. but instead Bob and Darth share secret key K_1 & Alice & Darth share secret key K_2. All future communication betn Bob & Alice is compromised in the following way.

① Alice Sends an encrypted message $M: E(K_2, M)$

② Darth intercepts the encrypted message & decrypts it to recover M.

③ Darth Sends Bob $E(K_1, M)$ or $E(K_1, M')$ where M' is any message.

In the first case, Darth Simply wants to eavesdrop on the Comm^n without altering it. In the Second case, Darth wants to modify the message going to Bob.

⇒ The key exchange protocol is vulnerable to Such an attack because it does not authenticate the participants.

* ELGamal Cryptographic System

→ In 1984, T. Elgamal announced a public-key scheme based on discrete logarithms closely related to the Diffie Hellman technique.

→ As with Diffie-Hellman, the global elements of Elgamal are a prime number q and α, which is primitive root of q.

Q. User A generates a private/public key pair as follows:

1. Generate a random integer X_A, such that $1 < X_A < q-1$.
2. Compute $Y_A = \alpha^{X_A} \bmod q$.
3. A's private key is X_A and A's public key is $\{q, \alpha, Y_A\}$.

Any user B that has access to A's public key can encrypt a message as follows:

1. Represent the message as an integer M in the range $0 \le M \le q-1$. Longer messages are sent as a sequence of blocks with each block being an integer less than q.
2. Choose a random integer k such that $1 \le k \le q-1$.
3. Compute a one-time key $K = (Y_A)^k \bmod q$.
4. Encrypt M as the pair of integers (C_1, C_2) where

$$C_1 = \alpha^k \bmod q; \quad C_2 = KM \bmod q$$

User A recovers the plaintext as follows:

1. Recover the key by computing $K = (C_1)^{X_A} \bmod q$
2. Compute $M = (C_2 K^{-1}) \bmod q$.

Let us demonstrate why Elgamal scheme works. First we show that how k is recovered by the decryption process

$K = (Y_A)^k \bmod q$ // K is defined during the encryption

$K = (\alpha^{X_A} \bmod q)^k \bmod q$ // Substitute using $Y_A = \alpha^{X_A} \bmod q$

$K = \alpha^{kX_A} \bmod q$ // by the rules of mod Arithmetic

$K = (C_1)^{X_B} \bmod q$ Substitute Using $C_1 = \alpha^k \bmod q$

Next using k we recover the plaintext as

$C_2 = KM \bmod q$

$(C_2 K^{-1}) \bmod q = KMK^{-1} \bmod q = M \bmod q = M$

Global public elements

q	prime no.
α	$\alpha < q$ and α is a primitive root of q

Key generation by Alice

Select private X_A	$X_A < q-1$
Calculate Y_A	$Y_A = \alpha^{X_A} \bmod q$
Public key	$\{q, \alpha, Y_A\}$
Private key	X_A

Encryption by Bob with Alice's public key

Plaintext	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext	$\{C_1, C_2\}$

Decryption by Alice with Alice's private key

Ciphertext	$\{C_1, C_2\}$
Calculate K	$K = (C_1)^{X_A} \bmod q$
Plaintext	$M = (C_2 K^{-1}) \bmod q$

[The Elgamal Cryptosystem]

Module-3

* **Elliptic curve Arithmetic** most of the products and standards use public key cryptography for encryption and digital signatures use RSA. The key length for secure RSA use has increased over recent years and this has put a heavier processing load on applications using RSA. This burden has ramifications, especially for electronic commerce sites that conduct large no. of secure transactions. A competing system challenges RSA: Elliptic curve Cryptography (ECC).

⇒ The principal attraction of ECC, compared to RSA is that it appears to offer equal security for a far smaller key size, thereby reducing processing overhead, on the other hand the confidence level in ECC is not yet as high as that in RSA.

Abelian Groups

→ An abelian group G, sometimes denoted by {G, ·} is a set of elements with a binary operation, denoted by ·, that associates to each ordered pair (a.b) of elements in G an element (a.b) in G, such that the following axioms are obeyed: (Fundamental assumptions)

(A1) **Closure**: If a and b belongs to G then a.b is also in G.

(A2) **Associative**: $a.(b.c) = (a.b).c$ for all a.b.c in G.

(A3) **Identity element**: There is an element e in G such that

$a.e = e.a = a$ for all a in G.

(A4) Inversed element : for each a in G there is an element a' in G such that $a.a' = a'.a = e$.

(A5) Commutative : $a.b = b.a$ for all a, b in G.

[The operator . is generic and can refer to addition, multiplication, or some other mathematical operation].

⇒ A no. of public key ciphers are based on the use of an abelian group. For eg, Diffie Hellman key exchange involves multiplying pairs of non-zero integers modulo a prime number q keys are generated by exponentiation over the group, with exponentiation defined as repeated multiplication.

for eg $a^k \bmod q = \underbrace{(a \times a \times \ldots \times a)}_{k \text{ times}} \bmod q$ To attack Diffie-Hellman, the attacker must determine k given a and a^k; this is the discrete logarithm problem.

→ For elliptic Curve Cryptography an operation over elliptic curves, called addition is used. multiplication is defined by repeated addition. for example,

$$a \times k = \underbrace{(a + a + \ldots + a)}_{k \text{ times}}$$

Here cryptanalysis involves determining k given a and $(a \times k)$.

⇒ An elliptic Curve is defined by an equation in two variables with coefficients.

* Elliptic curves over real numbers

Elliptic curves are not ellipses. They are so named because they are described by cubic equations, similar to those used for calculating the circumference of an ellipse. In general, cubic equations for elliptic curves take the following form, known as weierstrass equation:

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

where a, b, c, d, e are real numbers and x and y take on values in the real numbers. for our purpose, it is sufficient to limit ourselves to equations of the form

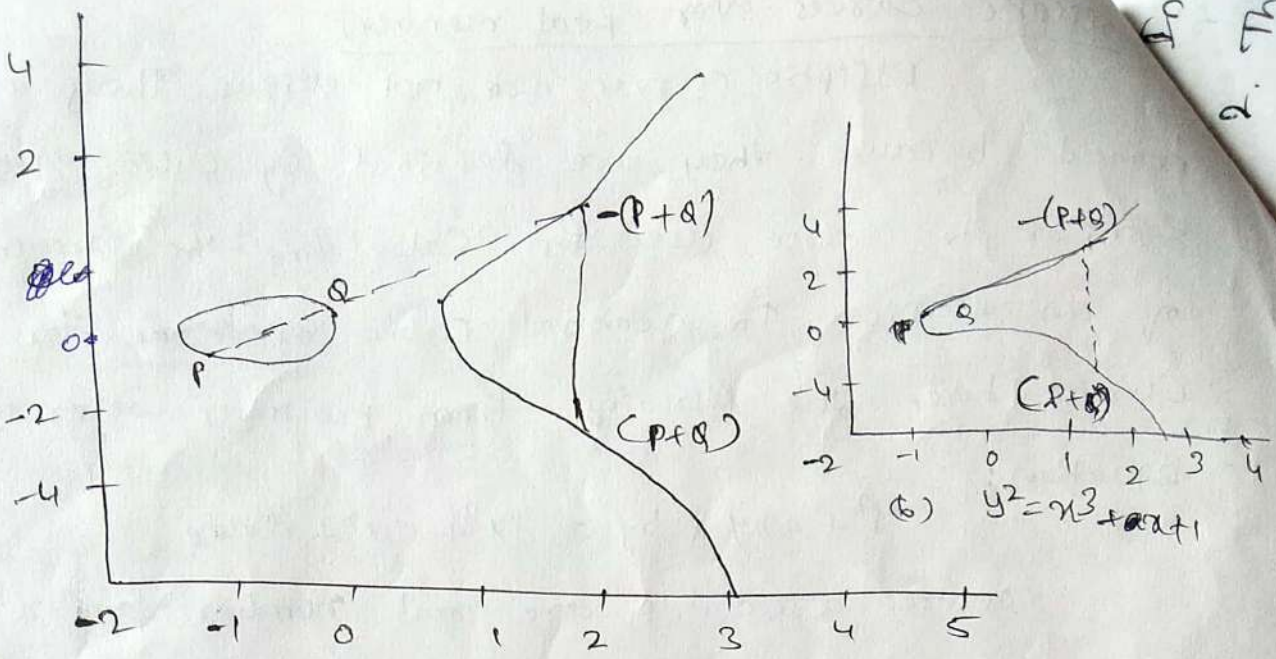
$$y^2 = x^3 + ax + b \quad \text{— (1)}$$

Such equations are said to be cubic or of degree 3, because the highest exponent they contain is a 3. To plot such a curve, we need to compute

$$y = \sqrt{x^3 + ax + b}$$

For given values of a & b the plot consists of positive & negative values of y for each value of x. Thus, each curve is symmetric about $y = 0$.

→ Now, consider the set of points $E(a, b)$ consisting of all of the points (x, y) that satisfy equation (1) together with the element $\mathcal{O}$. Using a different value of the pair (a, b) results in a different set $E(a, b)$. Using this terminology, the two curves in below figure shows the sets $E(-1, 0)$ & $E(1, 1)$ respectively. (figure shows two examples of elliptic curves.



(a) $y^2 = x^3 - x$ (b) $y^2 = x^3 + ax + 1$

Fig 10.4 (7th edition)

Refer Figure 4.6 (Page no. 287)

Geometric Description of Addition It can be shown that a group can be defined based on the set $E(a,b)$ for specific values of a and b in eqn (1) provided the following condition is met:

$$4a^3 + 27b^2 \neq 0 \quad \text{— (2)}$$

To define the group, we must define an operation called addition and denoted by +, for the set $E(a,b)$ where a & b satisfy equation (2).

→ In geometric terms the rules for the addition can be stated as follows: If three points on an elliptic curve lie on a straight line their sum is O. From this definition, we can define the rules of addition over an elliptic curve.

1. O serves as the additive identity. Thus $O = -O$; for any point P on the elliptic curve, $P + O = P$.

In what follows, we assume $P \neq 0$ and $Q \neq 0$.

2. The negative of a point P is the point with the same x co-ordinate but the negative of the y co-ordinate; i.e if $P = (x, y)$ then $-P = (x, -y)$. Note that these two points can be joined by a vertical line. Note that $P + (-P) = P - P = 0$.

3. To add two points P and Q with different x co-ordinates draw a straight line between them & find the third point of intersection R. It is easily seen that there is unique point R that is the point of intersection.

To form a group structure, we need to define addition on these three points: $P + Q = +R$. That is we define $P + Q$ to be the mirror image w.r.t x-axis of the third point of intersection.

4. The geometric interpretation of the preceding item also applies to two points, P & $-P$ with the same x co-ordinate. The points are joined by a vertical line, which can be viewed as also intersecting the curve at the infinity point.

5. To double a point Q, draw the tangent line and find the other point of intersection S. Then $Q + Q = 2Q = -S$.

With the preceding list of rules, it can be shown that the set $E(a, b)$ is an abelian group.

* Algebraic Description of Addition for two distinct points $P = (x_p, y_p)$ & $Q = (x_q, y_q)$ that are not negatives of each other

the slope of the line λ that joins them is $\Delta = \frac{y_Q - y_P}{x_Q - x_P}$. There is exactly one other point where λ intersects the elliptic curve and that is the negative of the sum of P & Q.

After some algebraic manipulation, we can express the sum $R = P + Q$ as

$$x_R = \Delta^2 - x_P - x_Q \quad \text{---(3)}$$
$$y_R = -y_P + \Delta(x_P - x_R)$$

We also need to be able to add a point to itself. $P + P = 2P = R$. When $y_P \neq 0$, the expressions are

$$x_R = \left(\frac{3x_P^2 + a}{2y_P}\right)^2 - 2x_P$$
$$y_R = \left(\frac{3x_P^2 + a}{2y_P}\right)(x_P - x_R) - y_P \quad \text{---(4)}$$

* Elliptic Curves over Z_p Two families of elliptic curves are used in cryptographic applns: prime curves over Z_p & binary curves over GF(2^m).

→ For a prime curve over Z_p, we use a cubic equation in which the variables & coefficients all take on values in the set of integers from 0 through $p-1$ & in which calculations are performed modulo p.

→ For elliptic curves over Z_p, as with real numbers, we limit ourselves to equations of the form of eq (3), but in this case with co-efficient & variables limited to Z_p:

$$y^2 \bmod p = (x^3 + ax + b) \bmod p \quad \text{---(5)}$$

For eg, eq(5) is satisfied for $a=1, b=1, x=9, y=7, p=23$

$$7^2 \bmod 23 = (9^3 + 9 + 1) \bmod 23$$
$$49 \bmod 23 = 739 \bmod 23$$
$$3 = 3$$

Now Consider the set $E_p(a,b)$ consisting of all pairs of integers (x,y) that satisfy eqn (5) together with a point at infinity O. The co-efficients a & b & the variables x and y are all elements of Z_p.

→ for the set $E_{23}(1,1)$ we are only interested in the non negative integers in the quadrant from (0,0) through $(p-1, p-1)$ that satisfy the eqn. mod p. Table below lists the points (other than O) that are part of $E_{23}(1,1)$

(0,1)	(6,4)	(12,19)
(0,22)	(6,19)	(13,7)
(1,7)	(7,11)	(13,16)
(1,16)	(7,12)	(17,3)
(3,10)	(9,7)	(17,20)
(3,13)	(9,16)	(18,3)
(4,0)	(11,3)	(18,20)
(5,4)	(11,20)	(19,5)
(5,19)	(12,4)	(19,18)

[Points (other than O) on the elliptic curve $E_{23}(1,1)$]

→ If we draw figure, there is symmetry at about 11.5.

It can be shown that a finite abelian group can be defined based on the set $E_p(a,b)$ provided that $(x^3 + ax + b) \bmod p$ has no repeated factors. This is equivalent to the condition

$$(4a^3 + 27b^2) \bmod p \neq 0 \bmod p \quad — (6)$$

→ The rules for addition over $E_p(a,b)$ correspond to the algebraic technique described for elliptic curves defined over real numbers. For all points $P, Q \in E_p(a,b)$:

1. $P + O = P$
2. If $P = (x_p, y_p)$ then $P + (x_p, -y_p) = O$. The point $(x_p, -y_p)$ is the negative of P denoted as $-P$.

For eg in $E_{23}(1,1)$ for $P = (13,7)$ we have $-P = (13,-7)$ *
But $-7 \bmod 23 = 16$, therfore $-P = (13,16)$ which is also in $E_{23}(1,1)$.

3. If $P = (x_P, y_P)$ & $Q = (x_Q, y_Q)$ with $P \neq -Q$, then $R = P + Q = (x_R, y_R)$ is determined by the following rules:

$$x_R = (\lambda^2 - x_P - x_Q) \bmod p$$

$$y_R = (\lambda(x_P - x_R) - y_P) \bmod p$$

where

$$\lambda = \begin{cases} \left(\frac{y_Q - y_P}{x_Q - x_P}\right) \bmod p & \text{if } P \neq Q \\ \left(\frac{3x_P^2 + a}{2y_P}\right) \bmod p & \text{if } P = Q \end{cases}$$

4. Multiplication is defined as repeated addition; for eg
$4P = P + P + P + P$

for eg let $P = (3,10)$ & $Q = (9,7)$ in $E_{23}(1,1)$ then

$$\lambda = \left(\frac{7-10}{9-3}\right) \bmod 23 = \left(\frac{-3}{6}\right) \bmod 23 = \left(\frac{-1}{2}\right) \bmod 23 = 11$$

$$x_R = (11^2 - 3 - 9) \bmod 23 = 109 \bmod 23 = 17$$

$$y_R = (11(3-17) - 10) \bmod 23 = -164 \bmod 23 = 20$$

So $P + Q = (17,20)$. To find $2P$

$$\lambda = \left(\frac{3(3)^2 + 1}{2 \times 10}\right) \bmod 23 = \left(\frac{5}{20}\right) \bmod 23 = \left(\frac{1}{4}\right) \bmod 23 = 6$$

The last step in the preceding eqn involves taking the multiplicative inverse of 4 in Z_{23}. This can be done using the extended Euclidean algorithm. To confirm, note that $(6 \times 4) \bmod 23 = 24 \bmod 23 = 1$

$$x_R = (6^2 - 3 - 3) \bmod 23 = 30 \bmod 23 = 7$$

$$y_R = (6(3-7) - 10) \bmod 23 = (-34) \bmod 23 = 12$$

& $2P = (7,12)$.

6, 11, 15, 16, 20, 25, 28, 34, 40, 46, 49.

Pseudorandom Number Generation based on an asymmetric cipher

⇒ Pseudo random number generator refers to an algorithm that uses mathematical formulas to produce sequences of random numbers. approximating the properties of random numbers. (Deterministic & efficient)

→ Because a symmetric block cipher produces an apparently random output, it can serve as the basis of a pseudor-andom number generator (PRNG). Similarly an asymmetric encryption algorithm produces apparently random output & can be used to generate open-ended PRNG bit streams. Rather the asymmetric approach is used for creating a pseudorandom key for generating a short pseudorandom bit sequence.

PRNG based on RSA - for a sufficient key length, the RSA algorithm is considered secure & is a good candidate to form the basis of a PRNG. Such a PRNG known as Micali-Schnorr PRNG is recommended in the ANSI standard X9.82 & in the ISO standard 18031 [Random Bit Generation]

[Random Number generation]

→ The PRNG is illustrated in below figure. This PRNG has much the same structure as the output feedback (OFB) mode used as a PRNG. A portion of the output is fed back to the next iteration of the encryption algorithm & the remainder of the o/p is used as pseudorandom bits. The motivation for this seperation of the o/p into two distinct parts

is so that the pseudorandom bits from one stage don't provide input to the next stage. This separation should contribute to forward unpredictability.

We can define the PRNG as follows.

Setup — Select p, q primes; $n = pq$; $\phi(n) = (p-1)(q-1)$. Select e such that $\gcd(e, \phi(n)) = 1$. In addition let $N = \lfloor \log_2 n \rfloor + 1$ (the bit length of n). Select r, k such that $r + k = N$.

Seed — Select a random seed x_0 of bit length r.

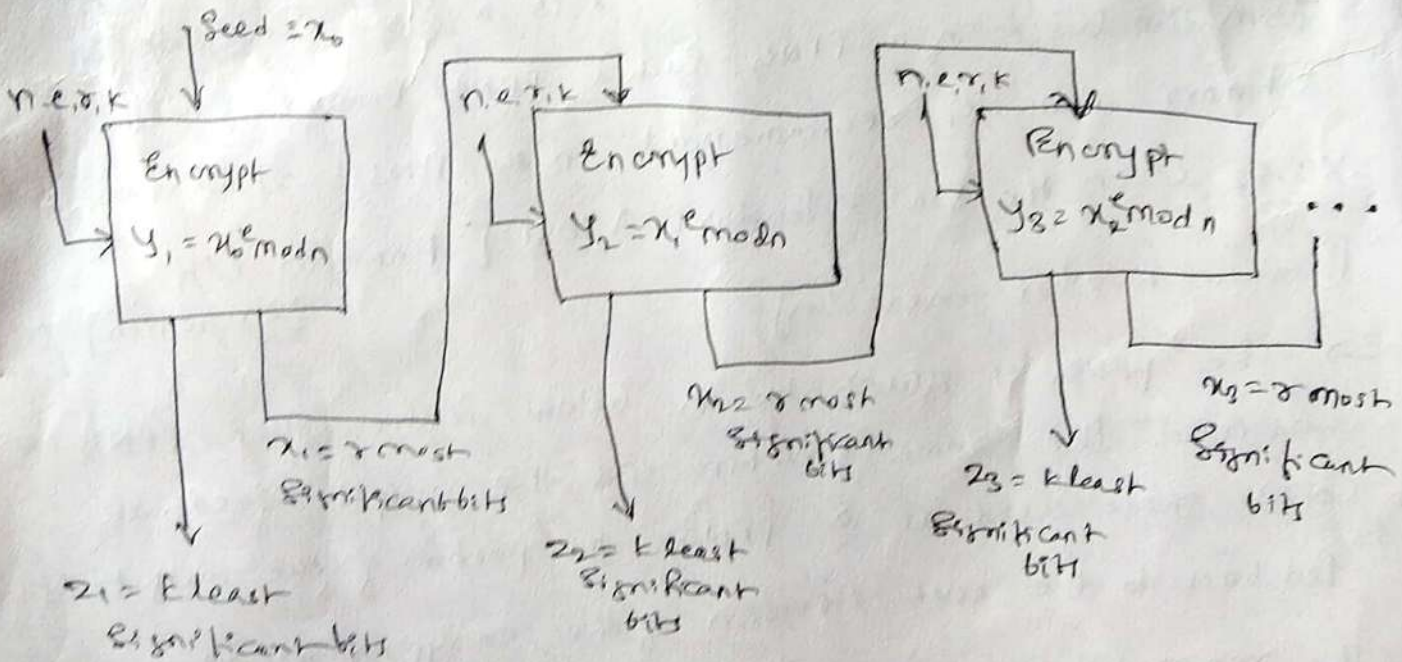
Generate — Generate a pseudorandom sequence of length km using the loop for i from 1 to m do

$$y_i = x_{i-1}^e \bmod n$$

x_i = r most significant bits of y_i

z_i = k least significant bits of y_i

Output — The o/p sequence is $z_1 \| z_2 \| \ldots \| z_m$.



(Figure: Micali - Schnorr pseudorandom bit Generator)

The parameters n, r, e, k are selected to satisfy the following six requirements:

1. $n = pq$ — n is chosen as the product of two primes to have the cryptographic strength required of RSA.
2. $1 < e < \phi(n)$; $\gcd(e, \phi(n)) = 1$ — Ensures that the mapping $s \to s^e \bmod n$ is 1 to 1.
3. $re \geq 2rN$ — Ensures that the exponentiation requires a full modular reduction.
4. $r \geq 2$ strength — Protects against a cryptographic attacks.
5. k, r are multiples of 8 — An implementation convenience.
6. $k \geq 8$; $r + k = N$ — All bits are used.

→ The variable strength is defined as follows: A number associated with the amount of work (i.e the no. of operations) required to break a cryptographic algorithm or system; a security strength is specified in bits & is a specific value from the set (112, 128, 192, 256). for this recommendation. The amount of work needed is 2^{strength}.

→ There is clearly trade-off between r and k. Because

[An adv or improvement that necessitates loss or degradation]

Eg Graphics vs battery life

RSA is computationally intensive compared to a block cipher we would like to generate as many pseudorandom bits per iteration as possible & therefore would like a large value of k.

For ex. if $e=3$ & $N=1024$, then we have the inequality $3r > 1024$, yielding a minimum required size for r of 683 bits. For r set to that size, $k=341$ bits are generated for each exponentiation. In this case, each exponentiation requires only one modular squaring of a 683-bit number & one modular multiplication. i.e we need only Calculate $(x_i \times (x_i^2 \bmod n)) \bmod n$.

* PRNG based on Elliptic Curve Cryptography Let P & Q be two known points on a given elliptic curve. The seed of the DEC PRNG is a random integer $s_0 \in \{0, 1, \ldots \#E(GF(p)) - 1\}$ where $\#E(GF(p))$ denotes the no. of points on the curve. Let x denote a fun that gives the x-coordinate of a point of the curve. Let $lsb_i(s)$ denote i least significant bits of an integer s. The DEC PRNG transforms the seed into the pseudorandom sequence of length $240k$, $k>0$ as follows.

for $i=1$ to k do
  Set $s_i \leftarrow x(s_{i-1} P)$
  Set $r_i \leftarrow lsb_{240}(x(s_i Q))$
end for
Return $r_1, \ldots, r_k$

Key management & Distribution

Symmetric key distribution using Symmetric Encryption

The strength of any cryptographic system rests with the key distribution technique, a term that refers to the means of delivering a key to two parties who wish to exchange data without allowing others to see the key.

For two parties A & B key distribution can be achieved in a no. of ways, as follows:

1. A can select a key & physically deliver it to B.
2. A third party can select the key & physically deliver it to A and B.
3. If A & B have previously & recently used a key, one party can transmit the new key to the other encrypted using the old key.
4. If A & B each has an encrypted conn. to a third party C, C can deliver a key on the encrypted links to A & B.

options 1 & 2 call for manual delivery of a key. For link encryption, this is a reasonable requirement because each link encryption device is going to be exchanging data only with its partner on the other end of the link. However for end-to-end encryption over a n/w, manual delivery is awkward. In a distributed system, any given host or terminal need to engage in exchanges with many other hosts or terminal may need to engage over time. Thus, each device needs a no. of keys, supplied dynamically.

The problem is especially difficult in a wide area distributed system.

The scale of the problem depends on the no. of communicating pairs that must be supported. If end-to-end encryption is done at a n/w or IP level, then a key is needed for each pair of hosts on the n/w that wish to communicate. Thus, if there are N hosts, the no. of required keys is $[N(N-1)]/2$. If encryption is done at the appln level then a key is needed for every pair of users or processes that require communication.

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

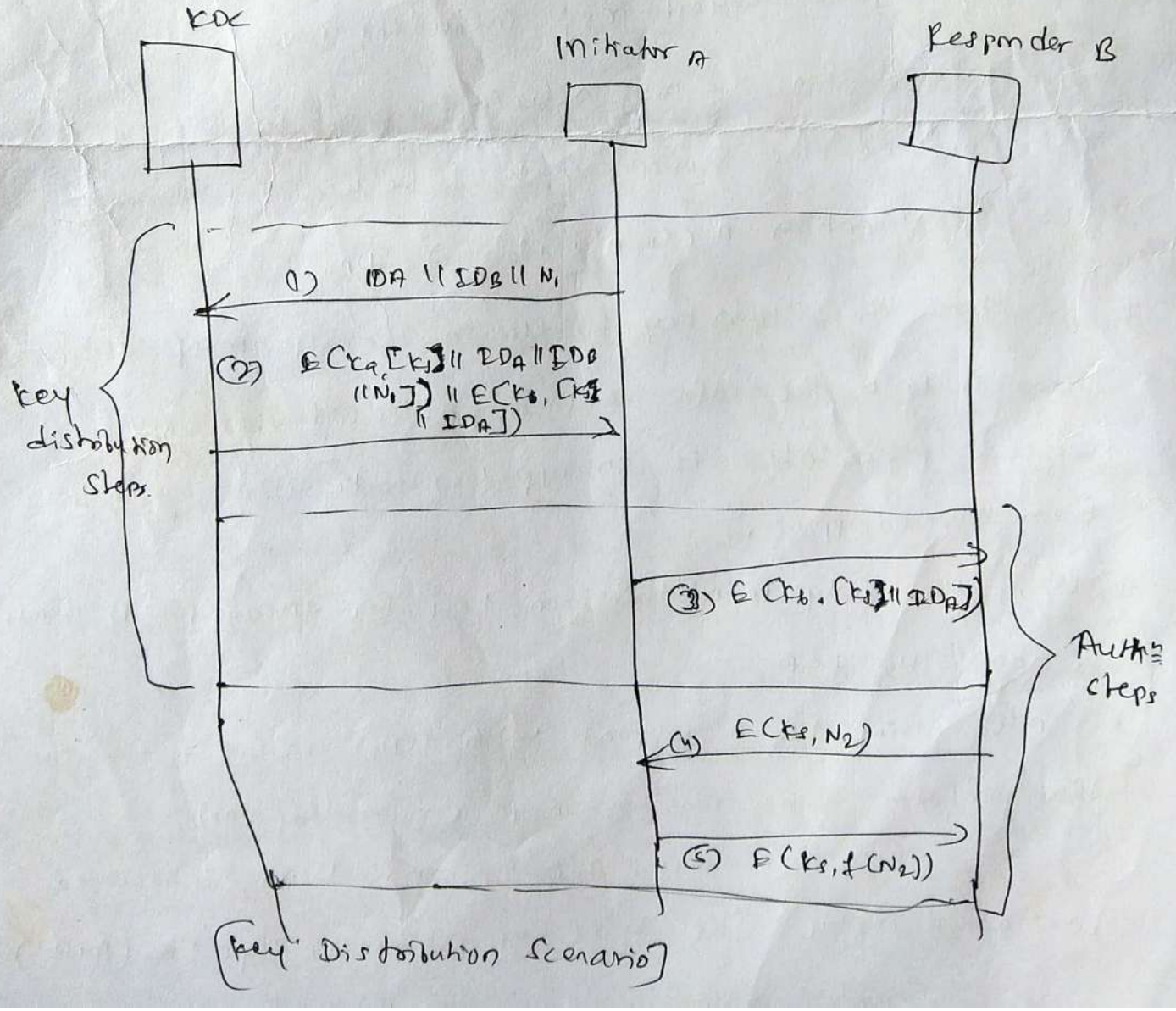
→ For end-to-end encryption, some variation on option 4 has been widely adopted. In this scheme, a key distribution center is responsible for distributing keys to pairs of users (hosts, processes, applns) as needed.

⇒ 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. Comm^{n} between end systems is encrypted using a temporary key, often referred to as a session key. Each session key is obtained from the key distribution center used for end-user communication. Accordingly, session keys are transmitted in encrypted form, using a master key that is shared by the key distribution center & an end system or user.

A key distribution Scenario

The key distribution concept can be deployed in a no.of ways. A typical scenario is illustrated in the figure below. The scenario assumes that each user shares a unique master key with the KDC (key distribution Center)

→ Let us assume that user A wishes to establish a logical conn with B and requires a one-time session key to protect the data transmitted over the conn. A has master key K_a known only to itself & the KDC; similarly B shares the master key K_b with the KDC.



The following steps occur:

1. A issues a request to the KDC for a session key to protect logical conn to B. The msg includes the identity of A & B & a unique identifier, N_1 for this transaction, which we refer to as a nonce. The nonce may be a timestamp, counter or a random no; the minimum requirement is that it differs with each request.

2. The KDC responds with a msg encrypted using K_a. Thus A is the only one who can successfully read the message & A knows that it originated at KDC. The msg includes (intended for A)
   - ↳ The one-time session key K_s to be used for the session.
   - ↳ The original request msg including nonce.

   In addition, the msg includes intended for B:
   - ↳ The one-time session key, K_s to be used for session
   - ↳ An identifier A (eg n/w address), ID_A

3. A stores the session key for use in the upcoming session & forwards to B the info that originated at the KDC for B. Because this info is encrypted with K_b, it is protected from eavesdropping.

4. Using the newly minted session key for encryption B sends a nonce, N_2 to A.

5. Also using K_s, A responds with $f(N_2)$ where f is a fun that performs some transform on N_2 (eg adding one).

These steps assures that the original msg is received was not a replay. Step 1 to 3 (key distribution) steps 3 to 5 (Auth)

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

If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a global KDC. This hierarchical concept can be extended to three or even more layers depending on the size of the user population & geographic scope of the internetwork.

→ A hierarchical scheme minimizes the effort involved in master key distribution. Also this scheme limits faulty KDC to its local area only.

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

→ For con oriented protocols one obvious choice is to use the same session key for the length of time that the con is open using a new session key for each new session.

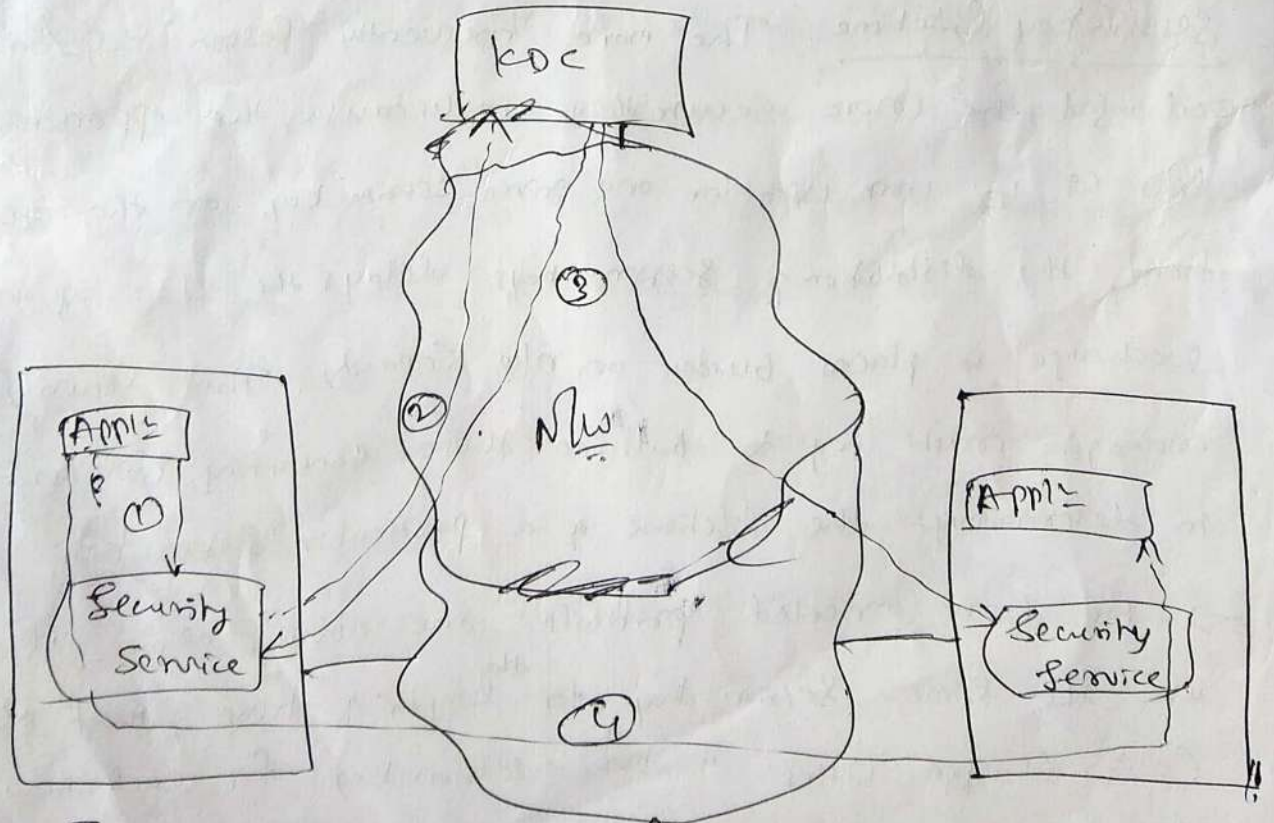
⇒ for a connectionless protocol such as transaction-oriented protocol there is no explicit connection initiation or termination. Thus it is not obvious how often one needs to change the session key. The most secure approach is to use a new session key for each exchange.

Benefits of connectionless protocol

→ minimizes overhead & delay for each transaction.

A Transparent key control scheme The scheme is useful for providing end-to-end encryption at a ntw or transport level in a way that is transparent to the end users. (shown below)

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



[Automatic key distribution for connection oriented protocol]

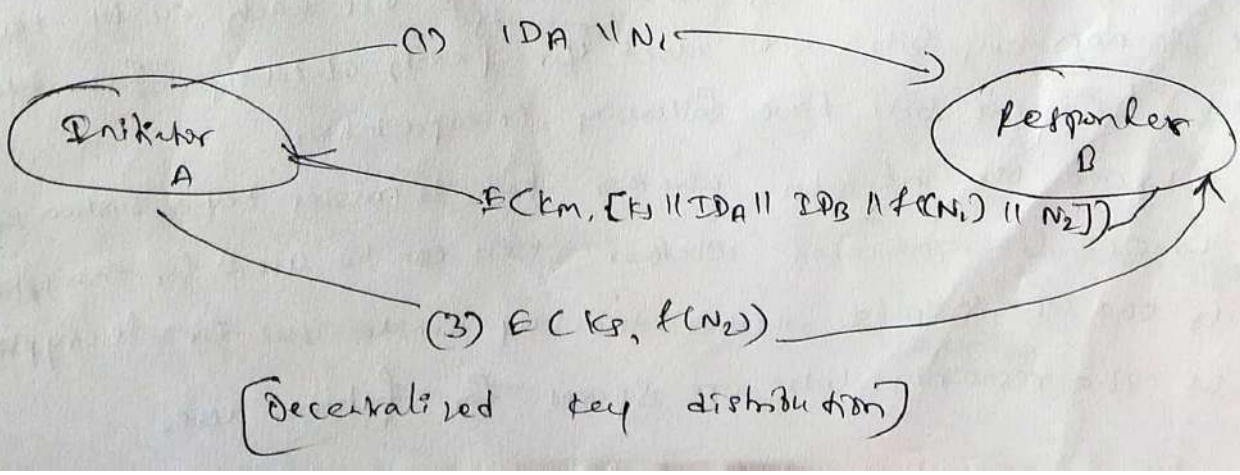
1. Host sends packet requesting conn
2. Security service buffers packet; asks KDC for session key.
3. KDC distributes session key to both hosts.
4. Buffered packet transmitted.

Decentralized key Control

⇒ A decentralized approach requires that each end system be able to communicate in a secure manner with all potential partner end systems for purposes of session key distribution. Thus, there may need to be as many as $[n(n-1)]/2$ master keys for configuration with n end systems.

⇒ A session key may be established with the following sequence of steps (figure 5)

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



[Decentralized key distribution]

* Controlling key usage : It also may be desirable to impose some control on the way in which automatically distributed keys are used. For eg in addition to seperating master keys from session keys, we may wish to define different types of session keys on the basis of use, such as

↳ Data encrypting key, for general communication across a n/w

↳ PIN-encrypting key, for personal Identification Numbers (PIN) used in electronic funds transfer & pos applications.

↳ File-encrypting key, for encrypting files stored in publicly accessible locations.

Normally, the master key is physically secured within the cryptographic h/w of the key distribution center & of the end systems. Session keys are encrypted with this master key are available to appln programs. However if a master key is treated as a session key, it may be possible for an unauthorized appln to obtain plaintext of session keys encrypted with that master key.

→ Thus it may be desirable to institute controls in systems that limit the ways in which keys are used, based on characteristics associated with those keys. One simple plan is to associate a tag with each key.

Eg IBM uses use of the extra 8-bits each 64-bit key, i.e 8 non-key bits are used (reserved) for parity checking form the key tag. The bits have following interpretation:

↳ One bit indicates whether key is master key or session key.

↳ One bit indicates whether key can be used for encryption.

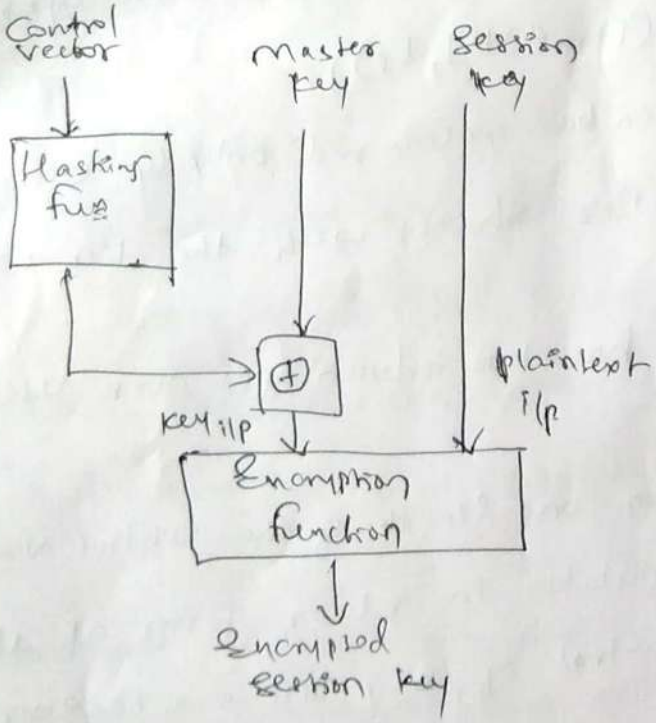
↳ One bit indicates whether the key can be used for decryption

↳ The remaining bits are spares for future use.

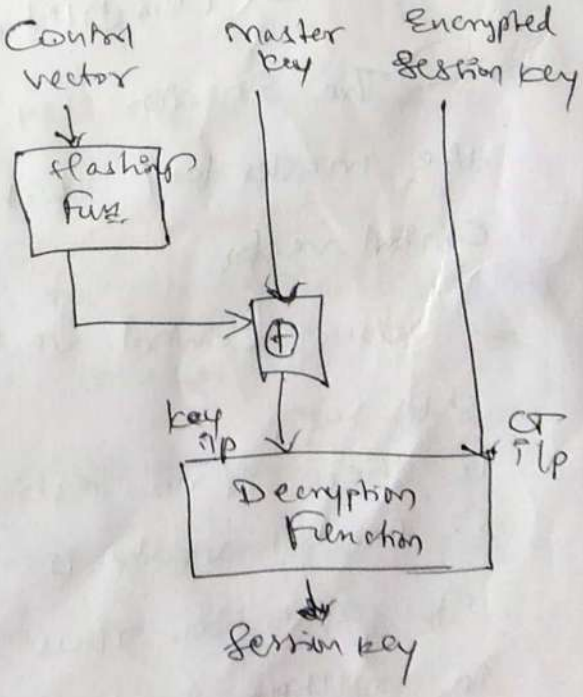
Because the tag is embedded in the key, it is encrypted along with the key when that key is distributed, thus providing protection. The drawbacks of this scheme are.

1. The tag length is limited to 8 bits, limiting flexibility & functionality.
2. Because the tag is not transmitted in clear form, it can be used only at the point of decryption, limiting the ways in which key use can be controlled.

→ A more flexible scheme, referred to as Control vector. In this scheme, each session key has an associated control vector consisting of a no. of fields that specify the uses & restrictions for that session key. The length of the control ~~key~~ vector may vary.



a) Control vector Encryption



b) Control vector decryption

[Control vector Encryption & decryption]

As a first step, the control vector is passed through a hash fun that produces a value whose length is equal the encryption key length. In essence a hash fun maps values from a larger range into a smaller range with a reasonably uniform spread.

The hash value is then xored with master key to produce an o/p that is used as the key i/p for encrypting the session key. Thus,

Hash value = $H = h(CV)$

Key input = $Km \oplus H$

Ciphertext = $E([Km \oplus H], Ks)$

where Km is the master key & Ks is session key.

The session key is recovered in pt by reverse operation:

$D([Km \oplus H], E([Km \oplus H], Ks))$

⇒ The session key can be recovered only by using both the master key that user shares with the KDC & the control vector.

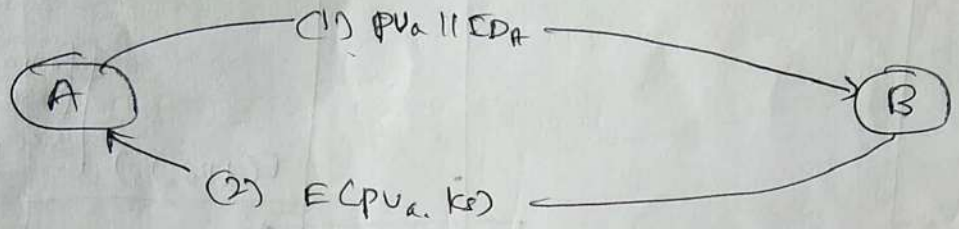
⇒ Use of control vector has two advantages over use of an 8 bit tag.

1. there is no restriction on length of the control vector
2. Control vector is available in clear form at all stages of operation. Thus control of key use can be exercised in multiple locations.

Symmetric key distribution using asymmetric Encryption

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

Simple Secret key Distribution An extremely simple scheme was put forward by merkle as illustrated in below figure.



Simple use of public key encryption to establish a session key

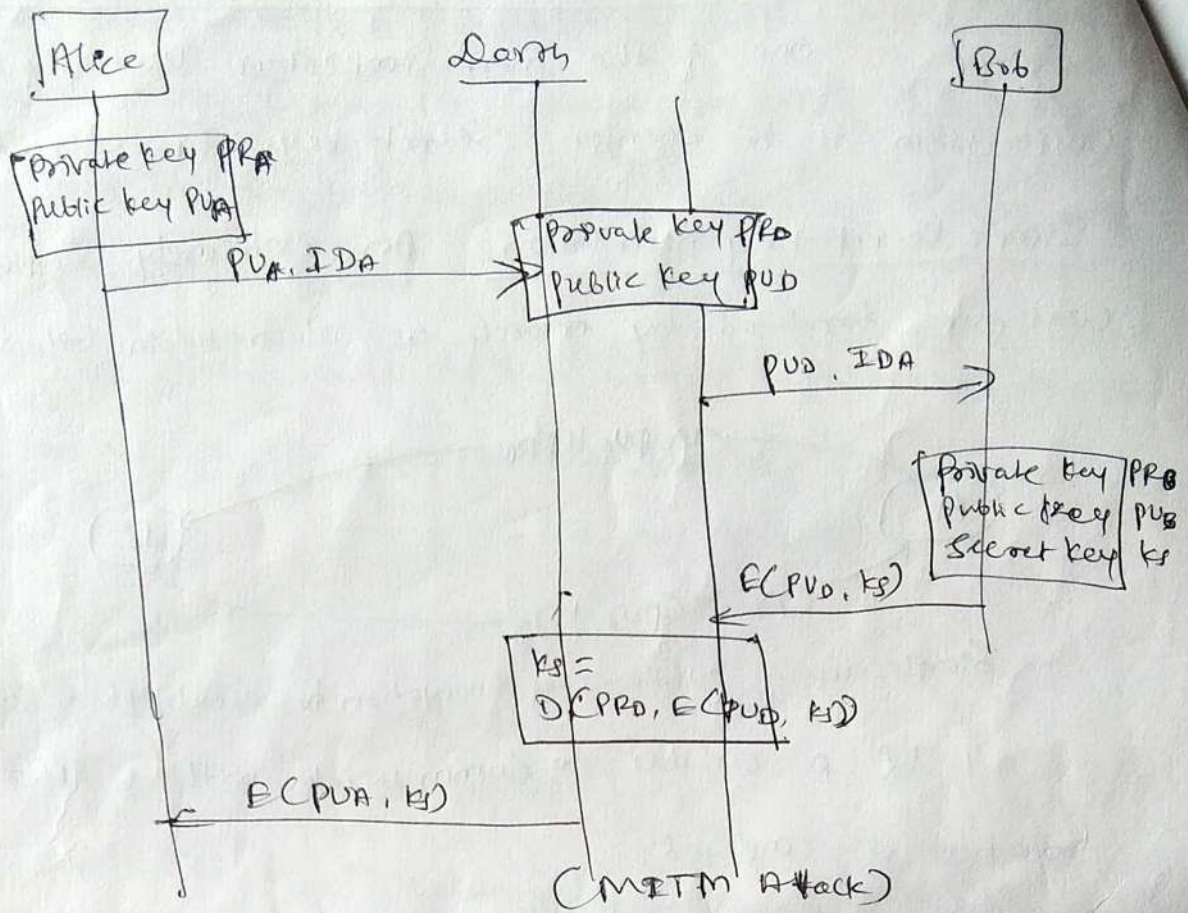
If A wishes to communicate with B, the following procedure is employed:

1. A generates a public/private key pair $\{PU_a, PR_a\}$ and transmits a msg to B consisting of PU_a & an identifier of A, ID_A.
2. B generates a secret key, K_s, and transmits it to A, which is encrypted with A's public key.
3. A computes $D(PR_a, E(PU_a, K_s))$ to recover the secret key. Because only A can decrypt the message.
4. A discards PU_a and PR_a & B discards PU_a.

A & B can now securely communicate using conventional encryption & the session key K_s. After the completion of exchange both A & B will discard K_s.

⇒ The protocol depicted above is insecure against an adversary who can intercept messages & then either relay the intercepted message or substitute another message.

Such an attack is known as a man-in the middle attack

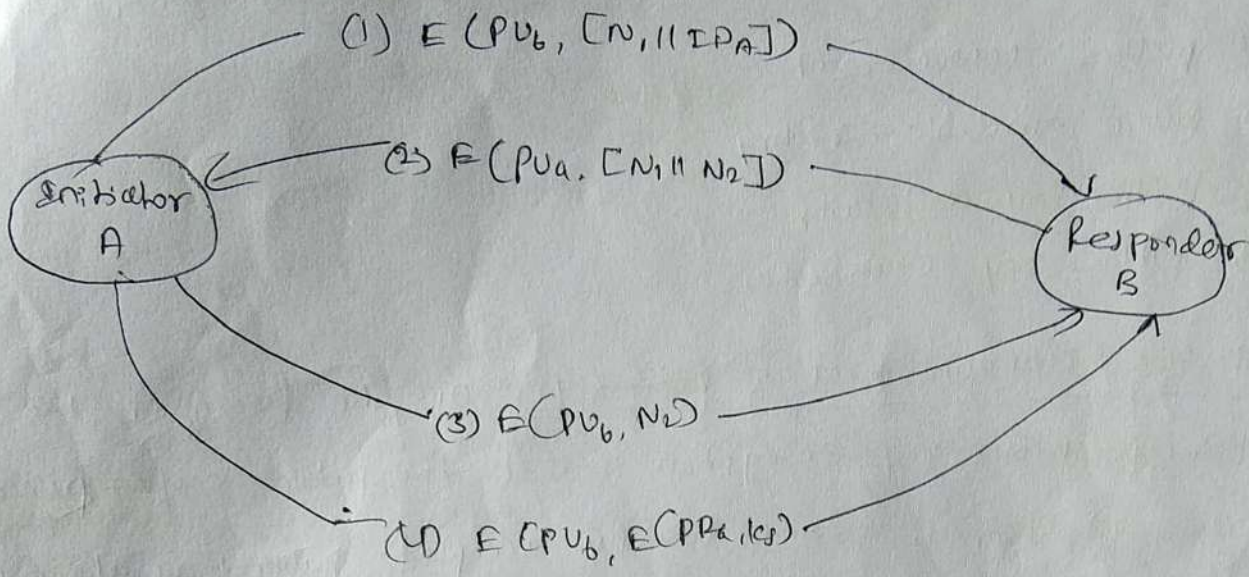


1. A generates a public/private key pair {PUa, PRa} & transmits a msg intended for B Consisting of PUa & an identifier of A, IDA.
2. D intercepts the message, creates its own public/private key pair {PUd, PRd} & transmits PUd || IDA to B.
3. B generates a secret key, Ks & transmits E(PUd, Ks).
4. D intercepts the message & learns Ks by computing D(PRd, E(PUd, Ks)).
5. D transmits E(PUa, Ks) to A.

The result is that both A & B know Ks and are unaware that Ks has also been revealed to D. A & B can now exchange message using Ks. D no longer interferes comm channel but eavesdrops.

Secret key distribution with Confidentiality & Authentication

The approach suggested with following figure ~~shows~~ provides protection against both active & passive attacks.



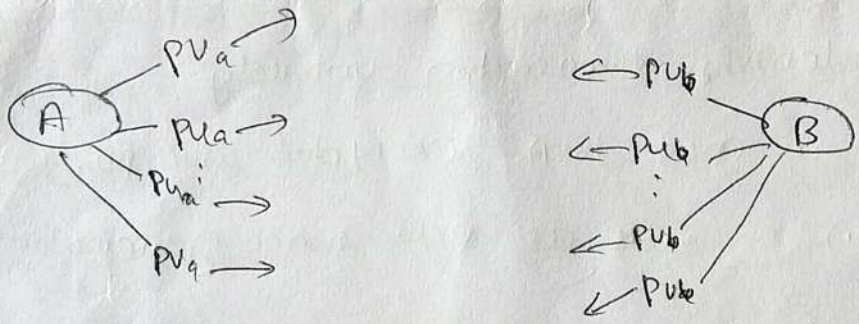
[Public-key Distribution of Secret keys]

1. A uses B's public key to encrypt a msg to B containing an identifier of A (IDA) & nonce (N1), which is used to identify transaction uniquely.
2. B sends a msg to A encrypted with PUa & containing A's nonce (N1) as well as new nonce generated by B (N2).
3. A returns N2 encrypted using B's public key to assure B that its correspondent is A.
4. 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.
5. B computes D(PUa, D(PRb, M)) to recover the secret key.

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

↳ public announcement
↳ public available directory
↳ public key authority
↳ public-key certificates.

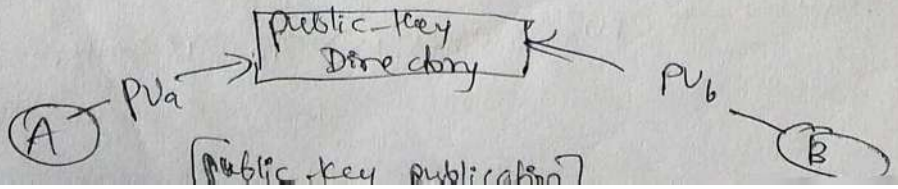
public announcement of public keys On the face of it, the point of public-key encryption is that the public key is public. Thus, if there is some broadly accepted public-key algorithm such as RSA, any participant can send his or her public key to any other participant or broadcast the key to the community at large.



(Uncontrolled public-key distribution)

weakness → Anyone can forge such a public announcement.

publicly available directory A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys. maintainance & distribution of the public directory would have to be the responsibility of some trusted entity or org^n



(public-key publication)

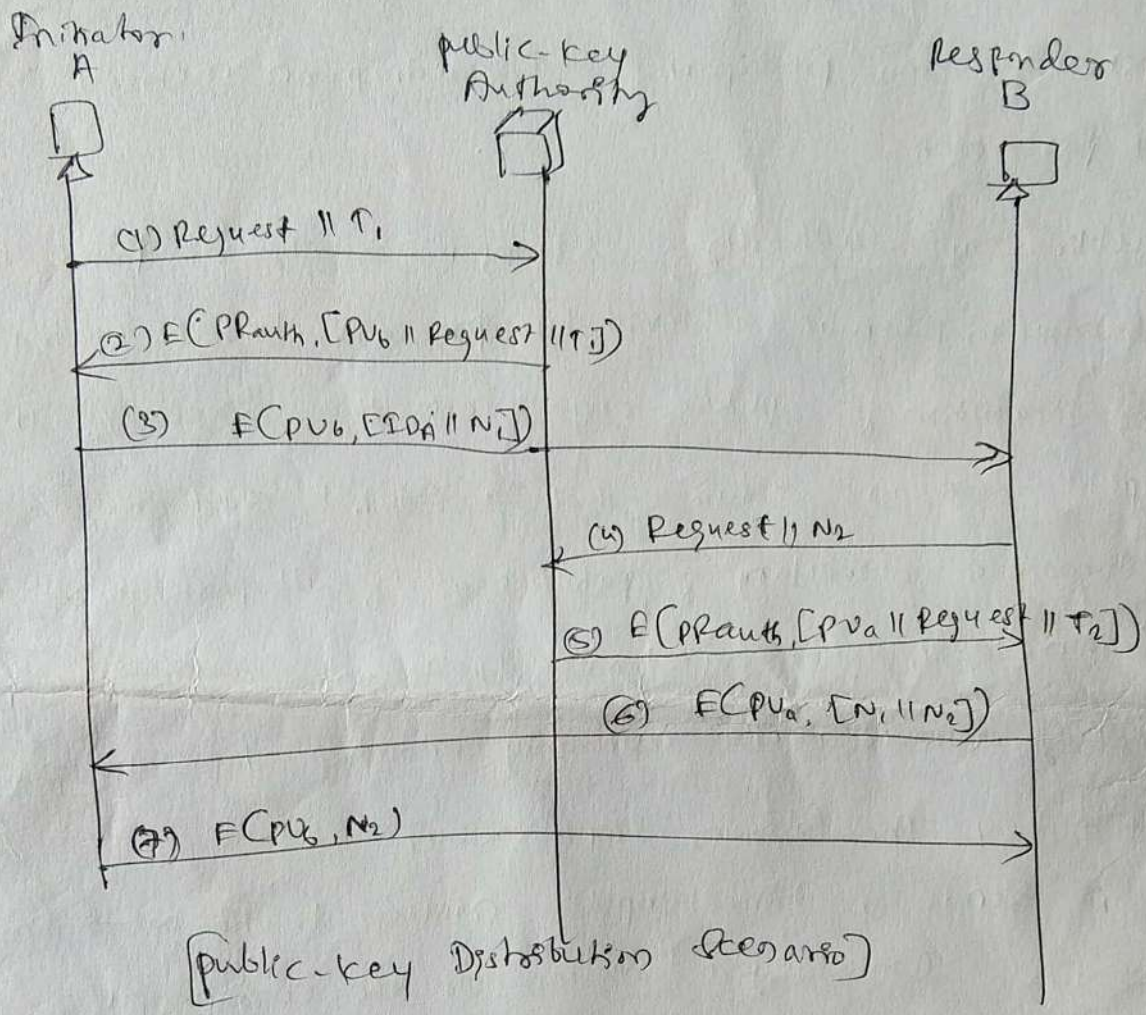
This scheme is clearly more secure than individual public announcements but still has vulnerabilities. If an adversary succeeds in obtaining or computing the private key of the directory authority, the adversary could authoritatively pass out counterfeit public keys & subsequently impersonate any participant & eavesdrop on messages sent to any participant.

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. The typical scenario assumes that a central authority maintains a dynamic directory of public keys of all participants. In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key. The following steps occur.

1. A sends a timestamped message to the public-key authority containing a request for the current public key of B.
2. The authority responds with a message that is encrypted using the authority's private key PRauth. Thus, A is able to decrypt the message using the authority's public key.
3. A stores B's public key & 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. 5. B retrieves A's public key from the authority in the same manner as A retrieved B's public key.

6) B sends a message to A encrypted with PU_a & containing A's nonce (N_1) as well as a new nonce generated by B (N_2).

7) A returns N_2, which is encrypted using B's public key, to assure B that its correspondent is A.



[public-key Distribution Scenario]

Publicly available Directory Contd...

1. The authority maintains a directory with a {name, public key} entry for each participant
2. Each participant registers a public key with the directory authority.
3. A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for large amount of data or because corresponding private key has been compromised in some way.
4. Participants could also access the directory electronically.

Public-key Certificates As before, the directory of names & public keys maintained by the authority is vulnerable to tampering.

An alternate approach, first suggested by Kohnfelder is to use certificates that can be used by participant to exchange keys without contacting a public key authority, in a way that is as reliable as if the keys were obtained directly from a public key authority. In essence a certificate consists of a public key, an identifier of the key owner, & the whole block signed by a trusted third party. Typically the third party is a certificate authority, such as govt. agency or a financial institution, that is trusted by the user community. A user can present his or her public key to the authority in a secure manner & obtain a certificate. The user can then publish a certificate. other participants can verify that the certificate was created by the authority.

⇒ we can place the following requirements on this scheme:

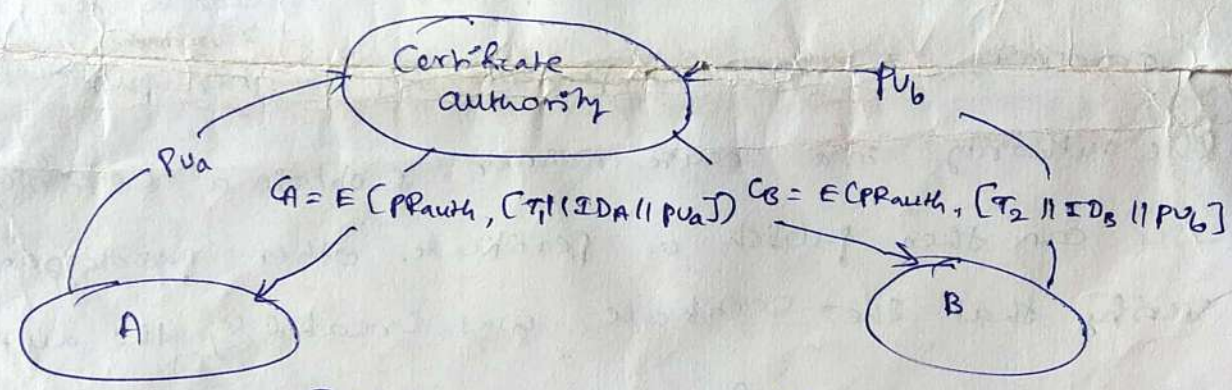
1. Any participant can read a certificate to determine the name & public key of the certificate's owner.
2. Any participant can verify that the certificate originated from the certificate authority & is not counterfeit.
3. only the certificate authority can create & update certificates.
4. Any participant can verify the currency of the certificate. [Additional requirement added by Denning]

A certificate scheme is illustrated in below figure. Each participant applies to the certificate authority, supplying a public keys & requesting certificate. Application must be in person or by some form of secure authenticated communication. For participant A, the authority provides a certificate of the form

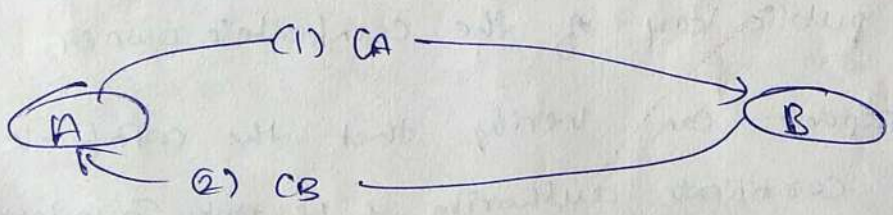
$$C_A = E(PR_{auth}, [T \| ID_A \| PU_a])$$

A may then pass this certificate on to any other participant, who reads & verifies the certificate as follows:

$$D(PU_{auth}, C_A) = D(PU_{auth}, E(PR_{auth}, [T \| ID_A \| PU_a]))$$
$$= (T \| ID_A \| PU_a)$$



[Obtaining certificates from CA]



[Exchanging Certificates]

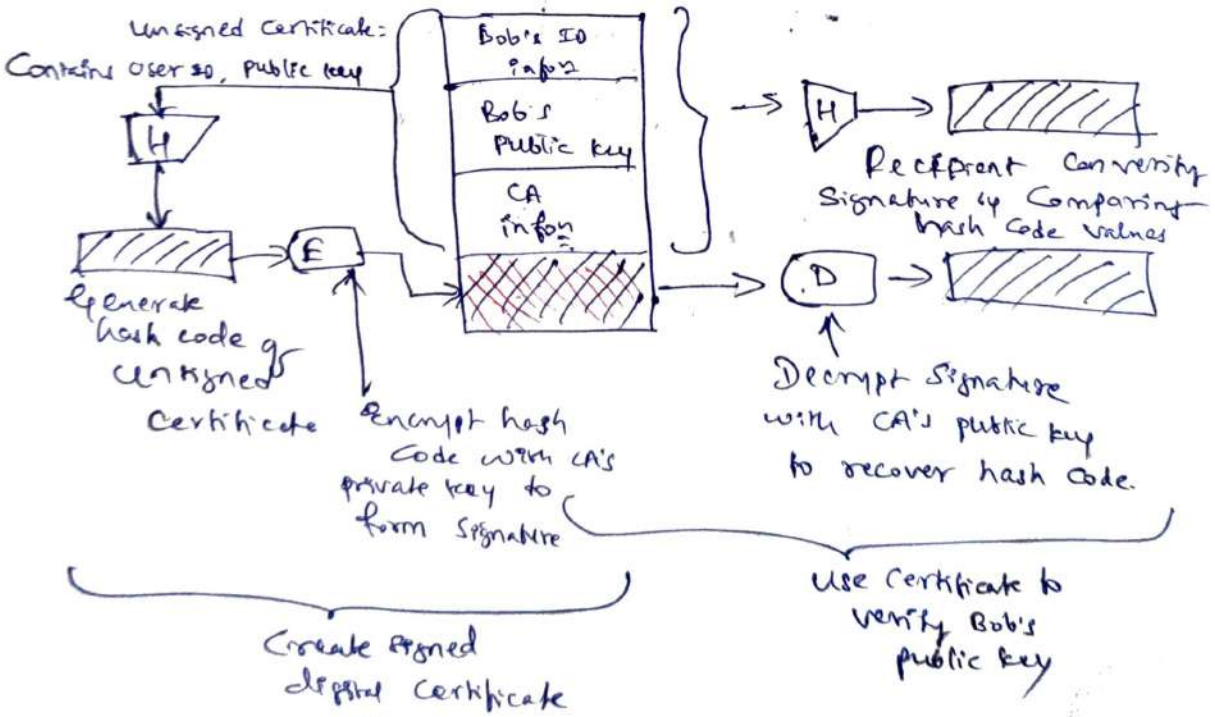
The recipient uses the authority's public key, PU_{auth} to decrypt the certificate. Because the certificate is readable only using the authority's public key,

X.509 Certificates ITU-T recommendation X.509 is part of the X.500 series of recommendations that define a directory service. The directory is, in effect a server or distributed set of servers that maintains a database of information about users.

→ X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public key certificates. Each certificate contains the public key of a user & is signed with the private key of a trusted certification authority.

→ The certificate structure & authentication protocols defined in X.509 are used in a variety of contexts such as S/MIME, IP security & SSL/TLS.

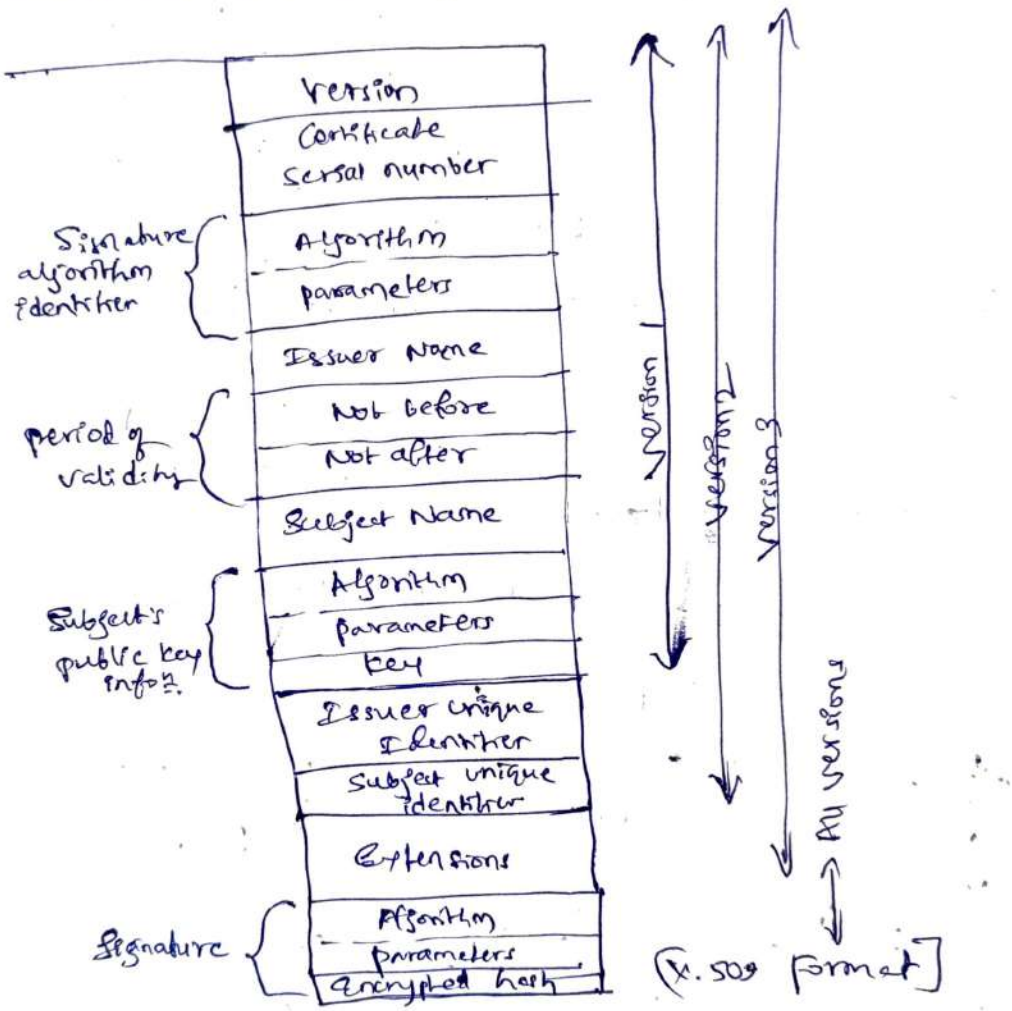
→ X.509 was initially issued in 1988. Later revised in 1993, 1995 & 2000.



(Public-key certificate use)

→ X.509 is based on the use of public key cryptography and digital signatures. The standard recommends RSA). The figure illustrates the generation of a public key certificate.

Certificates. Figure below shows the general format of a certificate which includes the following elements



Version - Differentiates among successive versions of the certificate format; the default version is 1.

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.

Issuer Name - X.500 name of the CA that created and signed this certificate.

Period of validity - Consists of two dates: the first and last on which the certificate is valid

Subject Name - The name of the user to whom this certificate refers.

Subject's public key information - The public key of the subject plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.

Issuer unique identifier - An optional-bit string field used to identify uniquely the issuing CA in the event of

Subject unique identifier - An optional-bit string field used to identify uniquely the subject.

Extensions - A set of one or more extensions.

Signature - Covers of 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.

Obtaining a User's Certificate - User certificates generated by a CA have the following characteristics:

↳ Any user with access to the public key of the CA can verify the user public key that was certified.

↳ No party other than the certification authority can modify the certificate without this being detected

→ If there is a large community of users, it may not be practical for all users to subscribe to the same CA.

with many users, it may be more practical for there to be a number of CAs each of which securely provides its public key to some fraction of the users.

→ Now suppose that A has obtained a certificate from certification authority X_1 & B has obtained certificate from CA X_2. If A does not securely know the public key of X_2, then B's certificate issued by X_2 is useless to A. A can read B's certificate, but A cannot verify the signature. However if the two CA's have securely exchanged their own public keys, the following procedure will enable A to obtain B's public key.

Step1: A obtains from the directory the certificate of X_2 signed by X_1. Because A knows X_1's public key, A can obtain X_2's public key from its certificate & verify it by means of X_1's signature on the certificate.

Step2: A then goes back to the directory & obtains the certificate of B signed by X_2. Because A now has a trusted copy of X_2's public key, A can verify the signature and securely obtain B's public key.

A has used a chain of certificates to obtain B's public key. In the notation of X.509, this chain is expressed as $X_1 \ll X_2 \gg X_2 \ll B \gg$

In the same fashion, B can obtain A's public key with the reverse chain: $X_2 \ll X_1 \gg X_1 \ll A \gg$

→ A chain with N elements would be expressed as

$$X_1 \ll X_2 \gg X_2 \ll X_3 \gg \ldots X_N \ll B \gg$$

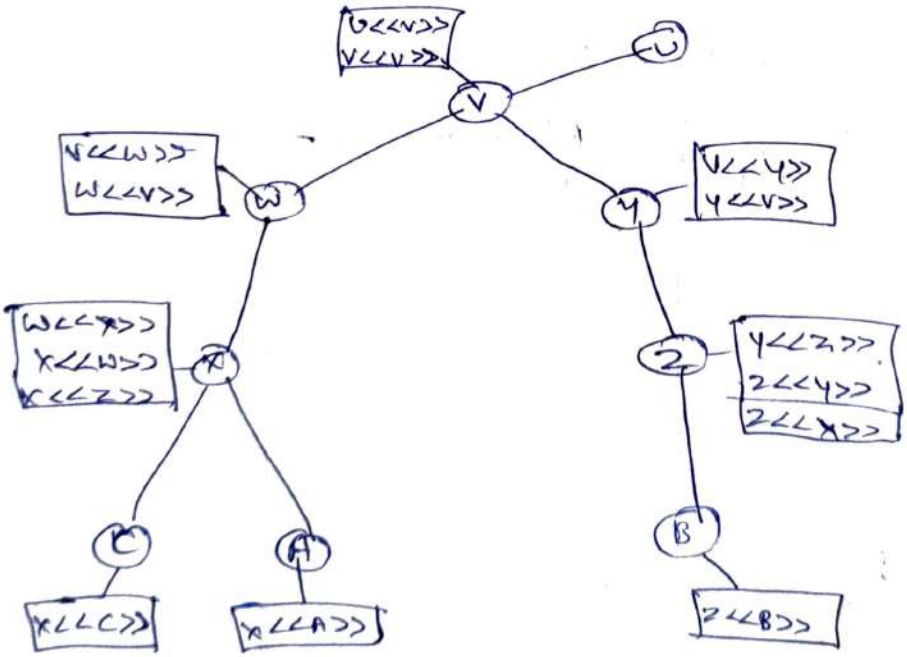
All these certificates of CAs by CAs need to appear in the directory and the user needs to know how they are linked to follow a path to another user's public key certificate

→ X.509 suggests that CA's be arranged in a hierarchy so that navigation is straightforward.

→ The following figure shows an example of such a hierarchy. The connected circles indicate the hierarchical relationship among CA's; the associated boxes indicates certificates maintained in the directory for each CA entry. The directory entry for each CA includes two types of certificates:

↳ Forward Certificates - Certificates of X generated by other CAs.

↳ Reverse Certificates - Certificates generated by X that are the certificates of other CAs.



[X.509 Hierarchy: Example]

In this example, User A can acquire the following certificate from the directory to establish a certification

Path to B:

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

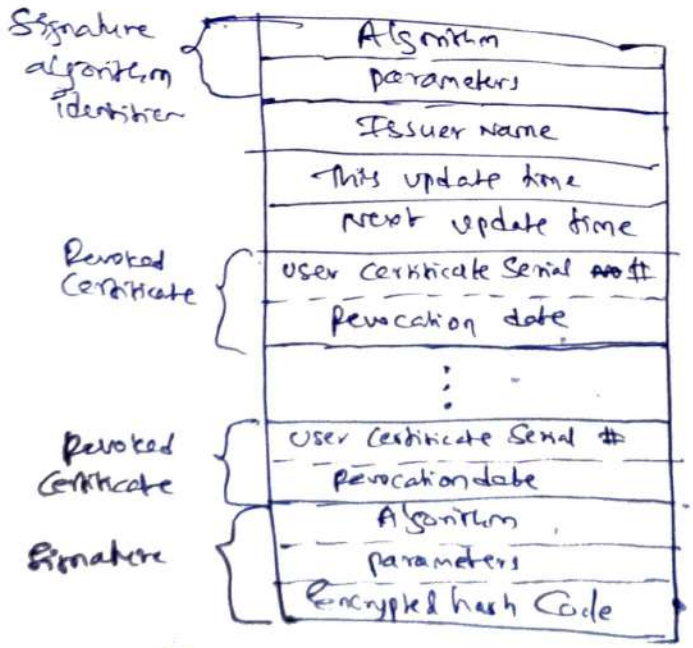
Similarly certification path from B to A.

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

* Revocation of Certificates In addition to period of validity, it may be desirable ~~to~~ on occasion to revoke a certificate before it expires, for one of the following reasons:

1. The user's private key is assumed to be compromised.
2. The user is no longer certified by this CA. (Reasons include that the subject's name has changed or the certificate was not issued in conformance with CA's policies.
3. The CA's certificate is assumed to be compromised.

→ Certificate Revocation List (CRL) must be maintained by the directory as below:



[Certificate Revocation List]

Y<< X>> - the certificate of user X issued by CA Y.

X509 version 3 Following requirements not satisfied by version 2.

1. The Subject field is inadequate to convey the identity of a key owner to a public key user.
2. The Subject field is also inadequate for many applications, which typically recognise entities by an Internet e-mail address, a URL, or some other Internet related identification.
3. There is a need to indicate security policy info. (such as IPsec)
4. There is a need to limit the damage that can result from a faulty or malicious CA by setting constraints on the applicability of a particular certificate.
5. It is important to be able to identify different keys used by the same owner at different times.

Thus version 3 includes a no. of optional ~~pe~~ extensions that may be added to version 2 format. The certificate extension fall into three main categories: key & policy info., Subject & Issuer attributes, & certification path constraints.

key and policy info.: These extensions convey additional info. about the Subject & Issuer keys plus 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 appln. with common security requirements. For ex a policy might be applicable to the authentication of electronic data interchange (EDI) transactions for the trading of goods within a given price range.

This area includes:

Authority key identifier Identifies the public key to be used to verify the signature on this certificate or CRL. Enables distinction among keys of same CA to be differentiated.

Subject key identifier - Identifies the public key being certified. Useful for subject key pairs updation (Eg: DS key & Encryption key)

Key usage - Indicates a restriction imposed as to the purposes for which, and the policies under which, the certified public key may be used. may indicate: digital signature, nonrepudiation, key encryption, data encryption, key agreement, CA signature verification on certificates, CA signature verification on CRLs.

Private key usage period: Indicates the period of use of the private key corresponding to public key, (Eg: signing key is typically shorter than verifying public key.

Certificate policies - Certificates may be used in environments where multiple policies apply.

Policy mappings - Used only in certificates for CAs issued by other CAs. One CA policy is equivalent to other CA's policy under same subject domain

Certificate Subject & Issuer attributes The extension fields in this area includes:

Subject alternative name: Contains one or more alternative names using any of a variety of forms. This field is important for supporting certain applications, such as electronic mail, EDI and IPSEC, which may employ their own name forms.

Issuer alternate name - Contains one or more alternative

names, using any of a variety of forms.

Subject directory attributes conveys any desired X.500 directory attribute values for the Subject of this Certificate.

Certificate path constraints These extensions allow constraint specifications to be included in certificates issued for CA's by other CA's The constraints may restrict the types of certificates that can be issued by the subject CA or that may occur subsequently in a certification chain.

The extension fields in this area include :

- Basic Constraints Indicates if the subject may acts as a CA. If so a certification path length constraint may be specified.

Name Constraints — Indicates a name space within which all subject names for subsequent certificates in a certification path must be located.

Policy Constraints - Specifies constraints that may require explicit certificate policy identification or inhibit policy mapping for the remainder of the certification path.

Subject directory attributes — may contain additional attributes associated with subject such as Date of Birth, place of birth, Country of Citizenship, Country of Residence etc

Basic Constraints - This extension describes whether the Certificate is a CA Certificate or an end entity certificate. (Path length)

Name Constraints - It specifies the constraints that apply on subject alternative names of subsequent certificates on the certificate path.

These Constraints can be applied to the form of permitted / excluded names.

If permitted then subsequent certificates must comply with this constraints.

(Ex: Directory name, DNS, Email address, URL, Ip Address)
→ (Constraints)

Ex: DN: Contoso
DNS: Contoso.com
URL: http://Contoso.com
Email: Bob@Contoso.com

⇒ Naming Constraints ensure conformance to set of naming rules.

Policy Constraints — are used to validate certification path. It can be used to prohibit policy mapping or require that an each certificate must be containing an acceptable policy identifier.

* public-key infrastructure - is the set of hardware, software, people, policies & procedures needed to create, manage, store, distribute & revoke digital certificates based on asymmetric cryptography.

→ The principal objective for developing a PKI is to enable secure, convenient & efficient acquisition of public keys.

→ The following figure shows the interrelationship among the key elements of the PKIX (public key infrastructure X509 model. These elements are

↳ End entity — A generic term used to denote end users

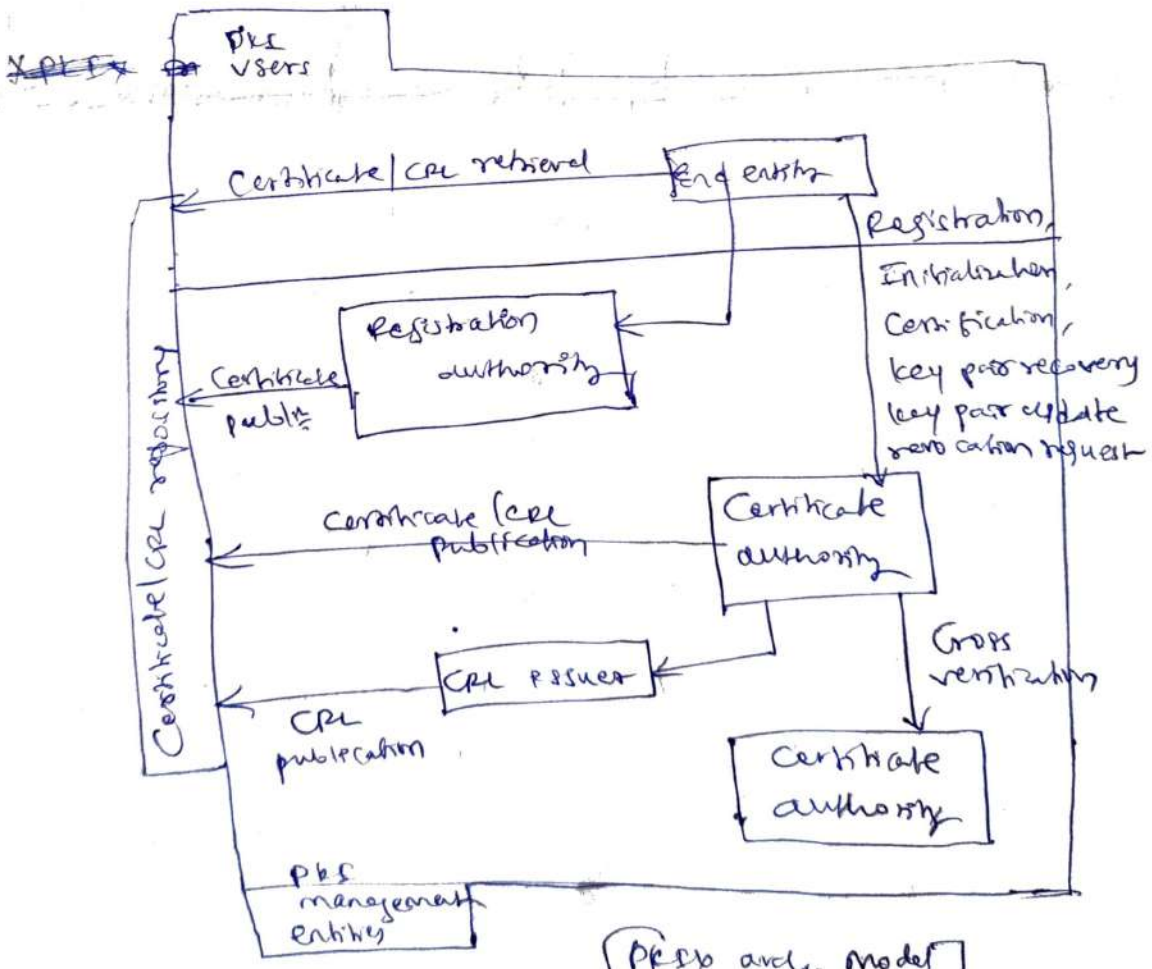
vices (eg Servers routers) or any other entity that can be identified in the Subject field of a public-key certificate.

Certification Authority - The issuer of certificates & CRLs. It may support variety of administrative functions.

Registration authority - RA is often associated with the end-entity registration process.

CRL issuer - An optional Component that a CA can delegate to publish CRLs.

Repository - A generic term used to denote any method for storing certificates & CRL's so that they can be retrieved by end entities.



PKIX arch. model

Registration PKIX management functions

Registration - Process whereby a user first makes itself known to a CA prior to that CA issuing a certificate. Regn usually involves some offline or online procedure for mutual authentication.

Initialization - Before a client system can operate securely, it is necessary to install key materials that have the appropriate relationship with keys stored elsewhere in the infrastructure. Eg. client need to initialize public key & other assured info of the CAs to be used in validating certificate paths.

Certification - process in which CA issues a certificate for a user's public key & returns to user's client system & posts in a repository.

key-pair recovery - key pair can be used to support digital signature creation & verification, encryption & decryption or both. This func allows an entity to restore keys from backup.

key-pair update - All key-pairs need to be updated regularly.

Revocation request - An authorized person advices a CA of an abnormal situation requiring certificate revocation.

Cross certification - Two CAs exchange info used in establishing a cross-certificate. A cross certificate is a certificate issued by one CA to another CA that contains a CA signature key used for issuing certificates.

User Authentication

Remote user authentication principles – User authentication as the process of verifying an identity claimed by or for a system entity. The process consists of two steps:

↳ **Identification step** – Presenting an identifier to the security system.

↳ **Verification step** – Presenting or generating authentication info. that corroborates the binding between the entity & the identifier.

Means of authentication – There are four general means of authenticating a user's identity, which can be used alone or in combination:

↳ **Something the individual knows** – Examples include a password, a PIN or answers to a prearranged set of questions.

↳ **Something the individual possesses** – Examples include cryptographic keys, electronic keycards, smart cards & physical keys.

↳ **Something the individual is (static biometrics)** – Examples include recognition by fingerprint, retina & face.

↳ **Something the individual does (dynamic biometrics)** – Examples include recognition by voice pattern, handwriting characteristics & typing rhythm.

Mutual Authentication – These protocols enable communicating parties to satisfy themselves mutually about each other's identity & to exchange session keys.

Central to the problem of authenticated key exchange are two issues: Confidentiality and timeliness.

Following lists the examples of replay attacks:

1. The simplest replay attack is one in which the opponent simply copies a message & replays it later.
2. An opponent can replay a timestamped msg within the valid time window.
3. In addition to above example 2, the opponent suppresses the original message. Thus, the repetition cannot be detected.
4. Another attack involves a backward replay without modification.

⇒ one approach to coping with replay attacks is to attach a sequence number to each message used for an authentication exchange. A new msg is accepted only if its sequence no. is in the proper order. The difficulty with this approach is that it requires each party to keep track of the last sequence number for each claimant it has dealt with.

Instead, one of the following two general approaches is used:

↳ Timestamps: party A accepts a msg as fresh only if the msg contains a timestamp that in A's judgement, is close enough to A's knowledge of current time. This approach requires that clocks among the various participants be synchronized.

↳ challenge/Response - party A, expecting a fresh msg from B, first sends B a nonce (challenge) & requires that the subsequent msg (response) received from B contain the correct nonce value.

One-way Authentication The very nature & its chief benefit, is that it is not necessary for the sender and receiver to be online at the same time. Instead the email msg is forwarded to the receiver's electronic mailbox, where it is buffered until the receiver is available to read it.

* Remote user-Authentication using Symmetric Encryption

Mutual Authentication

The Secret key distribution using a KDC initially put forth by Needham & Schroeder is summarized as below:

1. $A \rightarrow KDC : ID_A \| ID_B \| N_1$
2. $KDC \rightarrow A : E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$
3. $A \rightarrow B : E(K_b, [K_s \| ID_A])$
4. $B \rightarrow A : E(K_s, N_2)$
5. $A \rightarrow B : E(K_s, f(N_2))$ where $f()$ is a generic fun. that modifies the value of the nonce.

Despite the handshake of steps 4 and 5, the protocol is still vulnerable to a form of replay attack.

⟹ Denning proposes to overcome this weakness by a modification to the Needham/Schroeder protocol that includes the addition of a timestamp to steps 2 & 3. Her proposal assumes that the master keys, K_a & K_b, are secure & it consists of the following steps.

1. $A \rightarrow KDC : ID_A \| ID_B$
2. $KDC \rightarrow A : E(K_a, [K_s \| ID_B \| T \| E(K_b, [K_s \| ID_A \| T])])$
3. $A \rightarrow B : E(K_b, [K_s \| ID_A \| T])$

4. $B \rightarrow A : E(K_s, N_1)$

5. $A \rightarrow B : E(K_s, f(N_1))$

T is a timestamp that assures A & B that the session key has only just been generated. Thus, both A & B know that the key distribution is a fresh exchange.

⇒ Here is a risk with above approach i.e based on the fact that the distributed clocks can become unsynchronized as a result of sabotage on or faults in the clocks or the synchronization mechanism. The problem occurs when a sender's clock is ahead of the intended recipient's clock. In this case an opponent can intercept a message from the sender & replay it later when the timestamp in the message becomes current at the recipient's site. This replay could cause unexpected results. Denning refers to such attacks as Suppress-replay attacks.

⇒ The improved strategy was presented to fix the problems in the Needham/Schroeder protocol is :

1. $A \rightarrow B : ID_A \| N_a$
2. $B \rightarrow KDC : ID_B \| N_b \| E(K_b, [ID_A \| N_a \| T_b])$
3. $KDC \rightarrow A : E(K_a, [ID_B \| N_a \| K_s \| T_b]) \| E(K_b, [ID_A \| K_s \| T_b]) \| N_b$
4. $A \rightarrow B : E(K_b, [ID_A \| K_s \| T_b]) \| E(K_s, N_b)$

One-Way Authentication With some refinement, the KDC strategy is a candidate for encrypted electronic mail. Because we wish to avoid requiring that the recipient (B)

on line at the same time as the sender(A), steps 4 & 5 must be eliminated. For a message with Content M, the sequence is as follows:

1. $A \rightarrow KDC : ID_A \| ID_B \| N_1$
2. $KDC \rightarrow A : E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$
3. $A \rightarrow B : E(K_b, [K_s \| ID_A]) \| E(K_s, M)$

This approach guarantees that only the intended recipient of a message will be able to read it. It also provides a level of authentication that the sender is A.

* Kerberos

→ It an authentication service developed as part of Project Athena at MIT.

→ The problem that kerberos addresses is this: Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the n/w. We would like for servers to be able to restrict access to authorised users & to be able to authenticate requests for service.

In particular, the following three threats exist:

1. A user may gain access to a particular workstation & pretend to be another user operating from that workstation.
2. A user may alter the n/w address of a workstation so that the requests sent from the altered workstations appear to come from impersonated workstation.

3. A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

Motivation more common in a distributed architecture consisting of dedicated user workstations (clients) and distributed or centralized servers. In this environment, three approaches to security can be envisioned.

1. Rely on each individual client workstation to assure the identity of its user or users and rely on each server to enforce a security policy based on user ID.
2. Require that client systems authenticate themselves to servers, but trust the client system concerning the identity of its user.
3. Require the user to prove his or her identity for each service invoked. Also require that servers prove their identity to clients.

The first published report on Kerberos listed the following requirements.

↳ Secure - A n/w eavesdropper should not be able to obtain the necessary info. to impersonate a user.

↳ Reliable - Kerberos should be highly reliable & should employ a distributed server architecture with one system able to backup 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 no. of clients & servers.

Kerberos version 4

→ version 4 of kerberos makes use of DES, in a rather elaborate protocol, to provide authentication service.

A Simple Authentication Dialogue In an unprotected n/w environment, any client can apply to any server for service. The obvious security risk is that of impersonation. To counter this threat, servers must be able to confirm the identities of clients who request service. Each server can be required to undertake this task for each client/server interaction, but in an open environment this places a substantial burden on each server.

⟹ An alternative is to use an authentication server (AS) that knows the passwords of all users and stores these in a centralized database. In addition, the AS shares a unique secret key with each server. These keys have been distributed physically or in some other secure manner.

Consider the following hypothetical dialogue:

1) $C \rightarrow AS$: $ID_C \| P_C \| ID_V$

2) $AS \rightarrow C$: Ticket

3) $C \rightarrow V$: $ID_C \| Ticket$

$Ticket = E(K_V, [ID_C \| AD_C \| ID_V])$

where

C = client

AS = Authentication Server

V = Server

IDC = identifier of user on C

IDV = identifier of V

PC = password of user on C

ADC = n/w address of C

KV = Secret Encryption key shared by AS & V.

A more Secure Authentication Dialogue

Under the above scheme, it remains the case that a user would need a new ticket for every different service. If a user is wished to access a print server, a mail server, a file server & so on, the first instance of each access would require a new ticket & hence require the user to enter the password.

The second problem is that the earlier scenario involved a plaintext transmission of the password. An eavesdropper could capture the password & use any service accessible to the victim.

TO Solve these additional problems, we introduce a scheme for avoiding plaintext passwords & a new server, known as the ticket-granting Server (TGS). The scenario is as follows:

Once per login session:

1) C → AS : $ID_C \| ID_{tgs}$

2) AS → C : $E(K_C, Ticket_{tgs})$

Once per type of service:

3) $C \rightarrow TGS : ID_C \| ID_V \| Ticket_{tgs}$

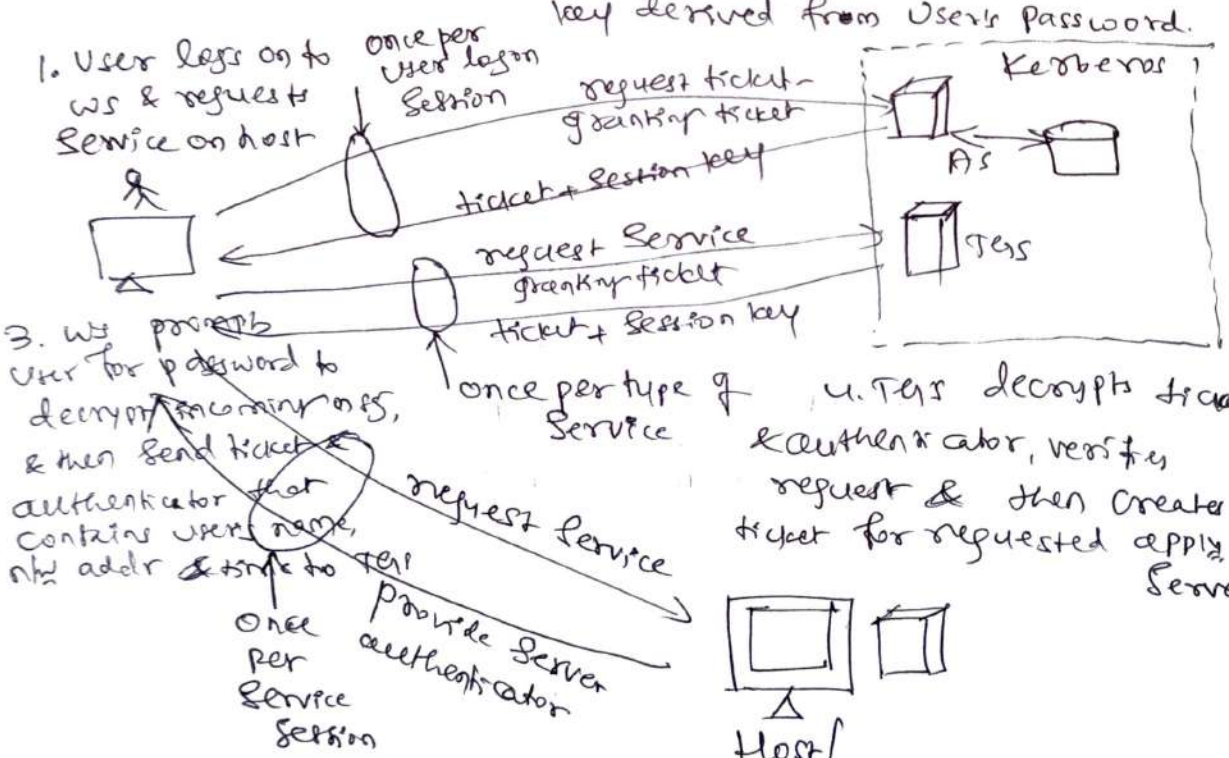
4) $TGS \rightarrow C : Ticket_v$

Once per service session:

5) $C \rightarrow V : ID_C \| Ticket_v$

$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$

$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_V \| TS_2 \| Lifetime_2])$



[overview of kerberos]

* Remote user-Authentication using Asymmetric Encryption

A protocol using timestamps is

1. $A \rightarrow AS : ID_A \| ID_B$
2. $AS \rightarrow A : E(PR_{as}, [ID_A \| PU_a \| T]) \| E(PR_{as}, [ID_B \| PU_b \| T])$
3. $A \rightarrow B : E(PR_{as}, [ID_A \| PU_a \| T]) \| E(PR_{as}, [ID_B \| PU_b \| T]) \| E(PU_b, E(PR_a, [K_s \| T]))$

This protocol is compact but, as before requires the synchronization of clocks. Another approach proposed by Woo and Lam, makes use of nonces. The protocol consists of the following steps:

1. $A \rightarrow KDC : ID_A \| ID_B$
2. $KDC \rightarrow A : E(PR_{auth}, [ID_B \| PU_b])$
3. $A \rightarrow B : E(PU_b, [N_a \| ID_A])$
4. $B \rightarrow KDC : ID_A \| ID_B \| E(PU_{auth}, N_a)$
5. $KDC \rightarrow B : E(PR_{auth}, [ID_A \| PU_a]) \| E(PU_b, E(PR_{auth}, [N_a \| K_s \| ID_B]))$

The above protocol seems to be a secure protocol that takes into account the various attacks. However, the authors themselves spotted a flaw & submitted a revised version of the algorithm:

1. $A \rightarrow KDC : ID_A \| ID_B$
2. $KDC \rightarrow A : E(PR_{auth}, [ID_B \| PU_b])$
3. $A \rightarrow B : E(PU_b, [N_a \| ID_A])$
4. $B \Rightarrow KDC : ID_A \| ID_B \| E(PU_{auth}, N_a)$
5. $KDC \rightarrow B : E(PR_{auth}, [ID_A \| PU_a]) \| E(PU_b, E(PR_{auth}, [N_a \| K_s \| ID_A \| ID_B]))$

6. $B \rightarrow A : E(PU_a, [N_b \| E(PR_{auth}, [\overline{N_a} \| K_s \| ID_A \| ID_B])])$

7. $A \rightarrow B : E(K_s, N_b)$

The identifier of A, ID_A is added to the set of items encrypted with the KDC's private key in steps 5 & 6. This binds the session key K_s to the identities of the two parties that will be engaged in the session.

one-way Authentication

If Confidentiality is the primary concern, then the following may be more efficient :

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

In this case, the message is encrypted with a one-time secret key. A also encrypts this one-time key with B's public key. Only B will be able to use the corresponding private key to recover one-time key and then use that key to decrypt the message.

If authentication is the primary concern, then a digital signature may suffice,

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

This method guarantees that A cannot later deny having sent the message.

→ Both the message and signature can be encrypted with the recipient's public key :

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

Module-5 IP Security

IP Security overview - In 1994 the Internet Architecture Board (IAB) issued a report titled "Security in Internet arch.". The report identified a key areas for security mechanisms. Among these were the need to secure the n/w infrastructure from unauthorized monitoring & control of n/w traffic & the need to secure end-to-end user traffic using authentication & encryption mechanisms.

⇒ **IPsec** - It a Secure n/w protocol suite that authenticates and encrypts the packets of data to provide Secure encrypted comm^n betⁿ two computers over an Internet Protocol n/w.

✱ **Applications of IPsec** IPsec provides the capability to secure communications across a LAN, across private & public WANs & across the Internet. Examples of its include:

↳ **Secure branch office Connectivity over the Internet**: A Company can build a secure virtual private n/w over the Internet or a over a public WAN.

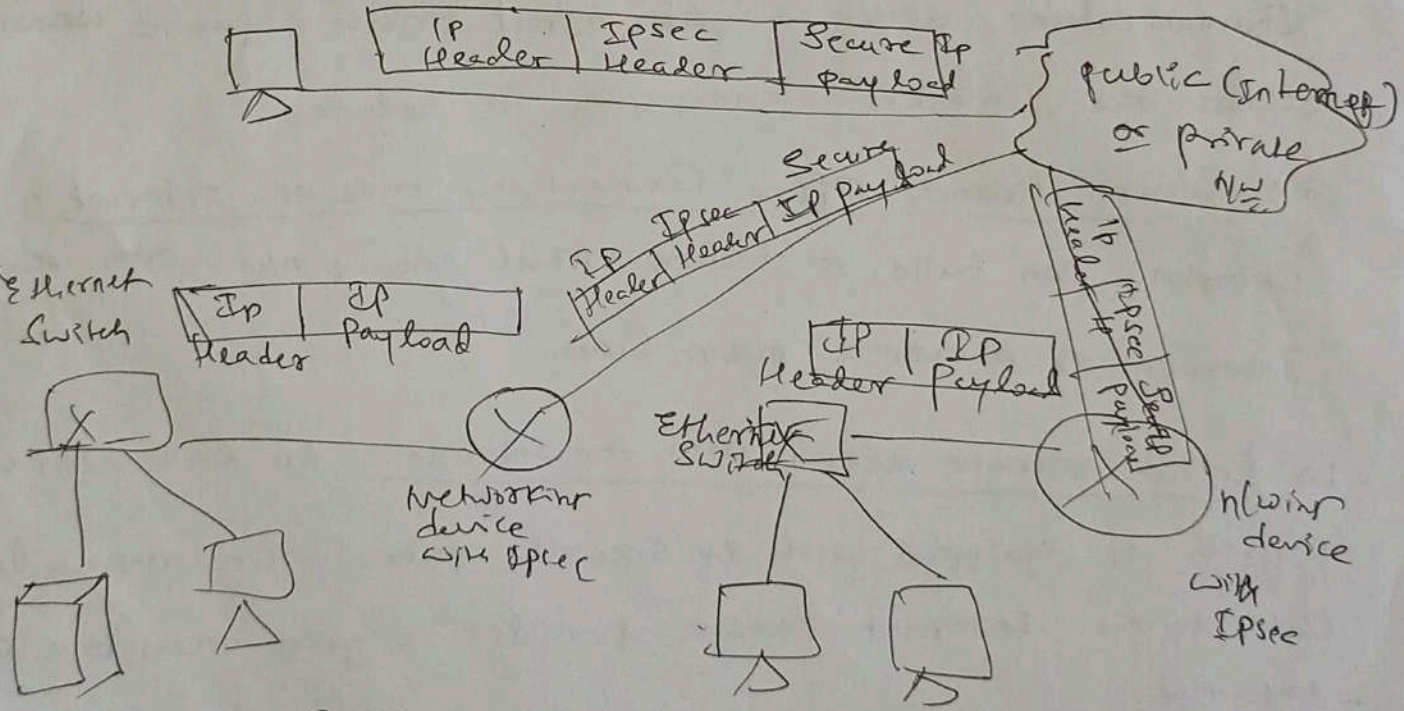
↳ **Secure remote accen over the Internet**: - An end user whose system is equipped with IP security protocols can make a local calls to an Internet Service provider & gain access to a Comp- any n/w.

↳ **Establishing extranet & intranet Connectivity with partners** Ipsec can used to secure communication with other orgⁿs, ensuring auth^n, confidentiality & providing key exchange mechanism.

↳ Enhancing electronic Commerce Security Even though some web & electronic Commerce apps have built-in Security protocols the use of Ipsec enhances security. Ipsec guarantees that all traffic designated by n/w administrator is both encrypted & authenticated, adding an additional layer of security to whatever is provided at the app. layer.

⇒ The principal feature of Ipsec that enable it to support these varied apps is that it can encrypt and/or authenticate all traffic at the Ip level. All distributed apps (remote logon, client/server, e-mail, file transfer, web access so on) can be secured.

→ Following figure shows a typical scenario of Ipsec usage.



[An IP Security Scenario]

→ An org maintains LANs at dispersed locations. Nonsecure IP traffic is conducted on each LAN. Ipsec protocols operate in networking devices such as router or firewall

that Connect each LAN to the outside world. The IPsec networking device will typically encrypt & compress all traffic going into the WAN & decrypt & decompress traffic coming from the WAN

* Benefits of IPsec Some of the benefits of IPSec:

↳ When IP Sec is implemented in a firewall or router, it provides strong security that can be applied to all traffic Crossing the perimeter.

↳ IPsec in a firewall is resistant to bypass if all traffic from the outside must use IP & the firewall is the only means of entrance from the Internet into the orgas.

↳ IPsec is below the transport layer (TCP, UDP) & so is transparent to apps. There is no need to change software on a user or server system when IP Sec is implemented in the firewall or router.

↳ 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 orgs.

↳ IPsec can provide security for individual users if needed.

* Routing apps In addn to supporting end users & protecting premises systems & nws, IPsec can play vital role in the routing arch required for internetworking. IPSec can assure that
↳ A router advertisement (a new router advertises its presence) comes from an authorized router.

↳ A neighbor advertisement (a router seeks to establish or maintain neighbor relationship with a router in another routing domain) comes from an authorized router.

↳ A redirect message comes from the router to which the initial IP packet was sent

↳ A routing update is not forged.

* IPSec Documents. The totality of IPsec specification is scattered across dozens of RFCs & draft IETF documents. The documents can be categorized into the following groups:

Architecture: Covers general concepts, security requirements, definitions & mechanisms defining IPsec technology. The current spec is RFC 4301, Security architecture for the Internet protocol.

Auth Header: AH is an extension header to provide message authentication. The current specification is RFC 4302.

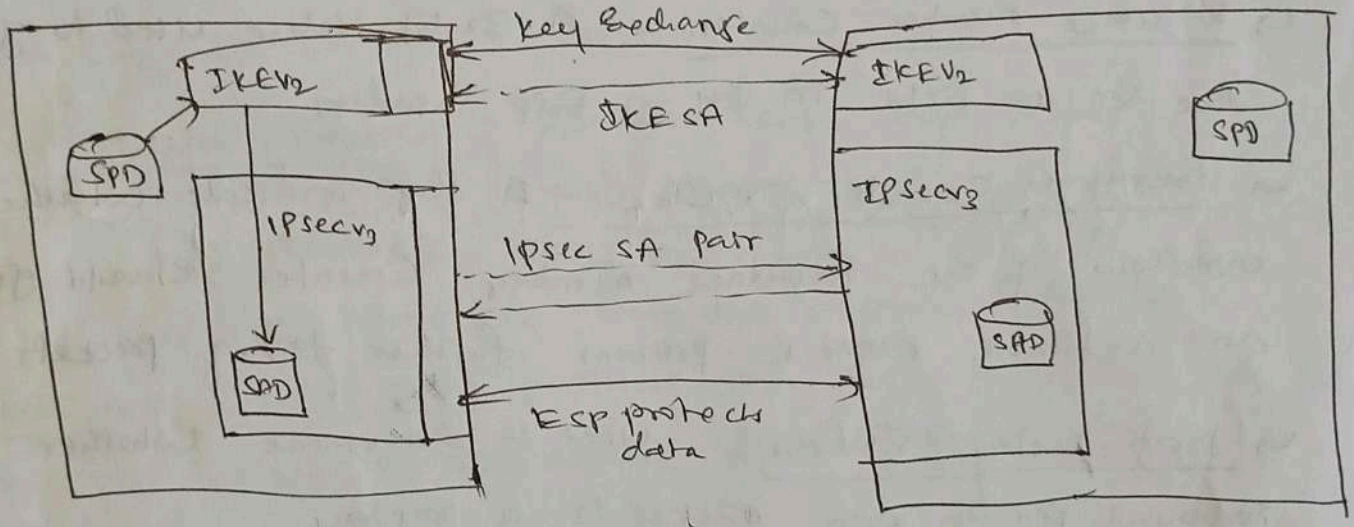
Encapsulating Security payload (ESP) - ESP consists of an encapsulating header & trailer used to provide encryption or combined encryption/authentication. The current spec is RFC 4303.

Internet key Exchange - This is a coll of documents describing the key management schemes for use with IPsec. The spec is RFC 5996.

Cryptographic algorithms - This category encompasses a large set of documents that define & describe cryptographic algorithms for encryp, msg auth, pseudorandom functions (PRFs) & cryptographic key exchange

* **IPSec Services** RFC 4301 lists the following services:
  - ↳ Access Control
  - ↳ Connectionless integrity
  - ↳ Data origin authn
  - ↳ Rejection of replayed packets (a form of partial Sequence integrity)
  - ↳ Confidentiality (encryption)
  - ↳ Limited traffic flow Confidentiality

* **IP Security Policy** - IPsec policy is determined primarily by the interaction of two databases, the security association database (SAD) & the security policy database (SPD).



[IPsec Architecture]

Security Associations An association is a one-way logical conn. betn. a sender & receiver that affords security services to the traffic carried on it.

⇒ A Security Association is uniquely identified by three parameters:

- ↳ **Security parameter index (SPI)** A 32-bit unstructured integer assigned to this SA having local significance only. The SPI is carried in AH & ESP header.

↳ Ip dest addrn - This is the address of the destn endpoint of the SA which may be an end-user system or a n/w system such as a firewall or router.

↳ Security Protocol identifier: This field is from the outer IP header indicates whether the association is an AH or ESP Security Association.

Security Association Database SAD defines the parameters associated with each SA. A Security Association is normally defined by the following parameters in an SAD entry.

↳ Security parameter index - A 32-bit value selected by the receiving end of an SA to uniquely identify the SA.

↳ Sequence number Counter - A 32-bit value used to generate the Seq no. field in AH or ESP headers.

↳ Sequence Counter overflow A flag indicate whether overflow of the Sequence number Counter should generate an auditable event & prevent further trans. of packets.

↳ Anti-replay window - Used to determine whether an inbound AH or ESP packet is a replay.

↳ AH info - Auth algo, keys, key lifetimes etc.

↳ ESP info - Encryption & Auth algorithm, keys, init^n values, key lifetimes etc.

↳ Lifetime of this Security Association - A time interval or byte count after which an SA must be replaced with a new SA.

↳ Ipsec protocol mode - Tunnel or transport mode

↳ Path MTU - maximum size of a packet that can be transmitted without fragmentation.

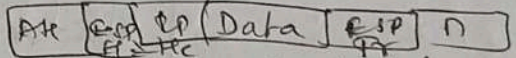
* Security policy Database - Each SPD entry is defined by a set of IP & upper-layer protocol field values, called Selectors.

The following Selectors determine an SPD entry:

Remote IP address: This may be a single IP address, an enumerated list or range of addresses. The latter two are required to support more than one dest^n System sharing the same SA. (behind a firewall)

Local IP address: This may be a single IP address, an enumerated list or range of addresses. The latter two are required to support more than one source System sharing the same SA. (behind a firewall)

Next Layer Protocol - defines next layer protocol



Name - IP precedes the AH or ESP header in packet

↳ A user identifier from the operating System.

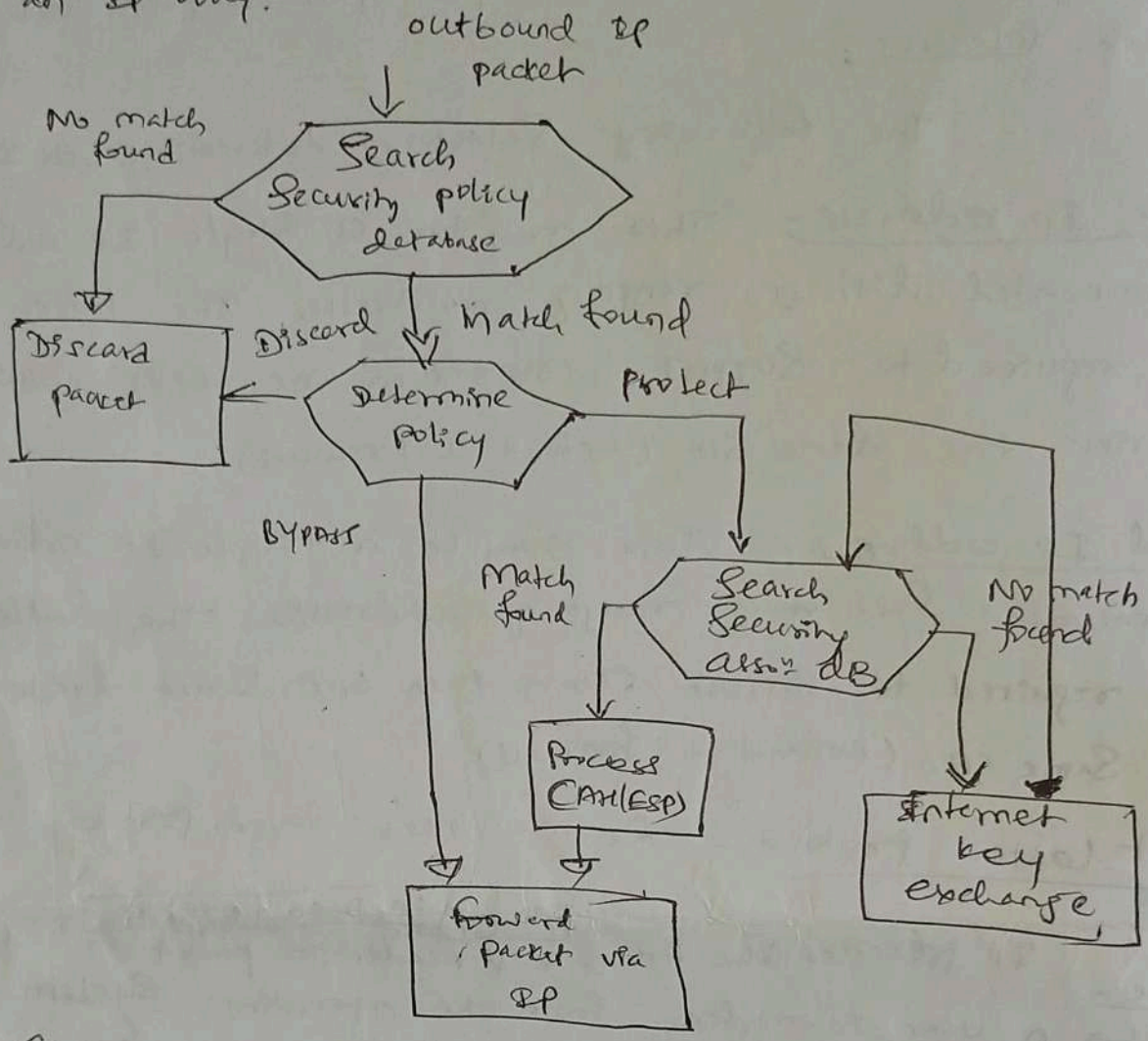
Local and Remote Ports: There may be individual TCP or UDP port values, an enumerated list of ports.

* IP Traffic processing

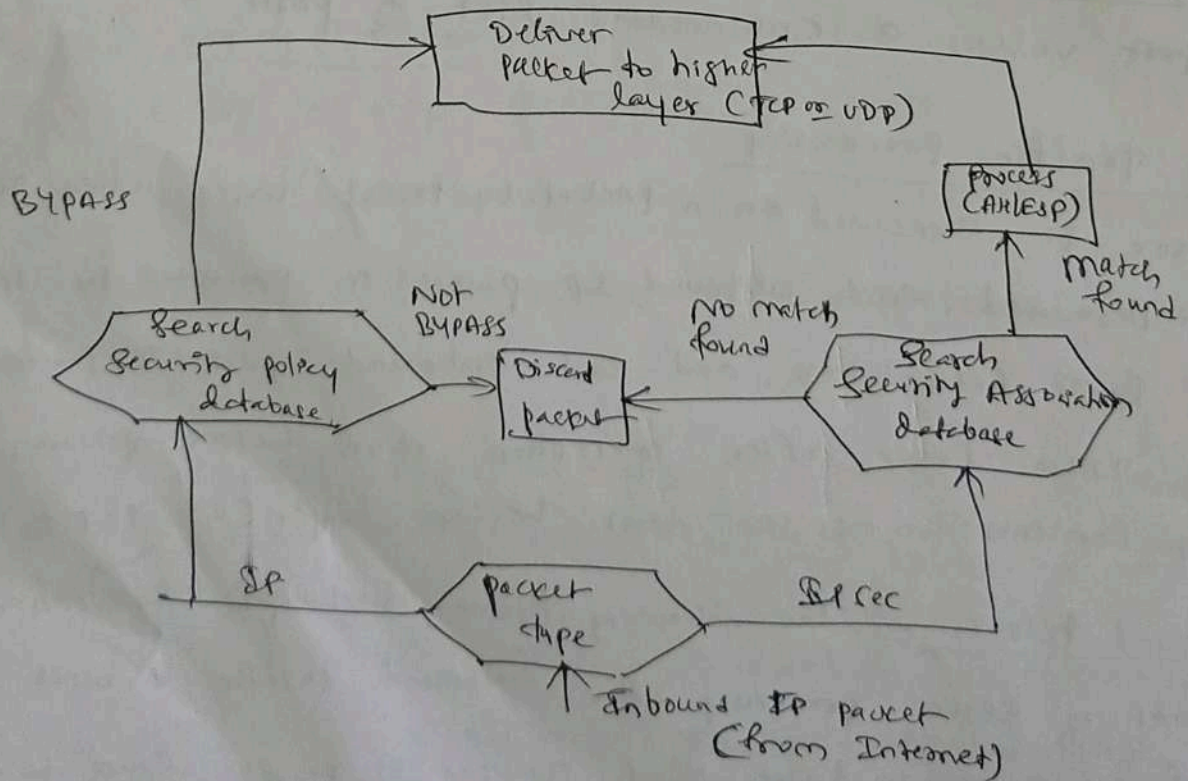
→ IPsec is executed on a packet-by-packet basis. When IPsec is implemented, each outbound IP packet is processed by the IPsec logic before trx^n, and each inbound packet is processed by the IPsec logic after reception and before passing the packet contents on to the next higher layer (ex TCP or UDP)

outbound Packets - The following figure highlights the main elements of IPsec processing for outbound traffic. A block of data from a higher layer, such as TCP, is passed down to the

IP layer e an IP packet fs formed, consisting of an IP header and an IP body.



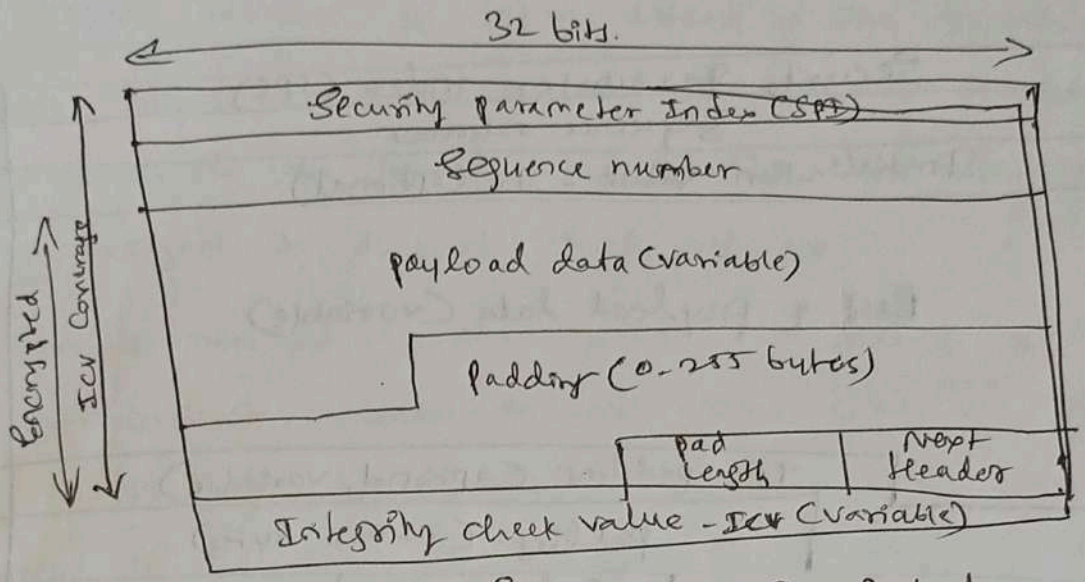
(Processing model for outbound packets)



* Encapsulating Security payload

→ Esp can be used to provide Confidentiality, data origin authn., Connectionless integrity, an anti-replay Service and traffic flow Confidentiality.

Esp format



(a) Top-level format of an Esp packet.

Security parameter Index - Identifies Security Association.

Sequence number - (32 bits) - A monotonically increasing Counter value; this provides an anti-replay function.

Payload data (variable) - This is a transport-level Segment (transport mode) or Ip packet (tunnel mode) that is protected by Encryption.

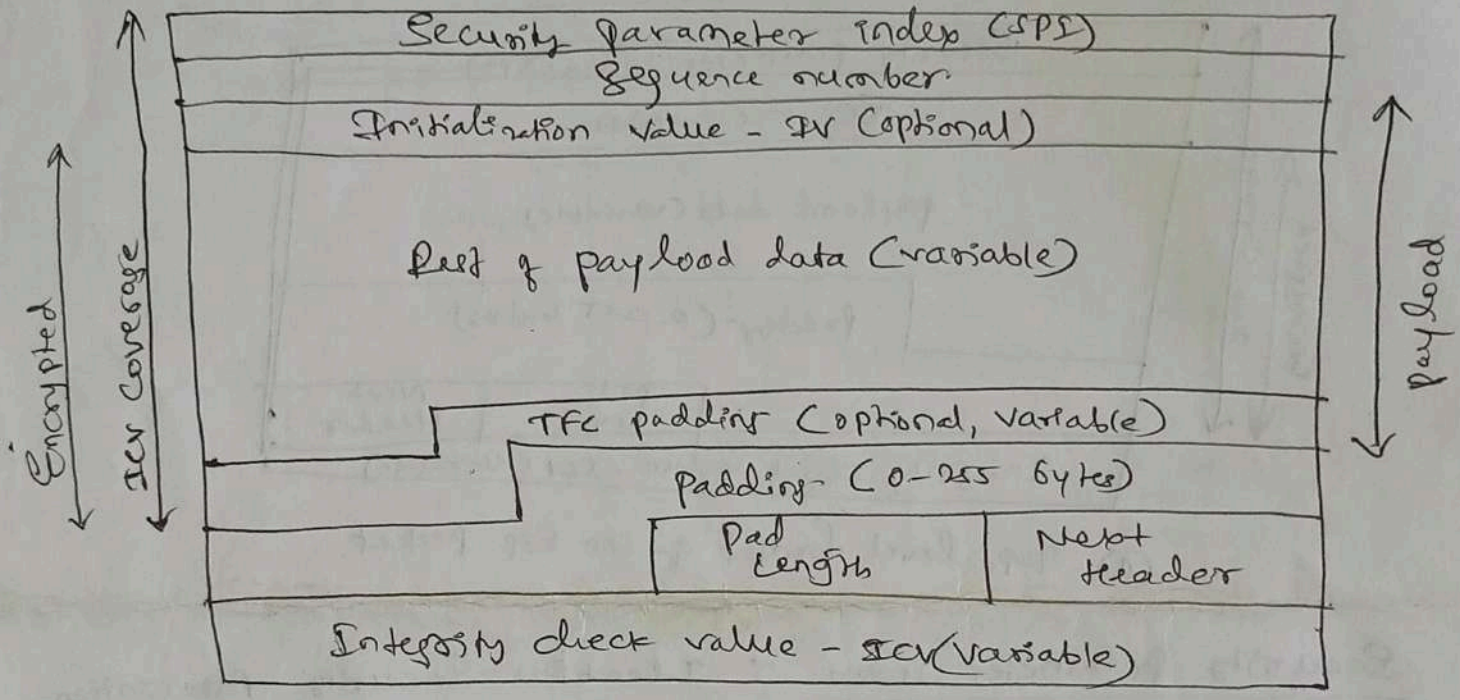
Padding (0 to 255 bytes) -

Pad length (8 bits) - Indicates the no. 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 header in that payload (eg an extension header in IPV6 or an

upper layer protocol such as TCP)

Integrity check value (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.



(b) Substructure of payload data

Two additional fields may be present in the payload. An initialization value (IV), or nonce if present if this is required by the encryption or authenticated encryption algorithm used for ESP. If tunnel mode being used, then the IPsec implementation may add traffic flow Confidentiality (TFC) padding after the payload data & before the padding field.

padding - The padding field serves several purposes:

- If an encryption algorithm requires the plaintext to be multiple of some no. of bytes, the padding field is used to expand the plaintext to the required length.

→ The Esp format requires that the pad length & Next header fields be right aligned within a 32-bit word.

→ Additional padding may be added to provide partial traffic flow Confidentiality by concealing the actual length of the payload.

Anti-Replay Service A replay attack is one in which an attacker obtains a copy of an authenticated packet & later transmits it to the intended destination. The sequence number field is designed to thwart such attacks.

→ when a new SA is established, the sender initializes a sequence number Counter to zero. Each time that a packet is sent on this SA, the sender increments this Counter & places the value in the sequence number field. Thus, the first value to be used is 1. If anti-replay is enabled, the sender must not allow the sequence number to cycle past $2^{32}-1$ back to zero. If the limit $2^{32}-1$ is reached, the sender should terminate this SA & negotiate a new SA with a new key.

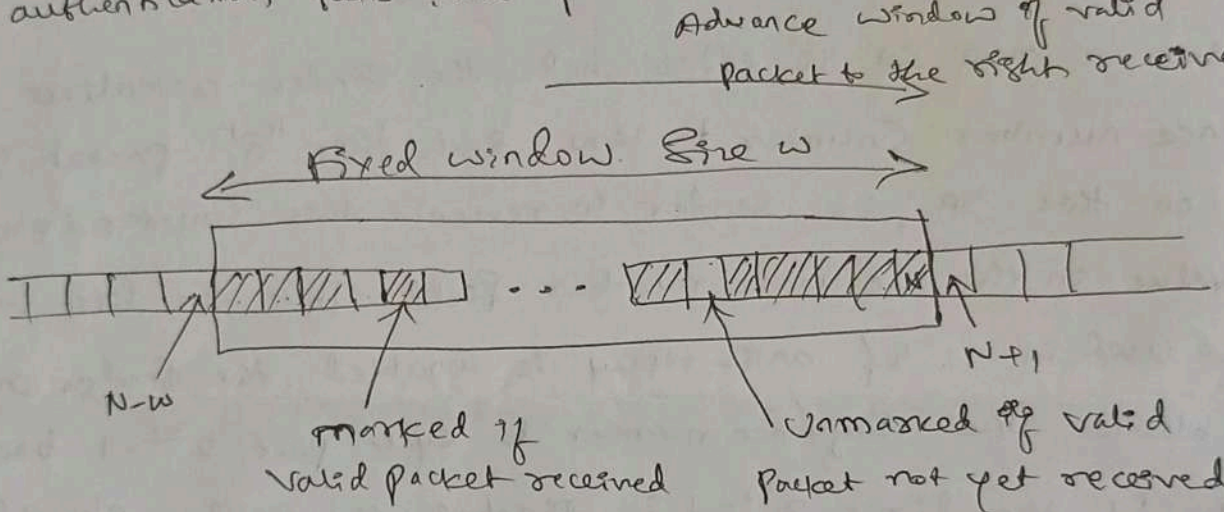
The Ipsec authentication document dictates that the receiver should implement a window of size w, with a default of w = 64. The right edge of the window represents the highest sequence number, N, so far received for a valid packet. For any packet with a sequence number in the range from N-w+1 to N that has not been Correctly received the Corresponding slot in the window is marked. Inbound processing proceeds as follows when a packet is received:

1. If the received packet falls within the window & is new

the MAC is checked. If the packet is authenticated, the corresponding slot in the window is marked.

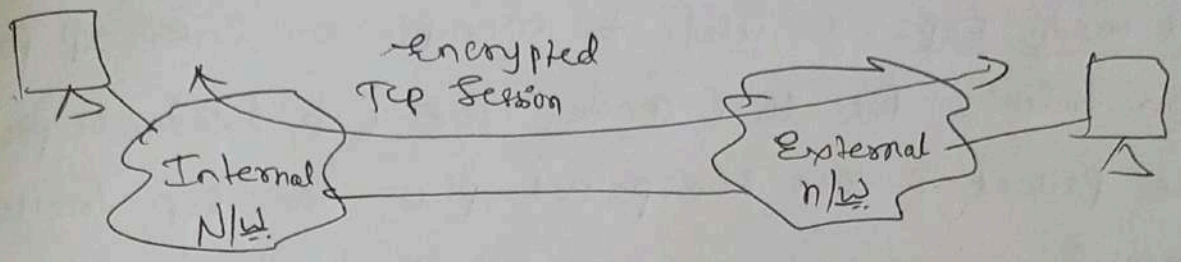
2. If the received packet is to the right of the window & is new, the MAC is checked. If the packet is authenticated the window is advanced so that this sequence number is the right edge of the window, & the corresponding slot in the window is marked.

3. If the received packet is to the left of the window or if authentication fails, the packet is discarded.

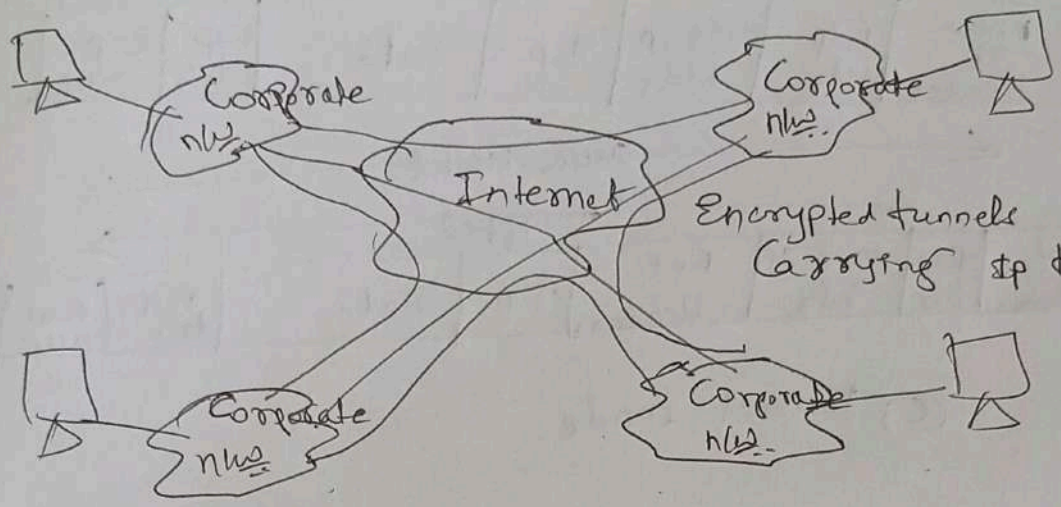


[Anti-Replay mechanism]

* Transport & Tunnel modes Below figure shows two ways in which the IPsec ESP service can be used. In the upper part of the figure encryption (and optionally authentication) is provided directly between two hosts. Figure b shows how tunnel mode operation can be used to set up a virtual private nlw. In this example an org. has four private nlws interconnected across the Internet. Hosts on the internal nlws use the Internet for transport of data but do not interact with other Internet-based hosts.

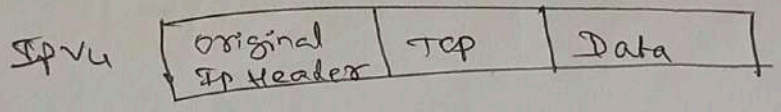


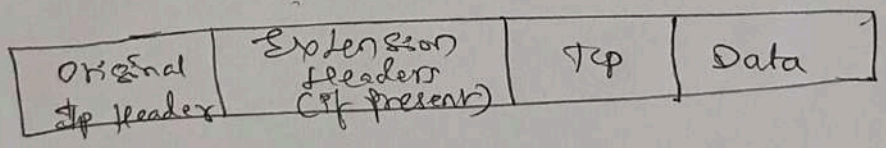
a) Transport-level Security



b) A virtual-private n/w. via tunnel mode.

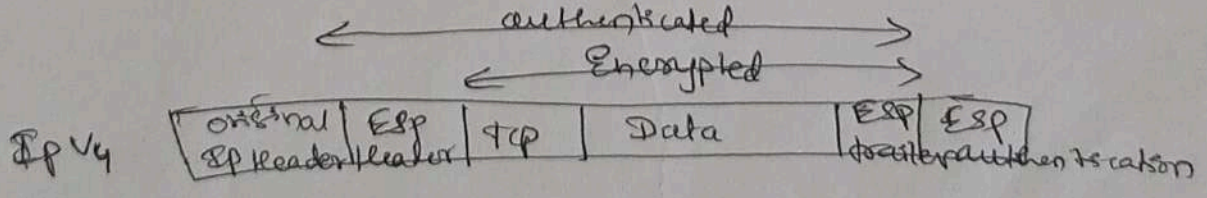
<> We use packet formats of below as starting point.

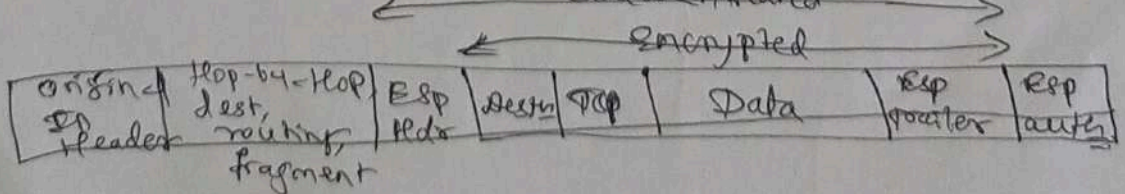




(a) Before applying ESP

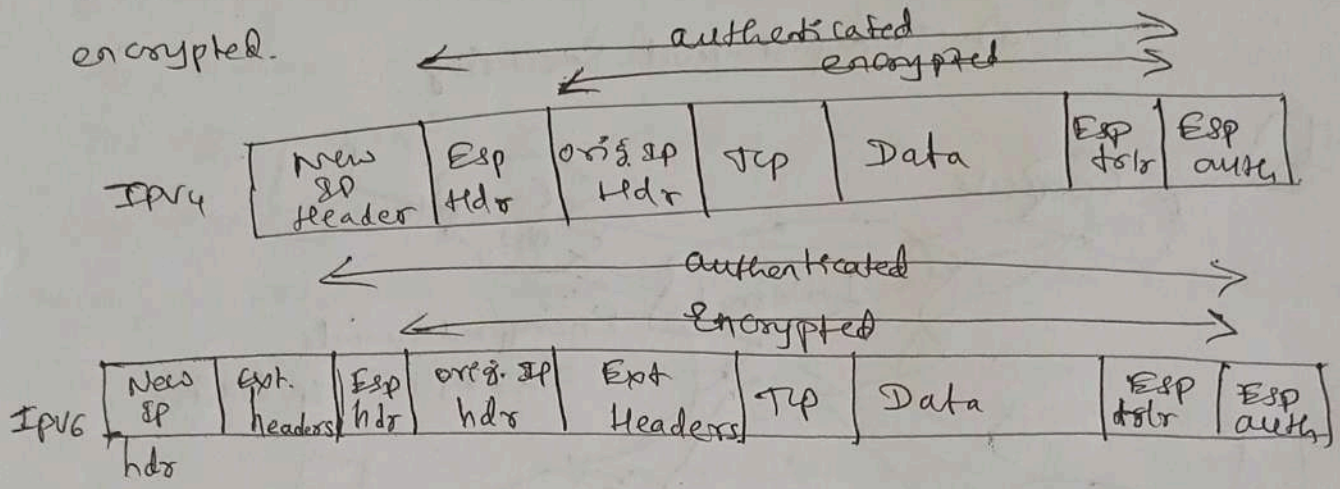
Transport mode ESP - is used to encrypt and optionally authenticate the data carried by IP as shown below.





(b) Transport mode

Tunnel mode ESP - is used to encrypt an entire IP packet shown below. For this mode, the ESP header is prefixed to the packet & then the packet plus the ESP trailer is encrypted.



(c) Tunnel mode