"Heaven's Light is Our Guide"

RAJSHAHI UNIVERSITY OF ENGINEERING & TECHNOLOGY

RUET

Name : Md. Ismail

Subject : CSE-4215, Network Security Year : ○1st ○2nd ○3rd ◉4th

Roll No : 1503091 Semester: ○1st ○2nd ○3rd ○4th ○5th ○6th ○7th ◉8th

Department : CE EE EEE ME CSE ETE IPE URP GCE MTU ARCH

1E

→ Ch-11

→ Ch-12

→ Ch-10

→ Ch-13

→ Ch-14

→ Ch-9

→ Ch-3

Chapter 11
Message Authentication and Hash Functions

- Message authentication: A procedure used to verify the integrity of a message
- Types of functions used to produce an authentication:
  - 3 classes
    - → message encryption
      - → ciphertext of the entire message
    - → message authentication code (MAC)
      - → A function of the message & a secret key
        - produces → fixed length value
    - → Hash function
      - → function that maps a message of any length into a fixed length

Message Encryption

→ provide a measure of authentication

2 types
- → Symmetric encryption
- → public-key encryption

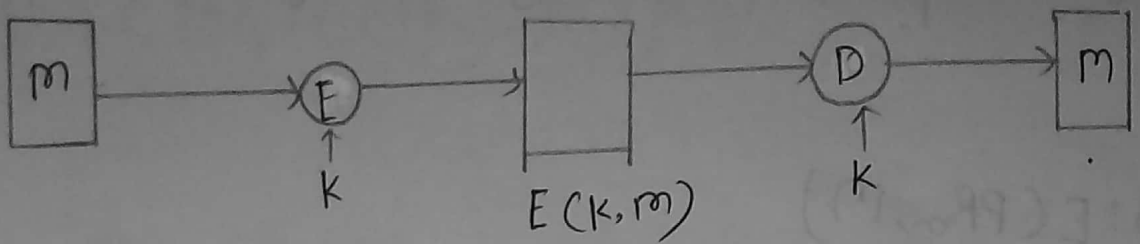
Symmetric encryption

$A \rightarrow B : E(K, M)$

→ provide confidentiality
- → only A and B share K
- → No other party can recover the plaintext of the message.

→ provide a degree of authentication
- → could come only from A
- → has not been altered in transmission

→ Doesn't provide digital signature
- → Sender could deny message
- → Receiver " forge "



Public key encryption

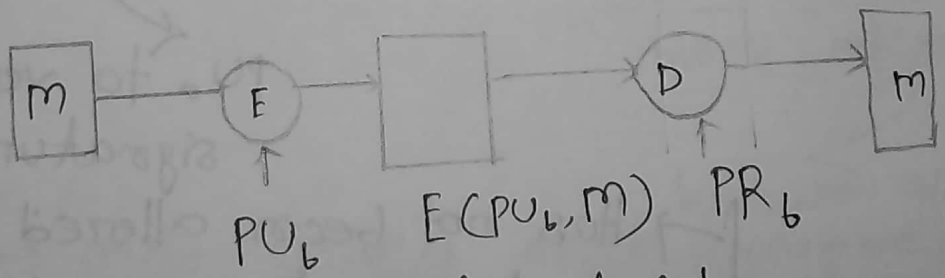
3 types

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

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

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

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

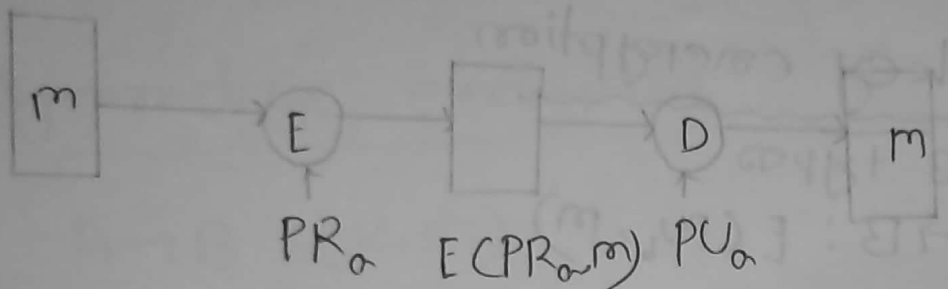


→ Provide confidentiality
→ only B has PR_b to decrypt

→ Provide no authentication
→ any party could use PU_b to encrypt message & claim to be A

→ doesn't ~~pf~~ provide digital signature.

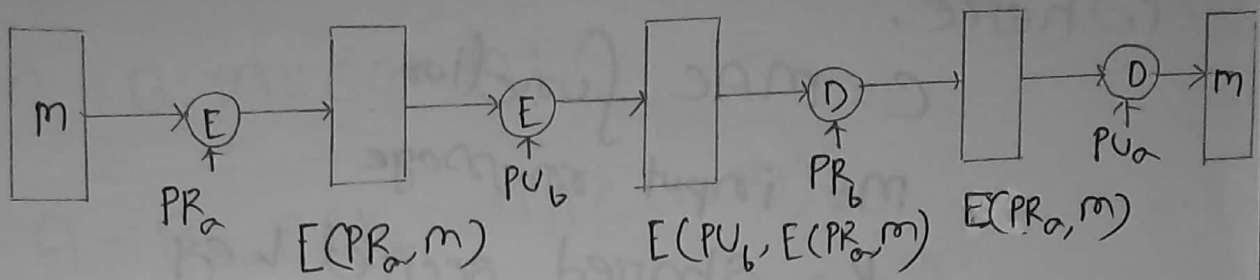
A → B : E(PR$_a$, M)



→ Provides authentication & signature
→ Only A has PR$_a$ to encrypt
→ ~~Has not~~
→ Any party can use
→ PU$_a$ to verify signature.
→ Has not been altered in transit
→ Require some formatting redundancy

→ doesn't provide confidentiality

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



→ Provides confidentiality because of PU_b

→ " authentication and signature because of PR_a

→ appended to the message.

☐ Message Authentication Code (MAC)

→ use a secret key

→ to generate a small fixed-size block of data

→ known as MAC

→ also called cryptographic checksum

→ two communicating parties say

→ share a common secret key, K

MAC = C(K, M)

Where,

C = MAC function

M = input message

K = Shared secret key

MAC = message authentication code.

~~Process~~

Process:

→ A send a message to B

→ A & B share a key, k

→ A calculate MAC

→ ~~MAC~~ MAC append with ~~M~~ message, M

→ B received message & MAC

→ B calculate MAC

→ Compared with received MAC

if calculated MAC = Received MAC

message accepted

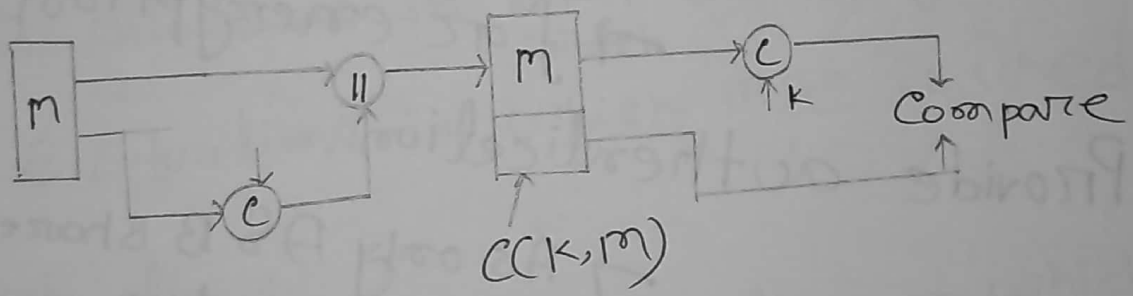
Basic use of MAC

$A \rightarrow B: M \| C(K, M)$

$A \rightarrow B: E(K_2, M \| C(K_1, M))$

$A \rightarrow B: E(K_2, M) \| C(K_1, E(K_2, M))$

$A \rightarrow B: M \| C(K, M)$:

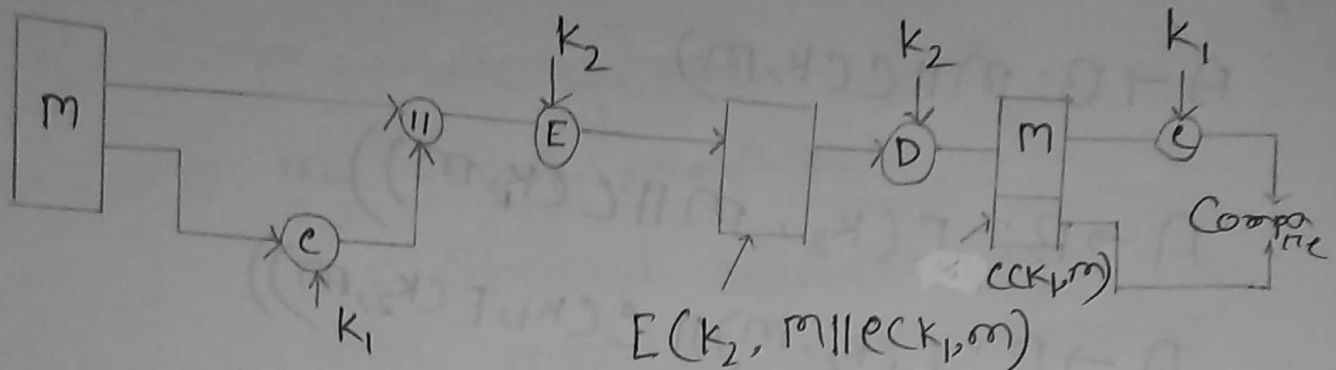


→ Provide authentication
  → Only A and B share K

→ Can't provide
  → confidentiality
  → Signature

message as a whole is transmitted

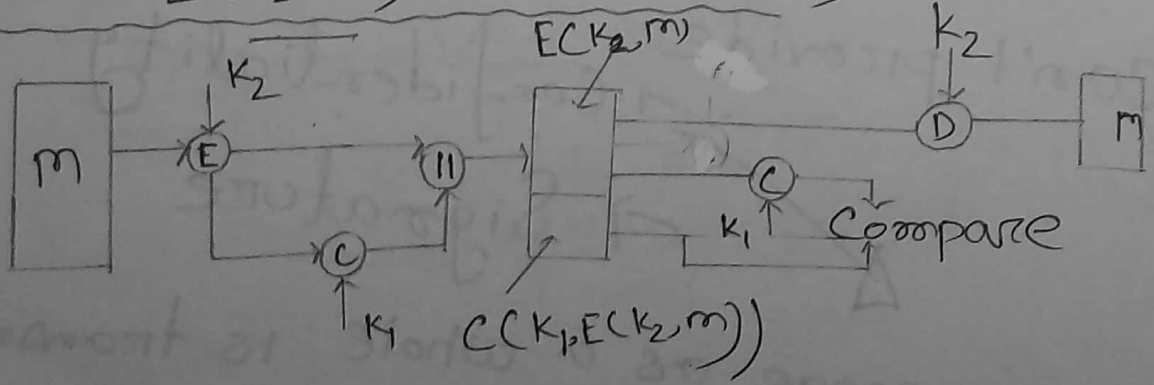
A→B: → Encryption after MAC
$E(K_2, M \| C(K_1, M))$



→ Provide confidentiality
→ As only A & B share K_2
→ for encryption

→ Provide authentication
→ As only A & B share K_1
→ for secret key K_1

→ Encryption before MAC
A→B: $E(K_2, M) \| C(K_1, E(K_2, M))$



→ Provide confidentiality
  → for encryption
  → using K_2

→ " authentication
  → for K_1

→ Can't provide signature

⊞ Reason to use MAC:

~~3 situations in with a message~~ which

3 situations in which MAC is used:

Situation-1.

Number of applications in which the same message is broadcast to number of destinations

Ex:
→ Notification to the users
  ↳ Network is now unavailable
→ Alarm signal in military control center.

Situation-2

An exchange in which

One side has a heavy load

and can't afford the time to decrypt all incoming messages

Situation-3.

Authentication of a computer program in plaintext is an attractive service.

3 other rationales

→ Applications not be concern to keep messages secret,

— but important to authenticate messages

Ex: Simple Network Management

Protocol version 3 (SNMPv3)

→ Separation of authentication and confidentiality function.

Ex:

Perform authentication → application layer

confidentiality → transport layer

→ user may wish to prolong the (extend the duration)

the period of protection

NB:

MAC doesn't provide a digital signature.

as both sender & receiver share the same key

Hash Code H(m) / message digest / hash value

A hash function accepts a variable size message m as input.

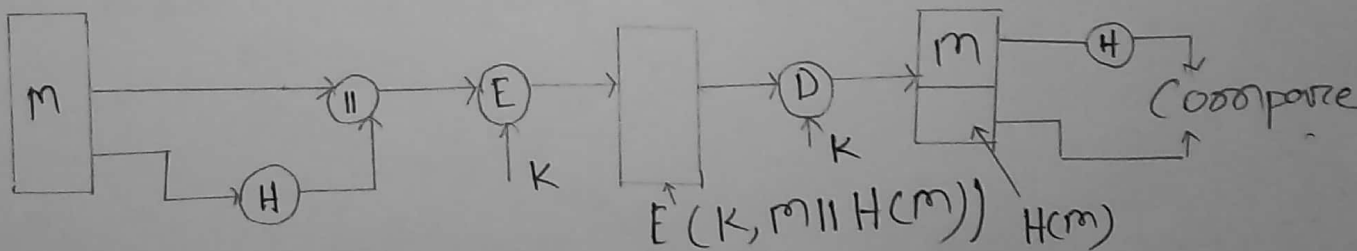
→ produces a fixed-size output
  → referred to as a hash code.

→ Also referred to as
  → message digest
  → hash value

Basic use of hash function

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

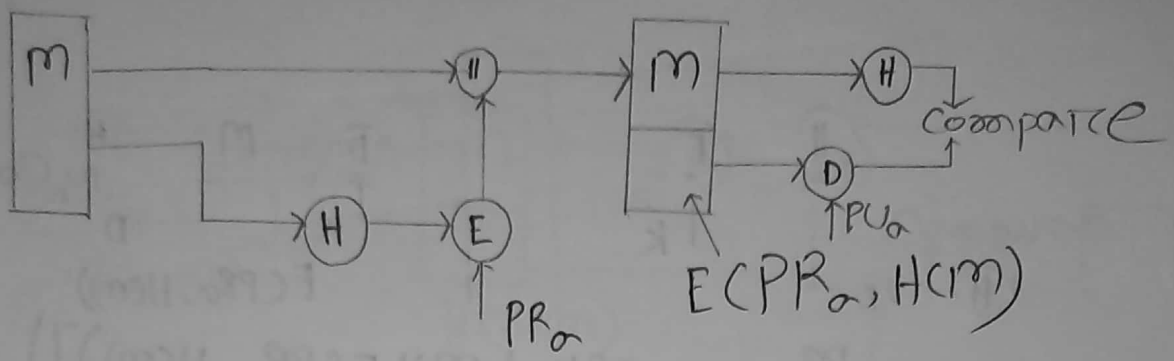
$A \rightarrow B: E(K, [M \| H(M)])$:



→ Provide confidentiality
  → for encryption
  → only A & B share K

→ Provide authentication
  → H(M) is criptographically protected

A → B: M || E(PR$_a$, H(M));

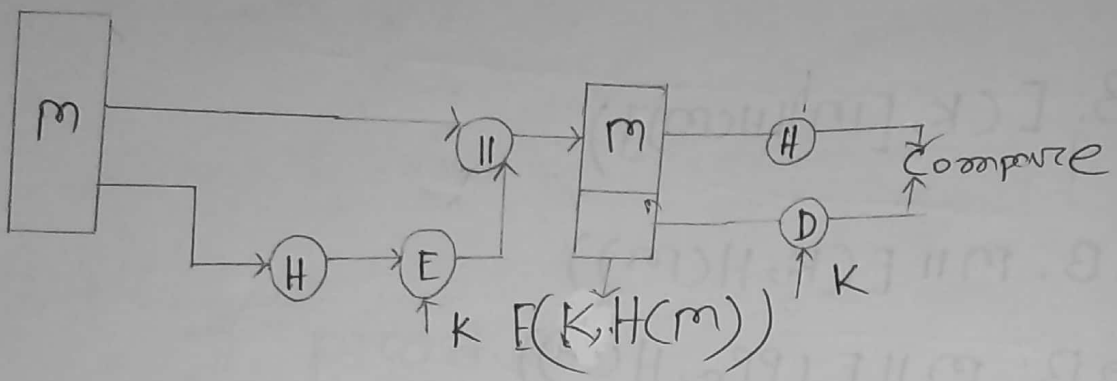


→ Provide authentication
  → H(M) is cryptographically protected

→ Provide digital signature.
  → Only A could create E(PR$_a$, H(M))

→ Can't provide confidentiality

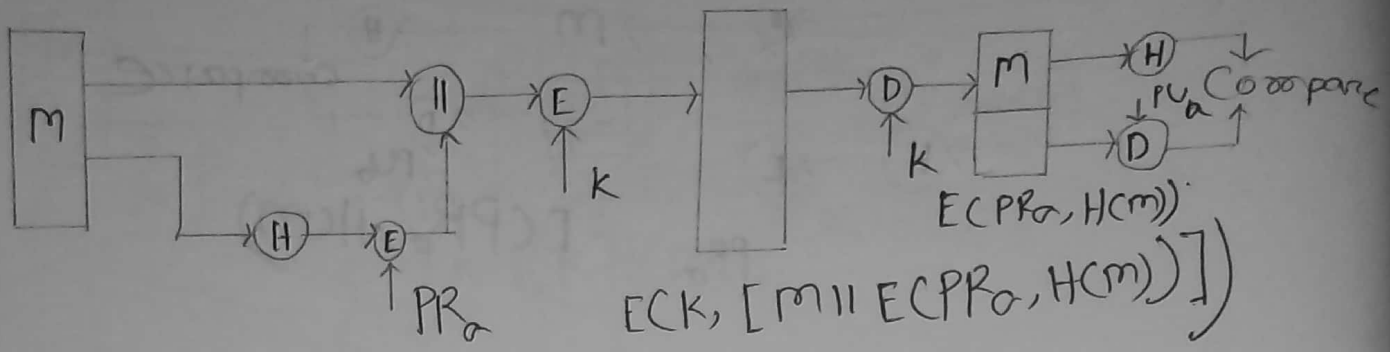
$A \rightarrow B : M \| E(K, H(M))$:



→ Provide authentication
→ H(M) is cryptographically protected

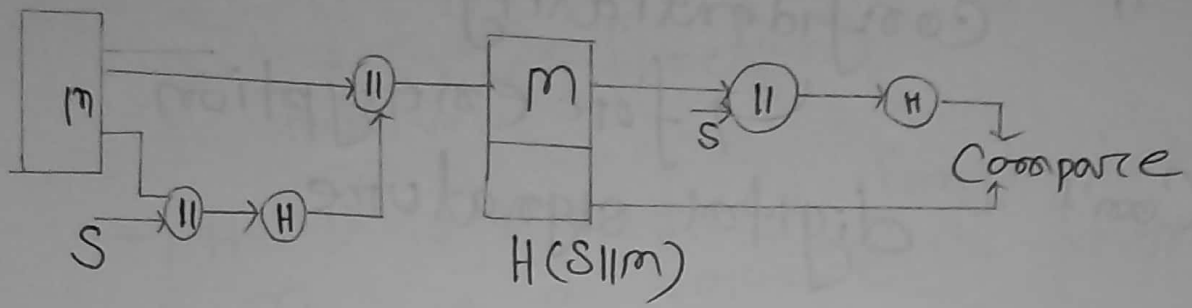
→ Can't provide → confidentiality
→ Signature

A→B: E(K, [M || E(PR$_a$, H(M))])



→ Provide authentication
  → H(M) is cryptographically protected
→ Provide signature
  → only A can encrypt
    E(PR$_a$, H(M))
→ Provide confidentiality

A → B : M || H(M||S)



→ Provide authentication

~~→ Only A & B~~

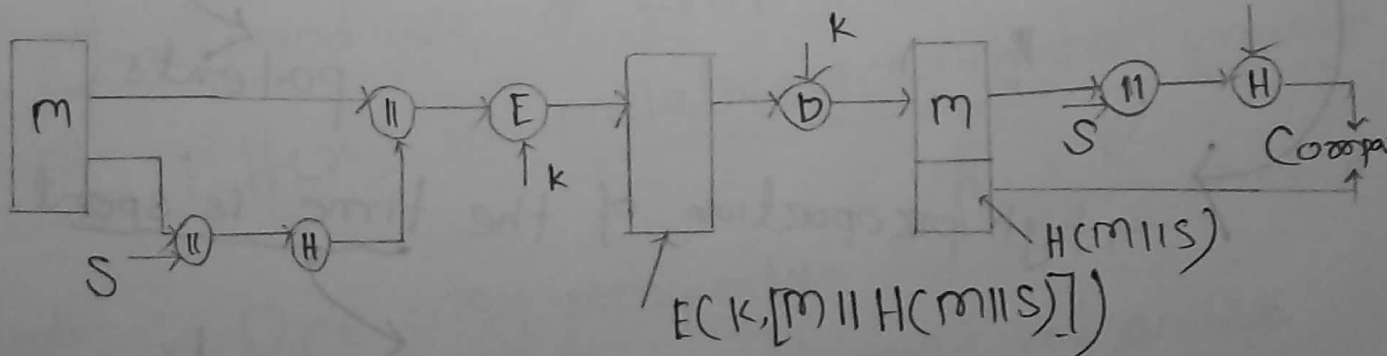
→ for hash function

☆ Hash code is protected

↳ As only A & B know secret value S

~~A → B : M~~

A → B : E(K, [M || H(M||S)])



→ Provide authentication
↳ for hash-function
→ " confidentiality
↳ for encryption
→ Can't " digital signature

Reason to avoid encryption:

→ Encryption s/w is relatively slow
→ " H/w costs are not negligible
→ " H/w is optimized toward
↳ large data sizes
→ " algorithm may be covered by
↳ patents

→ high proportion of the time is spent
↳ initialization
↳ invocation overhead

Properties of hash function

$h = H(M)$

h ↓ hashvalue

H ↓ hash function

→ $H(M)$ is the fixed length hash value
→ M is a variable " message.
→ function of all the bits of the message provides an error detection capability

Purpose:

→ Produce a "fingerprint" of
→ a file
→ message
→ or other block of data

Properties/Requirements:

6 properties

requirements for practical application

→ applied to a block of data
→ produces a fixed-length output
→ $H(x)$ is relatively easy to compute for any given x.

hash function → message authentication

→ **One-way property:**
it is computationally infeasible to find x such that $H(x) = h$

→ Weak collision resistance:

It is computationally infeasible to find $y \neq x$ such that $H(y) = H(x)$

→ Strong collision resistance:

infeasible to find any pair (x, y) such that $H(x) = H(y)$.

how resistant the hash function.

to the birthday attack

Chapter-12
Hash and MAC Algorithm

SHA

- developed by NIST along with NSA
- published as a federal information processing standard (FIPS-180)
  - in 1993
  - SHA-0
- A revised version was issued as FIPS 180-1.
  - in 1995
  - referred to as SHA-1
- based on MD4
- FIPS 180-2
  - in 2002
  - has 3 new versions of SHA
    - SHA-256
    - SHA-384
    - SHA-512

→ it has following versions

- → SHA 0
- → SHA 1
- → SHA 2
- → SHA 3

SHA-1

→ produces 160-bit hash values

Steps

Step-1: Padding bits

Step-2: Append length

Step3: Divide the input into 512-bit blocks

Step4: Initialize hash buffer / chining variable

Step5: Compression / Process block

Step6: Output

Step-1: Padding bits: ☆☆

→ Given an m bit message

→ a single bit "1" is appended as $(m+1)^{th}$ bit

→ $(448-(m+1)) \bmod 512$ zeros bits are appended.

$512 - (448-(L+1))$

$512 - 448 + (L+1)$

$64 + (L+1)$

Step2: A 64 bit representation of original length of m is appended.

Step3: Division into Blocks:

The result is divided into 512- bit blocks. denoted by

$M_1, M_2 \cdots M_n$

Step4: Initialize hash buffer/ Chining variable:

32 -bit words A, B, C, D, E used to keep the 160-bit chining value hi.

$A = (67452301)_H$

$B = (EFCDAB89)_H$

$C = (98BADCFE)_H$

$D = (10325476)_H$

$E = (C3D2E1F0)_H$

Step 1: Compression:

For each block the compression function $h_i = H(h_{i-1}, M_i)$ is applied on previous value of $h_{i-1} = (A, B, C, D, E)$ & message block, M_i

Step 5: Output:

The hash value is the 160-bit value $\boxed{h_i = (A, B, C, D, E)}$

Compression function H of SHA-1

→ Divide M_i into 16 32-bit words:

$$W_0, W_1 \text{ --- } W_{15}$$

→ For $t = 16$ to 79

$$W_t = S^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$$

→ ~~Set~~ Store hash variable

$$(A, B, C, D, E) \leftarrow H_{i-1}$$

→ For $t = 0$ to 79 do

→ $T = S^5(A_t) + f_t(B_t, C_t, D_t) + E_t + W_t + K_t$

→ $E_{t+1} = D_t$, $D_{t+1} = C_t$, $C_{t+1} = S^{30}(B_t)$

$B_{t+1} = A_t$, $A_{t+1} = T$

→ Output $A = A_0 + A_{80}$, $B = B_0 + B_{80}$.

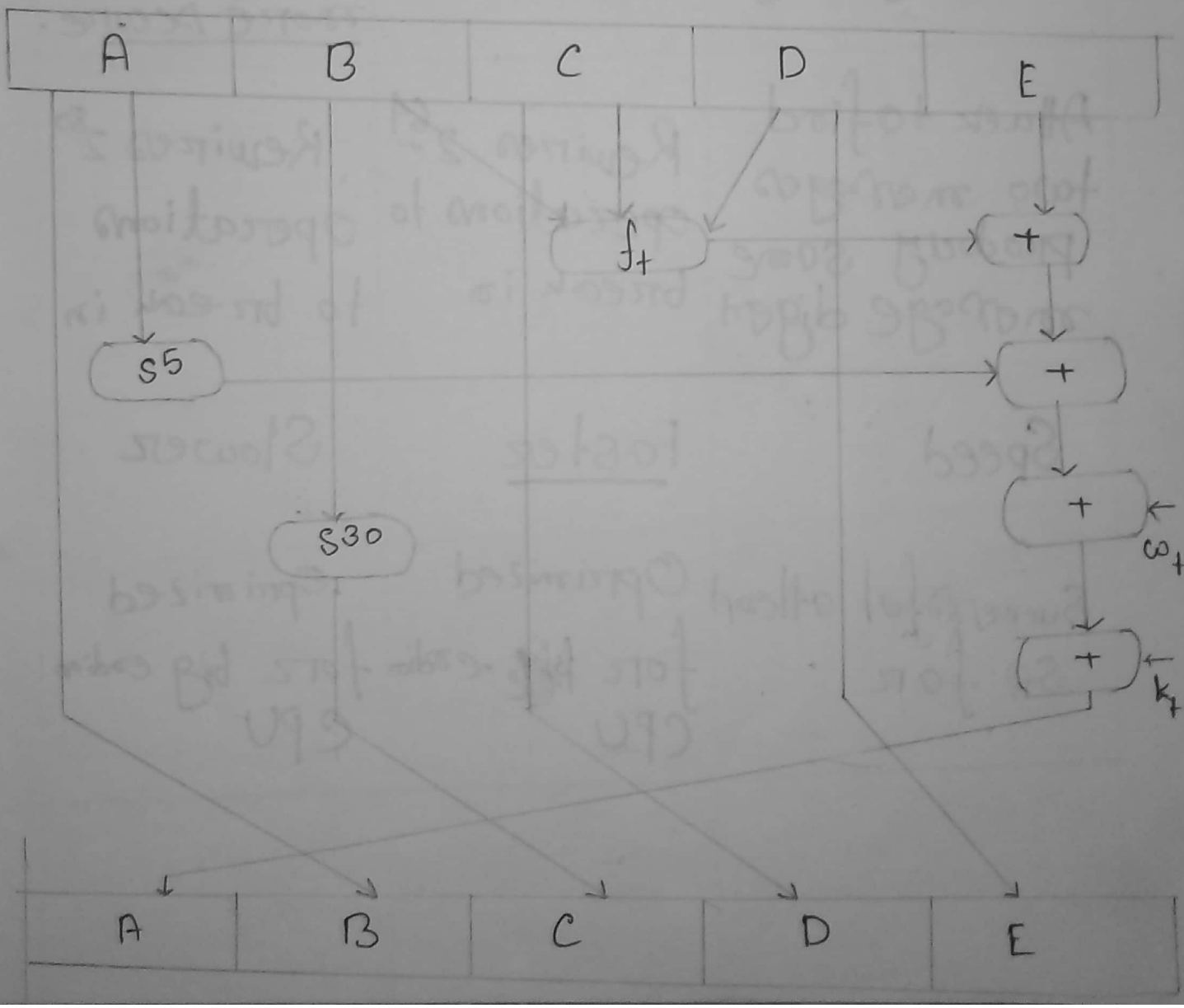
$C = C_0 + C_{80}$, $D = D_0 + D_{80}$

$E = E_0 + E_{80}$ (modulo 2^{32})

→ The function f_t and values K_t used above are:

**

Stage-1	$0 \leq t \leq 19$	$f_t(B,C,D) = (B \cap C) \vee ((\neg B) \cap D)$	$k_t = 5A827999$
Stage-2	$20 \leq t \leq 39$	$f_t(B,C,D) = B \oplus C \oplus D$	$k_t = 6ED9EBA1$
Stage-3	$40 \leq t \leq 59$	$f_t(B,C,D) = (B \cap C) \vee (B \wedge D) \vee (C \wedge D)$	$k_t = 8F1BBCDC$
Stage-4	$60 \leq t \leq 79$	$f_t(B,C,D) = B \oplus C \oplus D$	$k_t = CA62C1D6$



MD5 vs SHA-1

	MD5	SHA-1
message digest length in bits	128	160
Attack to find the original message given a message digest	Requires 2^{128} operations to break in	Requires 2^{160} operations to break in. therefore more secure.
Attack to find two messages producing some message digest	Requires 2^{64} operations to break in	Requires 2^{80} operations to break in
Speed	Faster	Slower
	Optimized for little-endian CPU	Optimized for big endian CPU

Chapter-10
key management; Other Public-key cryptosystems

Primitive root:

If a is a primitive root of a prime number p. then the numbers

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

are distinct and consist of the integer from 1 to p-1.

Diffie-Hellman key exchange:

→ first published public-key algorithm.

defined public-key cryptography.

→ purpose: Enable two users to securely exchange a key used for subsequent encryption of messages.

Description of algorithm:

→ 2 global public elements

→ q prime number q

→ α, $\alpha < q$ and α a primitive root of q.

→ User A key generation.

→ Select private x_A $x_A < q$

→ Calculate public Y_A $Y_A = \alpha^{x_A} \bmod q$

→ User B key generation.

→ Select private x_B $x_B < q$

→ Calculate public Y_B $Y_B = \alpha^{x_B} \bmod q$

→ Calculation of secret key by User A

$$K = (Y_B)^{x_A} \bmod q$$

→ Calculation of secret key by User B

$$K = (Y_A)^{x_B} \bmod q$$

Show that A and B produce identical results:

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

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

$$= (\cancel{X_B})$$

$$= (\alpha^{X_B})^{X_A} \bmod q \quad [\text{by the rules of modular arithmetic}]$$

$$= \alpha^{X_B X_A} \bmod q$$

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

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

$$= (Y_A)^{X_B} \bmod q$$

(Shown)

referred to as discrete logarithm of b base a, mod P.

Unique exponent:

For any integer b and a primitive root a of a prime number P, we can find a unique exponent i such that,

$b \equiv a^i \pmod P$ where $0 \leq i \leq (P-1)$

$= 24^2$

$= (73^4 \bmod 353 \times 96 \times 69) \bmod 353$

$= ((34^2 \bmod 353) \times 96 \times 69) \bmod 353$

$= (97 \times 96 \times 69) \bmod 353$

2017 3(e)

Given that, $q = 13$

$\alpha = 2$

i

$2^1 \bmod 13 = 2$

$2^2 \bmod 13 = 4$

$2^3 \bmod 13 = 8$

$2^4 \bmod 13 = 3$

$2^5 \bmod 13 = 6$

$2^6 \bmod 13 = 12$

$2^7 \bmod 13 = 11$

$2^8 \bmod 13 = 9$

$2^9 \bmod 13 = 5$

$2^{10} \bmod 13 = 10$

$2^{11} \bmod 13 = 7$

$2^{12} \bmod 13 = 1$

As for $k, k \le q-1$ and $2^k \bmod q$ is distinct and $1 \le m \le q-1$

$\therefore$ 2 is a primitive root of 13.

ii

$X_A = 3$

$Y_A = ?$

$Y_A = \alpha^{X_A} \bmod q$

$= 2^3 \bmod 13$

$= 8$ Ans

iii

$Y_B = 3$

$x_B = ?$

$Y_B = \alpha^{x_B} \bmod 2$

$\Rightarrow 3 = 2^{x_B} \bmod 13$

$\therefore x_B = 4$ [from i]

Ans

iv

$K = (Y_n)^{x_B} \bmod 2$

$= 8^4 \bmod 13$ [from i, ii, iii]

$= ((8^2 \bmod 13)(8^2 \bmod 13) \bmod 13)$

$= (12 \times 12) \bmod 13$

$= 1$

Or, $K = (Y_B)^{x_A} \bmod 2$

$= 3^8 \bmod 13$ [from i, ii, iii]

$= 27 \bmod 13$

$= 1$

$\therefore$ Shared key $k = 1$ Ans

2016/6(a), Prob-10.2

Given that

$q = 11$

$\alpha = 2$

i

$2^1 \bmod 11 = 2$
$2^2 \bmod 11 = 4$
$2^3 \bmod 11 = 8$
$2^4 \bmod 11 = 5$
$2^5 \bmod 11 = 10$
$2^6 \bmod 11 = 9$
$2^7 \bmod 11 = 7$
$2^8 \bmod 11 = 3$
$2^9 \bmod 11 = 6$
$2^{10} \bmod 11 = 1$

As for $1 \le x \le q-1$ $m = 2^x \bmod q$ is distinct and $1 \le m \le q-1$.

$\therefore$ 2 is a primitive ~~of~~ root of 11

(Shown)

ii $Y_A = 9$

$X_A = ?$

$Y_A = \alpha^{X_A} \bmod q$

$\Rightarrow 9 = 2^{X_A} \bmod 11$

$\therefore X^A = 6$ [from i]

Ans

iii

$Y_B = 3$

Secret key, $K = ?$

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

$= 3^6 \bmod 11$

$= ((3^3 \bmod 11)(3^3 \bmod 11)) \bmod 11$

$= 25 \bmod 11$

$= 3$ Ans

Again,

$Y_B = 3$

$\Rightarrow \alpha^{X_B} \bmod q = 3$

$\Rightarrow 2^{X_B} \bmod 11 = 3$

$\therefore X_B = 8$ [from i]

Now, $K = (Y_A)^{X_B} \bmod q$

$= (9^8) \bmod 11$ [from i, ii]

$= ((9^2 \bmod 11)(9^2 \bmod 11)(9^2 \bmod 11)(9^2 \bmod 11)) \bmod 11$

$= (4 \times 4 \times 4 \times 4) \bmod 11$

$= 256 \bmod 11 = 3$ Ans

Problem 10.1

Given that,

$q = 71$

$\alpha = 7$

a) $x_A = 5$

$Y_A = \alpha^{x_A} \bmod q$

$= 7^5 \bmod 71$

$= 16807 \bmod 71$

$= 51$ Ans

b) $x_B = 12$

$Y_B = \alpha^{x_B} \bmod q$

$= 7^{12} \bmod 71$

$= (7^5 \bmod 71)(7^5 \bmod 71)(7^2 \bmod 71) \bmod 71$

$= (51 \times 51 \times 49) \bmod 71$

$= 4$ Ans

c) $K = (Y_A)^{x_B} \bmod q$

$= (51)^{12} \bmod 71$

$=$

d) $K = (Y_B)^{x_A} \bmod q$

$= (4)^5 \bmod 71 = 30$ Ans

Security of Diffie Hellman key exchange

- easy to perform $\alpha^x \bmod q \xrightarrow{\text{easy}} ?y$
- hard to reverse $\alpha^{?} \bmod q \xrightarrow{\text{hard}} y$
- Brute force to determine the secret key → difficult

Man-in-the-Middle Attack:

①

→ Darth prepares for the attack

→ generate 2 random private key x_{D1}, x_{D2}

→ calculate public keys Y_{D1}, Y_{P2}

→ Alice transmits Y_A to Bob

→ Darth intercepts Y_A
→ " transmit Y_{D1} to Bob.
~~→ Bob calculate $K_2 =$~~
→ " calculate $K_2 = (Y_A)^{x_{D2}} \bmod 2$

→ Bob receive Y_{D1}
→ " calculate $K1 = (Y_{D1})^{x_B} \bmod 2$
→ Bob transmit Y_B to Alice.

→ Darth intercept Y_B
→ " transmit Y_{D2} to Alice
→ " calculate $K_1 = (Y_B)^{x_{D1}} \bmod 2$

→ Alice receive Y_{D2} and
→ " calculate $K_2 = (Y_{D2})^{X_A} \bmod 2$

Communication between Bob and Alice:

1) Alice send an encrypted message $m : E(K_2, m)$

2) Darth intercepts the encrypted message

decrypts it to recover

3) Darth sends Bob $E(K_1, m)$ or $E(K_1, m)$

Soln :

→ use of digital signatures
→ " of public key certificates.

Chapter-13
Digital signature & Authentication Protocol

Authentication process:

Consists of 2 steps.

1. Identification step → presenting an identifier to the security system
2. Verification step → presenting or generating authentication information.
   → corroborate the binding between the entity and the identifier

Means of authenticating a user's identity

4 general means:

→ Something the individual knows
password answers to a prearranged
PIN. set of questions

→ Something the individual processes
referred as a token cryptographic keys. Smartcard.
electronic keycards. physical key

→ Something the individual is

recognition by → fingerprint
→ retina
→ face

→ " " " does

recognition by → voice pattern
→ handwriting
characteristics
→ typing rhythms.

Problem of authenticated key exchange:

2 issues
- confidentiality → to prevent masquerade
- timeliness → to protect session key
  - important because of the threat of message replay
  - → to prevent replay attack

Replay attack:

A valid message is copied and later resent.

Example:
- Simple replay → copies a message & replays it later
- Repetition that can be logged → replay a timestamped message within the valid time window.

Original message → replaced with a new message → Backward replay without modification → replay back to the message sender

" " can't be detected

Soln of replay attack:

→ ~~attac~~

→ attaching a sequence number to each message

→ Timestamp

→ Challenge/Response

Sequence number

→ message is accepted

↳ only if its sequence number is in the proper order

requires

→ each party to keep track of the last sequence number for each element

→ impractical

Timestamp:

→ clock among the various participants be synchronized

Challenge / response

→ Using unique nonce

4. Symmetric Encryption Approach:

→ 2 level hierarchy of symmetric encryption keys can be used

→ A KDC is used

master key: → used for distributing session key

Each party in the network shares a secret key with the KDC

session key:

key that is used for a short time over a connection between two parties.

Types of protocol:

→ Needham and Schroeder protocol

→ Denning modification

→ Corrected protocol

#Needham and Schroeder protocol

$A \rightarrow KDC : ID_A \| ID_B \| N_1$

$KDC \rightarrow A : E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$

$A \rightarrow B : E(K_b, [K_s \| ID_A])$

$\therefore \rightarrow A : E(K_s, N_2)$

$A \rightarrow B : E(K_s, f(N_2))$

→ everyone has shared secret key with KDC
→ KDC generates session keys

Problem:

If someone can crack one K_s, he can replay 3, block 4.

→ masquerade as A.
(pretend)

Denning modification protocol:

$A \rightarrow KDC$: $ID_A \| ID_B$

$KDC \rightarrow A$: $E(K_a, [K_s \| ID_B \| T \| E(K_b, [K_s \| ID_A \| T])])$

$A \rightarrow B$: $E(K_b, [K_s \| ID_A \| T])$

$B \rightarrow A$: $E(K_s, N_2)$

$A \rightarrow B$: $E(K_s, f(N_2))$

→ includes timestamp
↳ session key has only just been generated.

→ can verify timeliness by checking
$|Clock - T| < \Delta t_1 + \Delta t_2$

→ Needs synchronized clock.

Problem:

can occur suppress-replay attack

Sol^n:

→ parties regularly check their clocks
against the KDC's clock

→ handshaking protocols ~~use~~
↳ using nonces

Connected protocol:

$A \rightarrow B : ID_A \| N_a$

$B \rightarrow KDC : ID_B \| N_b \| E(K_b, [ID_A \| N_a \| T_b])$

$KDC \rightarrow A : E(K_a, [ID_B \| N_a \| K_S \| T_b]) \| E(K_b, [ID_A \| K_S \| T_b])$

$A \rightarrow B : E(K_b, [ID_A \| K_S \| T_b]) \| E(K_S, N_b)$

Next time:

$A \rightarrow B : E(K_b [ID_A \| K_S \| T_b]) \| N_a'$

$B \rightarrow A : ~~E(A~~ N_b' \| E(K_S, N_a')$

$A \rightarrow B : E(K_S, N_b')$

→ include timestamp and nonce.

→ can setup next connection without KDC

⊞ One-way authentication for Email:

For a message with content M, the sequence is as follows:

$A \rightarrow KDC : ID_A \| ID_B \| N_1$

$KDC \rightarrow A : E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$

$A \rightarrow B : E(K_b, [K_s \| ID_A]) \| E(K_s, M)$

→ Only B / recipient of a message will be able to read it.

→ Provides a level of authentication.

Chapter-14
Authentication Application

Requirements for kerberos:

4 requirements

- Secure → eavesdropper should not be able to obtain necessary info: to impersonate a user
- Reliable
- Transparent → user should not be aware that authentication is taking place, beyond the requirement to enter a password.
- Scalable → system should be capable of supporting large number of clients & servers

kerberos should be highly reliable & employ a distributed server architecture.

Simple Authentication Dialogue:

→ any client can apply to any server for service.

→ an opponent can → pretend to be another client

→ obtain unauthorized privileges on server machines

→ Soln

→ Server must be able to confirm the identities of client

↳ who request service.

→ each server can be required to undertake this task

↳ for each / server interaction

→ alternative: use authentication server (AS)

↳ knows the passw. of all user

↳ stores these in database

share a unique secret key with each server.

C → AS : $ID_C \| P_C \| ID_V$

AC → C : Ticket

C → V : $ID_C \|$ Ticket

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

→ user has supplied the proper password for this user ID

→ user is permitted access to server V

→ V decrypt the ticket

→ user ID in the ticket = unencrypted user ID

Problems:

→ more number of times that a user has to enter a password.

→ plaintext transmission of password.

Soln:

→ avoiding plaintext passwords.

→ introduce a new server, known as TGS

→ minimize the number of times that a user has to enter a password.

Hypothetical scenario:

Once per user logon session:

C → AS: $ID_c \| ID_{tgs}$

AS → C: $E(K_c, Ticket_{tgs})$

Once per type of service:

C → TGS: $ID_c \| ID_v \| Ticket_{tgs}$

TGS → C: $Ticket_v$

Once per service session:

C → V: $ID_c \| Ticket_v$

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

$Ticket_v = E(K_v, [ID_c \| AD_c \| ID_v \| TS_2 \| Lifetime_2])$

Description:

→ Client requests a ticket-granting ticket on behalf of the user

→ AS responds with a ticket that is encrypted with a key
  → that is derived from the user's password.

→ client request a service-granting ticket on behalf of the user.

→ TGS decrypts the incoming ticket
→ verifies the success of decryption by the presence of its ID
→ make sure that the lifetime has not expired.

→ client requests access to a service on behalf of the user.

message containing → user's ID

→ service-granting ticket

* Only one password query per user session
* protection of the user password

Chapter - 9

Public-key Cryptography and RSA

Requirements for RSA:

1. It is possible to find value e, d, n such that

$M^{ed} \bmod n = M$ for all $M < n$

2. Respectively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M < n$

3. Infeasible to determined d given e and n

RSA Algorithm:

Select
3 Steps
→ key generation
→ Encryption
→ Decryption

key generation

→ Select p, q — p and q prime, p ≠ q
→ Calculate $n = p \times q$
→ Calculate $\phi(n) = (p-1) \times (q-1)$
→ Select e — integer; $\gcd(\phi(n), e) = 1 ; 1 < e < \phi(n)$
e is a relative prime of $\phi(n)$
→ Calculate d — $d = e^{-1} \pmod{\phi(n)}$
→ using extended Euclid's algorith
→ public key : $PU = \{e, n\}$
→ private key : $PR = \{d, n\}$

Encryption

plaintext: $m < n$

ciphertext: $c = m^e \bmod n$

Decryption

Ciphertext c

plaintext $m = c^d \bmod n$

Something about RSA:

→ known as (Rivest-Shamir-Adleman) algorithm

→ asymmetric cryptographic algorithm.

→ developed in 1977 at MIT

→ published in 1978

~~→ block cipher~~

→ it is a block cipher

↳ plaintext and ciphertext are integers between 0 and $n-1$

→ Typical size of n is 1024 bits
→ 309 decimal digits

Example-1

Given that,

$p = 17$
$q = 11$
$m = 88$

$\therefore n = pq = 17 \times 11 = 187$

$\phi(n) = (q-1)(p-1)$
$= (17-1)(11-1)$
$= 16 \times 10 = 160$

Now select e such that e is relatively prime to $\phi(n)$.

$\therefore$ i.e. gcd $(\phi(n), e) = 1$

$e = 7$ is chosen

$\therefore$ $d = e^{-1} \bmod (\phi(n))$
$= e^{-1} \bmod 160$
$= 7^{-1} \bmod 160$ and $d < 160$

b d = 23. because $23 \times 7 = 161$

$= (1 \times 160) + 1$,

$1 = 7 + 6(-1)$

$= 7 + [160 + 7(-22)](-1)$

$= 7 + 7(22) - 160$

$= 7 \times (23) - 160$

$\therefore PU = \{e, n\} = \{7, 187\}$

$PR = \{d, n\} = \{23, 187\}$

$\therefore m = 88$

$C = m^e \bmod n$

$= 88^7 \bmod 187$

$= (88^2 \bmod 187 \times 88^2 \bmod 187 \times 88 \bmod 187) \bmod 187$

$= (77 \times 77 \times 77 \times 88) \bmod 187$

$= 77^2 \bmod 187$

$= (77^3 \times 88) \bmod 187$

$= (77^3 \bmod 187 \times 88 \bmod 187) \bmod 187$

$= (66 \times 88) \bmod 187$

$= 11$ Ans

$m = c^d \bmod n$

$= 11^{23} \bmod 187$

$= (11^4 \times 11^4 \times 11^4 \times 11^4 \times 11^4 \times 11^3) \bmod 187$

$= (11^4 \bmod 187 \times 11^4 \bmod 187 \times 11^4 \bmod 187 \times 11^4 \bmod 187 \times 11^4 \bmod 187 \times 11^3 \bmod 187) \bmod 187$

$= (55 \times 55 \times 55 \times 55 \times 22 \times 55) \bmod 187$

$= (55^5 \times 22) \bmod 187$

$= (55^2 \bmod 187 \times 55^3 \bmod 187 \times 22 \bmod 187) \bmod 187$

$= (33 \times 132 \times 22) \bmod 187$

$= 88$ Ans

2018- 2(b)

Given that,

$c = 10$

$e = 5$

$n = 35$

$m = c^e \bmod n$

$= 10^5 \bmod 35$

$d = e^{-1} \bmod 35$

$= 5^{-1} \bmod 35$

2018-2(b) / Problem 9.3

Given that

$C = 10$

$e = 5$

$n = 35$

$\therefore n = pq$

Let $p = 5 \quad q = 7$

$\phi(n) = (p-1)(q-1)$

$= (5-1)(7-1)$

$= 4 \times 6$

$= 24$

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

$= 5^{-1} \bmod 24$

$\therefore d = 5$; because $(5 \times 5) = 25$

$= (24 \times 1) + 1$

$= (1 \times 24) + 10$

$1 = 5 + 4(-1)$

$= 5 + [24 - 5(4)]$ (-1)

$= 5 + 5(4) - 24$

$= 5 * (5) - 24$

$m = c^d \bmod n$

$= 10^{25} \bmod 35$

$= (10^4 \times 10^4 \times 10^4 \times 10^4 \times 10^4 \times 10^4 \times 10) \bmod 35$

$= (25 \times 25 \times 25 \times 25 \times 25 \times 25 \times 10) \bmod 35$

$= (25^6 \times 10) \bmod 35$

$= (25^3 \bmod 35 \times 25^3 \bmod 35 \times 10 \bmod 35) \bmod 35$

$= (15^2 \times 15 \times 10) \bmod 35$

$= 10$

$m = c^d \bmod n$

$= 10^5 \bmod 35$

$= 5$ Ans

2017- 4(d)

i Given that,

$n = 10$

$= 5 \times 2$

$\therefore \Phi(n) = (5-1) \times (2-1)$

$= 4 \times 1 = 4$ Ans

ii Given that,

$n = 39$

$= 13 \times 3$

$\therefore \Phi(n) = (13-1) \times (3-1)$

$= 12 \times 2$

$= 24$ Ans

iii) Given that,

$n = 59$

$= 3 \times 19$

$\phi(n) = (3-1) \times (19-1)$

$= 2 \times 18$

$= 36$ Ans

2016-3(c) / problem 9.2

Given that,

i) $P = 7$,

$q = 11$

$e = 17$

$m = 8$

$\therefore n = pq$

$= 7 \times 11 = 77$

$\phi(n) = (p-1) \times (q-1)$

$= (7-1)(11-1)$

$= 6 \times 10 = 60$

ii)

$e = 17$

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

$= 17^{-1} \bmod 60$

$= 53$ as

$(17 \times 53) = 901$

$= (15 \times 60) + 1$

$\therefore PU = \{e, n\} = \{17, 77\}$

$PR = \{d, n\} = \{53, 77\}$

$C = m^e \bmod n$

$= 8^{17} \bmod 77$

$= (8^6 \times 8^6 \times 8^5) \bmod 77$

$= (8^6 \bmod 77 \times 8^6 \bmod 77 \times 8^5 \bmod 77) \bmod 77$

$= (36 \times 36 \times 43) \bmod 77$

$= 57$

Ans

$M = C^d \bmod n$

$= 57^{53}$

$m = e^d \bmod n$

$= 57^{53} \bmod 77$

$= 57^{3 \times 5}$

$= (8^{17} \times 15) \bmod 77$

$= (15 \times 15 \times 15 \times 15 \times 15 \times 8) \bmod 77$

$= (15^5 \times 8) \bmod 77$

$= 8$ Ans

ii/d Given that,

$p = 11$

$q = 13$

$e = 11$

$m = 7$

$\therefore n = pq$

$= 11 \times 13$

$= 143$

$\phi(n) = (p-1)(q-1)$

$= (11-1)(13-1)$

$= 10 \times 12 = 120$

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

$= 11^{-1} \bmod 120$

$= 11$ as $11 \times 11 = 121$

$= (120 \times 1) + 1$

$c = m^e \bmod n$

$= 7^{11} \bmod 143$

$= (7^5 \times 7^5 \times 7) \bmod 143$

$= (7^5 \bmod 143 \times 7^5 \bmod 143 \times 7^{\bmod 143}) \bmod 143$

$= (76 \times 76 \times 7) \bmod 143$

$= (76^2 \bmod 143 \times 7 \bmod 143) \bmod 143$

$= (56 \times 7) \bmod 143$

$= 106$ Ans

$m = e^d \bmod n$

$= 106^{11} \bmod 143$

$= (106^2 \times 106^2 \times 106^2 \times 106^2 \times 106) \bmod 143$

$= (82^5 \times 106) \bmod 143 = (3 \times 3 \times 82 \times 106) \bmod 143$

$= 7$ Ans

b Given that

$p = 5$

$q = 11$

$e = 3$

$m = 9$

$n = pq = 5 \times 11 = 55$ Ans

$\phi(n) = (p-1)(q-1) = (5-1)(11-1) = 4 \times 10 = 40$ Ans

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

$= 3^{-1} \bmod 40$

$= 27$, as $(3 \times 27) = 81 = (40 \times 2) + 1$

Ans

~~$c = m^e$~~

$PU = \{e, n\} = \{3, 55\}$

$PR = \{d, n\} = \{27, 55\}$

$c = m^e \bmod n$

$= 9^3 \bmod 55$

$= 14$ Ans

$m = c^d \bmod n$

$= 14^{27} \bmod 55$

~~$= 17$~~ $m = e^d \bmod n$

$= 14^{27} \bmod 55$

$= (14^3 \times 14^3 \times 14^3 \times 14^3 \times 14^3 \times 14^3 \times 14^3 \times 14^3 \times 14^3) \bmod 55$

$= 49 \bmod 55$

$= (49^3 \times 49^3 \times 49^3) \bmod 55 = (4 \times 4 \times 4) \bmod 55$

$= 9$ Ans

a Given that,

$P = 3$

$q = 11$

$e = 7$

$m = 5$

$\therefore n = pq = 3 \times 11 = 33$ Ans

$\phi(n) = (p-1)(q-1) = (3-1)(11-1)$

$= 2 \times 10$

$= 20$ Ans

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

$= 7^{-1} \bmod 20$

$= 3$ Ans as $(7 \times 3) = 21 = (20 \times 1) + 1$

$\therefore PU = \{e, n\} = \{7, 33\}$

$PR = \{d, n\} = \{3, 33\}$

$C = m^e \bmod n$

$= 5^7 \bmod 33$

$= 14$ Ans

$m = C^d \bmod n$

$= 14^3 \bmod 33$

$= 5$ Ans

e Given that.

$p = 17$

$q = 31$

$e = 7$

$m = 2$

$n = pq = 17 \times 31$

$= 527$ Ans

$\phi(n) = (p-1)(q-1)$

$= (17-1)(31-1)$

$= 16 \times 30$

$= 480$ Ans

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

$= 7^{-1} \bmod 480$

$= 343$ and $(7 \times 343) = 2401$

$= 2(480 \times 5) + 1$

Ans

e

$PU = \{e, n\} = \{7, 527\}$

$PR = \{d, n\} = \{343, 527\}$

$C = m^e \bmod n = 2^7 \bmod 527$

$= 128$ And

One way function:

★★ → calculation of the function is easy

→ " of " inverse is infeasible

→ maps a domain into a range

every function value has a unique inverse

Trap-door one way function:

★★

→ easy to calculate in one direction

→ infeasible to " " other "

→ inverse can be calculated in polynomial time

with the additional information.

Strength of RSA Algorithm:

→

Security of RSA

** Approaches to attacking the RSA:

→ Brute force, involves trying all possible private key

→ mathematical attack

→ Timing attack

→ Chosen ciphertext attack

3 approaches to attacking mathematically

→ Factor n into its two prime factors

→ Determine $\phi(n)$ without determining p and q

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

Constraints on p and q

→ p and q should differ in length by only a few digits.
For a 1024 bit key both p and q should be on the order of magnitude of 10^{75} to 10^{100}

→ Both (p-1) and (q-1) should contain a large prime factor

→ gcd (p-1, q-1) should be small

NB :
if $e < n$ and $d < n^{1/4}$ then
d can be easily determined

Application of RSA / based on the application public-key cryptosystem

→ Encryption/decryption
→ key generation → 2 side co-operate to exchange a session key
→ Digital signature → sender signs a message with its private key

Chapter 3
Block Ciphers and Data Encryption Standard (DES)

Stream cipher:

One that encrypts a digital data stream one bit or one byte at a time.

Example:
autokeyed vigenere cipher
Vernam "

block cipher:

One in which a block of ciphertext is treated as a whole and used to produce a ciphertext block of equal length.

NB: block size 64 or 128 bits is used

→ Symmetric block encryption algorithms are based on Feistel Cipher Structure.

Feistel Cipher Structure:

→ input to the algorithm are
  → a plaintext block of length 2b bits
  → a key K.

→ plaintext block is divided into two halves
  → L_0
  → R_0

→ data pass through n rounds of processing & combine to produce the ciphertext block.

→ each round i has an input L_{i-1} and R_{i-1}
derived from previous round

→ a subkey K_i derived from overall k.

→ a substitution is performed on the left half of the data

→ round function F to the right half of data.
→ taking the exclusive-OR of output of that function
→ parameterized by the round subkey K_i

→ permutation is done by exchanging two left half of data.
half of data.

Parameter and design feature of Feistel structure:

Feistel network depends on followings:

1. Block size:
   - AES uses a 128-bit block size
   - larger block sizes means greater security
   - reduce speed for a given algorithm
2. key size:
   - key size of 128 bits is a common
3. Number of rounds:
   - multiple rounds offer increasing security
   - typical size is 16 rounds
4. Subkey generation algorithm:
5. Round function:

greater complexity means greater resistance to cryptoanalysis

2 other considerations

1) fast sw encryption/decryption.
2) Ease of analysis

History of DES:

→ 1977: adopted by National Bureau of Standards

→ 1960s: IBM setup a research project in computer cryptography

↳ Horst Feistel

→ 1971: development of algorithm

or

LUCIFER

↳ 64 bits block

↳ 128 bits keysize

↳ use in a cash-dispensing

→ IBM effort to develop a marketable commercial encryption product.

↳ implemented on a single chip

DES Encryption:

→ inputs: 2 input

→ plaintext to be encrypted
  → 64 bits long

→ the key.
  → 56 bits long

→ processing of plaintext:

3 phases

→ 64 bit plaintext passes through an initial permutation (IP)

→ 16 rounds of same function
  → permutation
  → substitution

pre output:

left and right halves of the output of 16th round are swapped

→ inverse initial permutation of preoutput. (IP^{-1})

Subkey (k_i) generation:

→ 56-bit key is passed through a permutation function

→ Combination of → left circular shift
→ a permutation

Initial permutation:

→ defined by a table

→ ~~int~~ input to a table consists of 64 bits numbered from 1 to 64

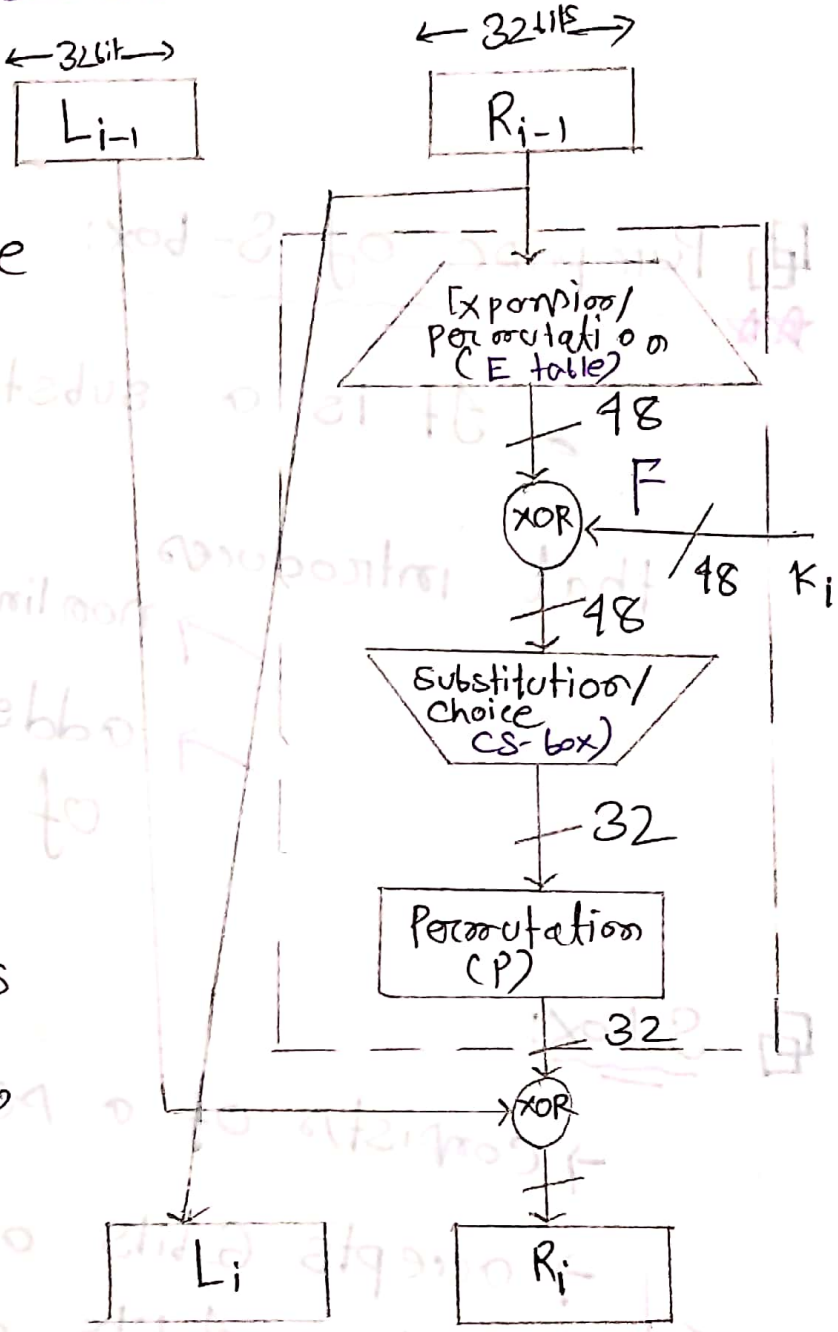
→ 64 entries contain the a permutation of the numbers from 1 to 64.

→ e entry in the table indicates

position of a numbered input bit in the output.

~~Detail~~

Details of single Round:



→ 64-bits inputs are treated as separate 32-bit quantities
- → L(left)
- → R(right)

$L_i = R_{i-1}$

$R_i = L_{i-1} \times F(R_{i-1}, K_i)$

→ round key K_i is 48 bits

→ 32-bits R_i is expanded to 48-bits

→ perform $K_i \oplus R_i$

→ Sub perform substitution by using S-box

→ permutation

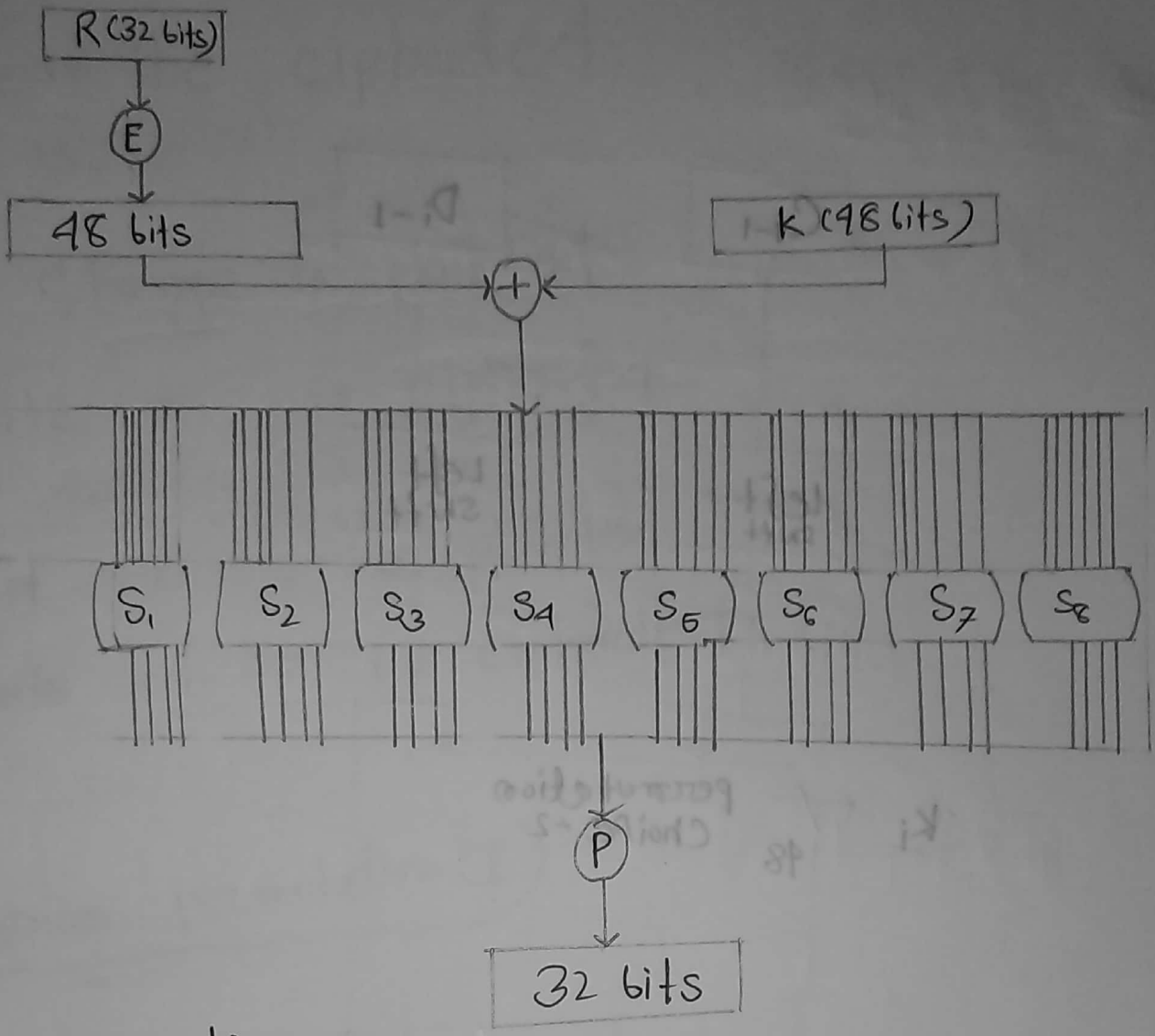
→ XOR between L_{i-1} and 32-bits permutated result.

Purpose of S-box:

- It is a substitution function that introduces
  - → nonlinearity
  - → adds to the complexity of transformation.

S box:

- → consists of a set of eight S boxes
- → accepts 6-bits as input
- → produced 4-bits as output
- → ~~1st and~~
- → 6-bits input $x_0 x_1 x_2 x_3 x_4 x_5$
  - $x_0 x_5$ → one of the sixteen columns
  - $x_1 x_2 x_3 x_4$ → one of the four rows



Key

Key-Generation:

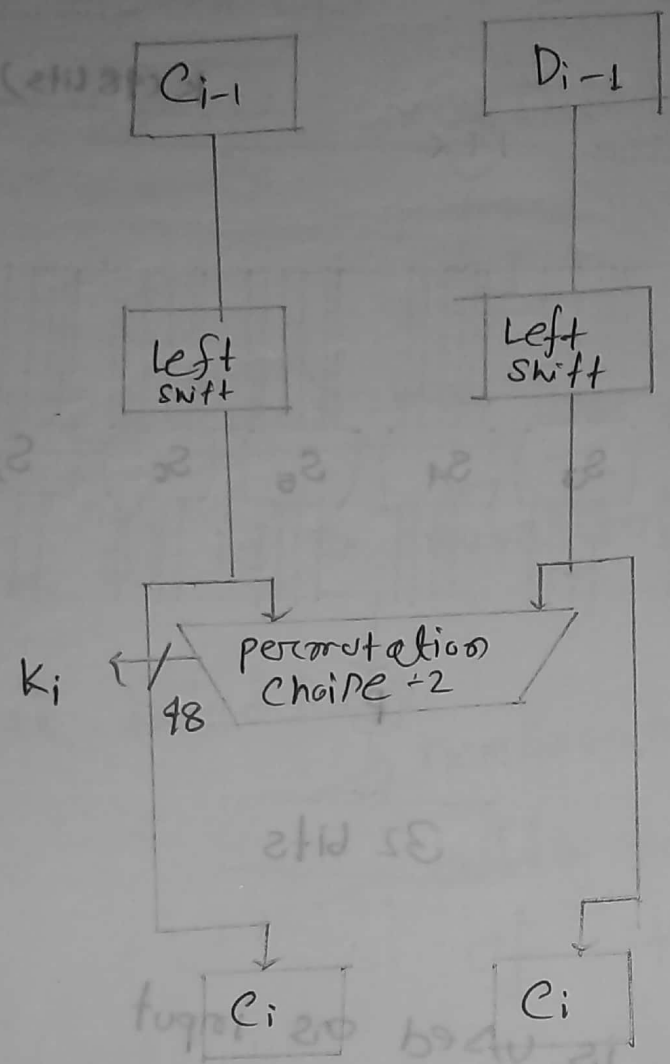
→ 64 bit key is used as input

→ every eighth bit is ignored

→ first subjected to a permutation.

→ resulting 56-bits key
  → two 28-bit quantities
    → C_0
    → D_0

→ C_{i-1} and D_{i-1} subjected to
  → left shift or rotation of 1 or 2 bits



Avalanche effect:

→ it is a property of any encryption algorithm

→ a small change in either the plaintext or the key

→ produces a signifi

change in the ciphertext.

- 1-bit change in plaintext
  34 bits " in ciphertext
- 1-bit " in plaintext
  35-bits " in ciphertext

Expansion permutation (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

2017-1(c)

i) input of S-box(S1) 101101

∴ row no: 11 = 3

column no: 0110 = 6

∴ output of S-box (S1) = 1

$= (0001)_2$

Ans

ii) input of S-box 100110

∴ row no: 10 = 2

column no: 0011 = 3

∴ output of S-box (S1) = 8

$= (1000)_2$

Ans

(iii) input of S-box(S1) 011110

∴ row no: 00 = 0

column no: 1111 = 15

∴ Output of S-box (S1) = 7

$= (0111)_2$ Ans

2016 (2-c)

Input to S-box $(110111)_2$

$\therefore$ row no $= (11)_2 = 3$

Column no $= (1011)_2 = 11$

$\therefore$ output to S-box $= 11$

$= (1011)_2$ Ans

Weakness of DES

→ weak against brute force attack.

As technology existed to build a parallel machine with 1 million encryption devices.

Soln / Alternative of DES

→ AES

→ triple DES

Properties of DES

→ Avalanche effect:

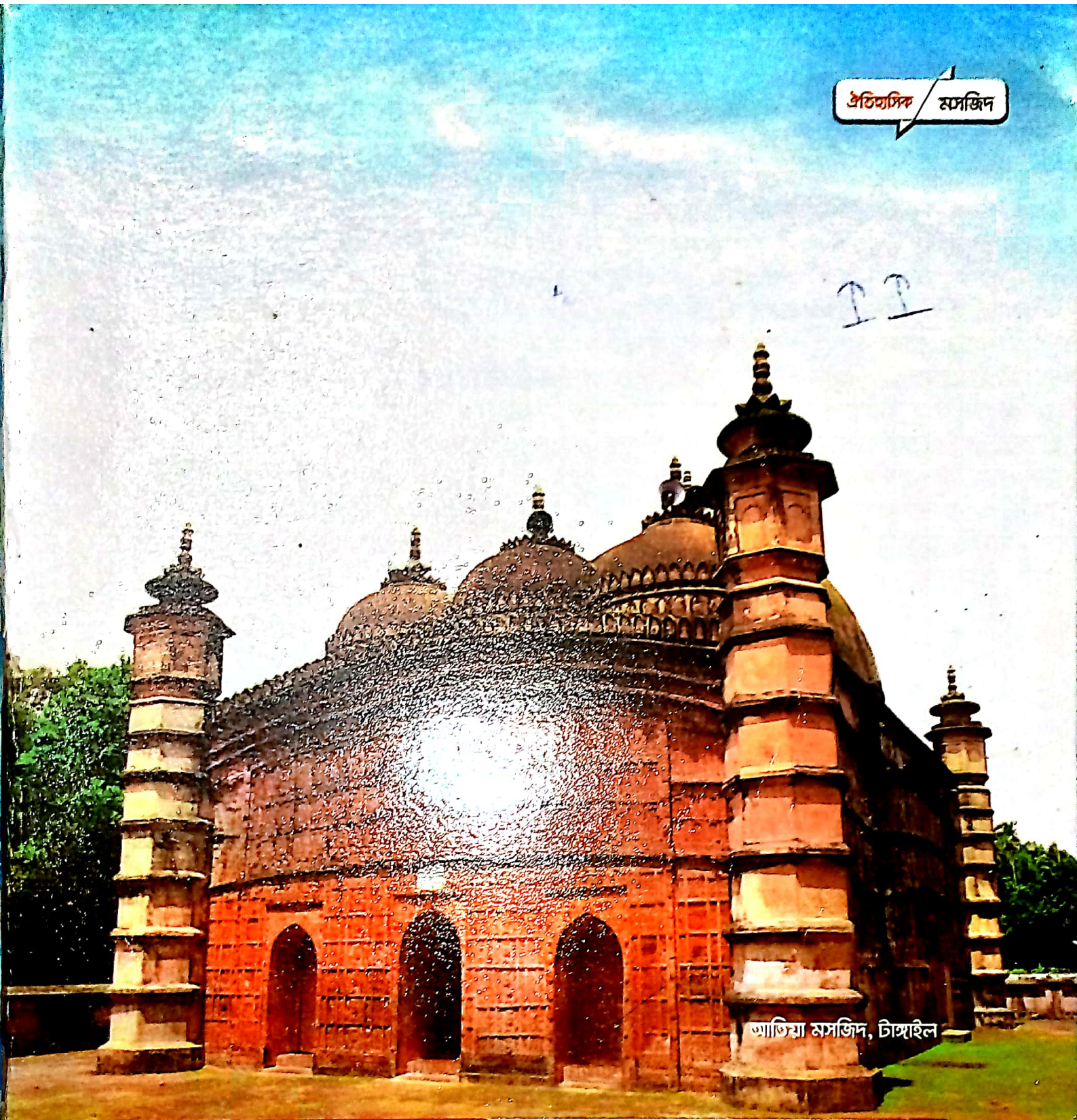
A small change in plaintext/key results in the very large change in the ciphertext

→ Completeness:

Each bits of ciphertext depends on many bits of plaintext.

properties of block cipher

ঐতিহাসিক / মসজিদ



আতিয়া মসজিদ, টাঙ্গাইল

Bashundhara

Exercise Book

Write Your Future

Md. ISMAIL
CSE-4215, Network Security
1503094 ; 4-2; CSE

→ ch-6

→ ch-13

→ ch-17

Chapter 6
More on Symmetric Cipher:

Multiple Encryption:

technique in which an multiple encryption algorithm is used multiple times.

Types of multiple Encryption:

2 types
→ double DES
→ triple DES

Double DES:

→ has two encryption stages

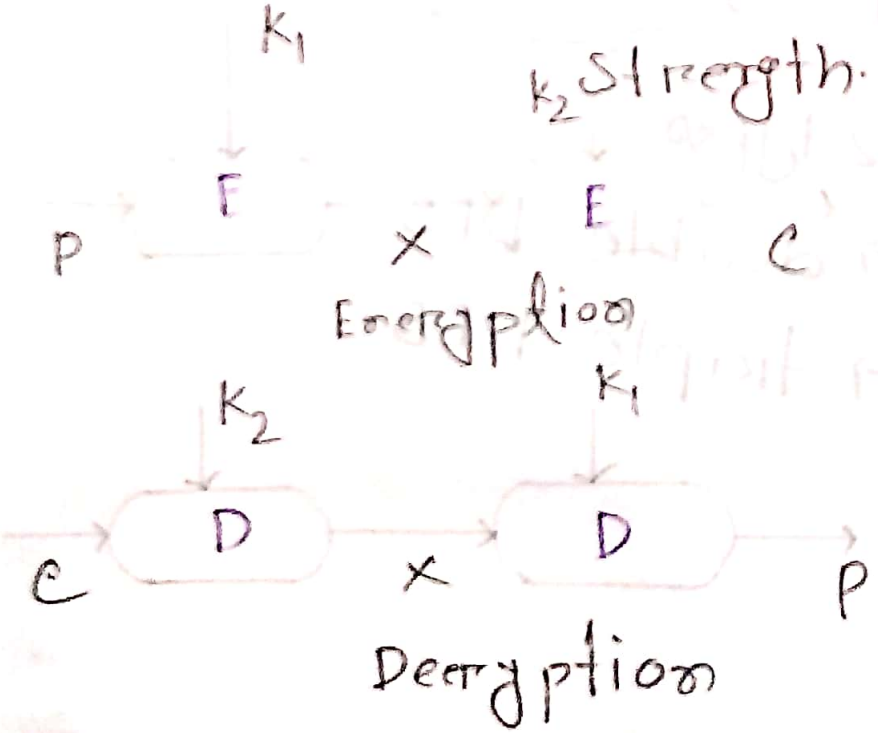
→ has two keys K_1 & K_2

Ciphertext, $C = E(K_2, E(K_1, P))$

Plaintext, $P = D(K_1, D(K_2, C))$

→ key length = $(56 \times 2) = 112$ bits

increase cryptographic K_2 strength.



Meet in the Middle Attack:

For a given known pair (P, C)

→ 1. First

→ encrypt P for all 2^{56} possible values of K_1

$(Y = E(K_1, P))$

→ Store the results in a table & then sort the table by the values of Y.

→ decrypt C for all 2^{56} possible values of K_2

$(X = D(K_2, C))$

→ check the result again the table for a match.

→ If a match occurs, test the two key

↳ again to new known (P, C) pair

→ if the two key produce the correct ciphertext,
accept them as the correct key.

Complexity of MITM:

If the keysize is k, the attack use only 2^{k+1} encryption

∴ Complexity is $O(2^k)$

Soln of meet-in-the-middle attack:

→ Triple DES
  - → with two keys
  - → " three "

Triple DES with two keys:

→ ciphertext is defined by follows

$$C = E(K_1, D(K_2, E(K_1, P)))$$

→ decryption is defined by

$$P = D(K_1, E(K_2, D(K_1, C)))$$

→ currently there are no practical cryptanalytic attack on 3DES

Triple 3DES with three keys:

→ effective key length = 168 bits

$$C = E(K_3, D(K_2, E(K_1, P)))$$

Application:

→ internet-based application adopted 3 key 3DES

Chapter-13
Digital Signature and Authentication Protocols

Problem of message authentication

→ does not protect the two parties
  → against each other

Sol^{n}: Digital signature

Properties of digital signature:

→ verify the author and the date and the time of the signature

→ authenticate the contents
  → at the time of signature

→ verified by third parties
  → to resolve disputes

~~Properties of~~ requirements for a digital signature

→ signature must be a bit pattern
  ↳ depends on the message being signed

→ must use some information unique to the sender
  ↳ to prevent both
    * forgery
    * denial

→ relatively easy to produce the digital signature

→ relatively easy to recognize and verify the digital signature

→ computationally infeasible to forge a digital signature

→ a new message for an existing digital signature

→ by constructing

→ a fraudulent digital signature for a given message

→ practical to retain a copy of digital signature

→ in storage

Approaches of digital signature:

→ direct digital signature

→ arbitrated " "

Digital Signature:

It is an authentication mechanism
→ enables the creator of a message
→ to attach a code that acts as a signature

→ taking the hash of the message
→ encrypting the message
→ with the creators private key

Approaches of digital signature

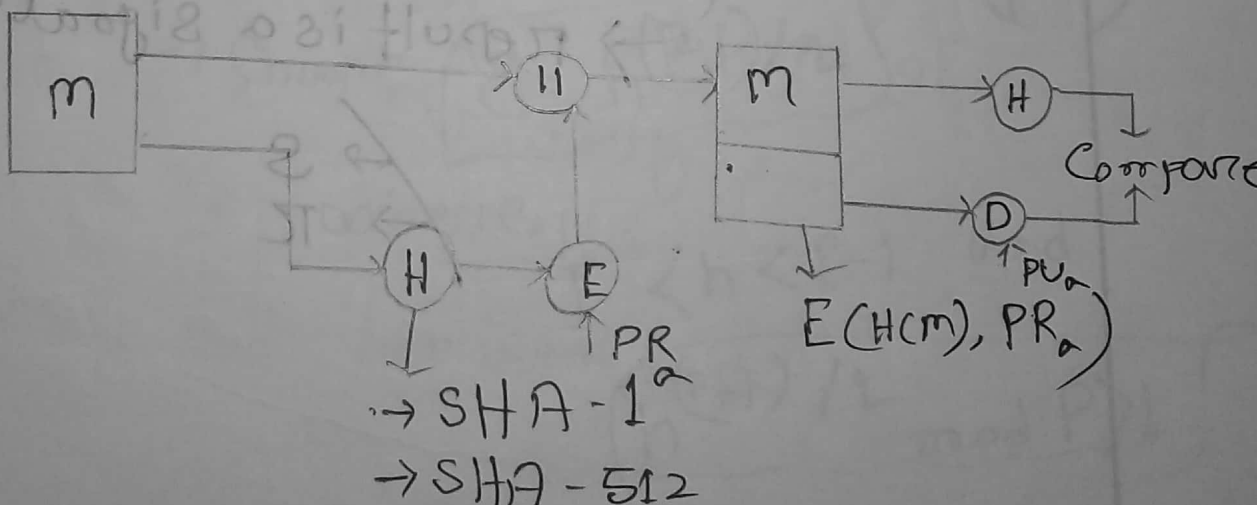
2 approaches

→ RSA approach

→ DSS "

RSA vs DSS approach (Block diagram)

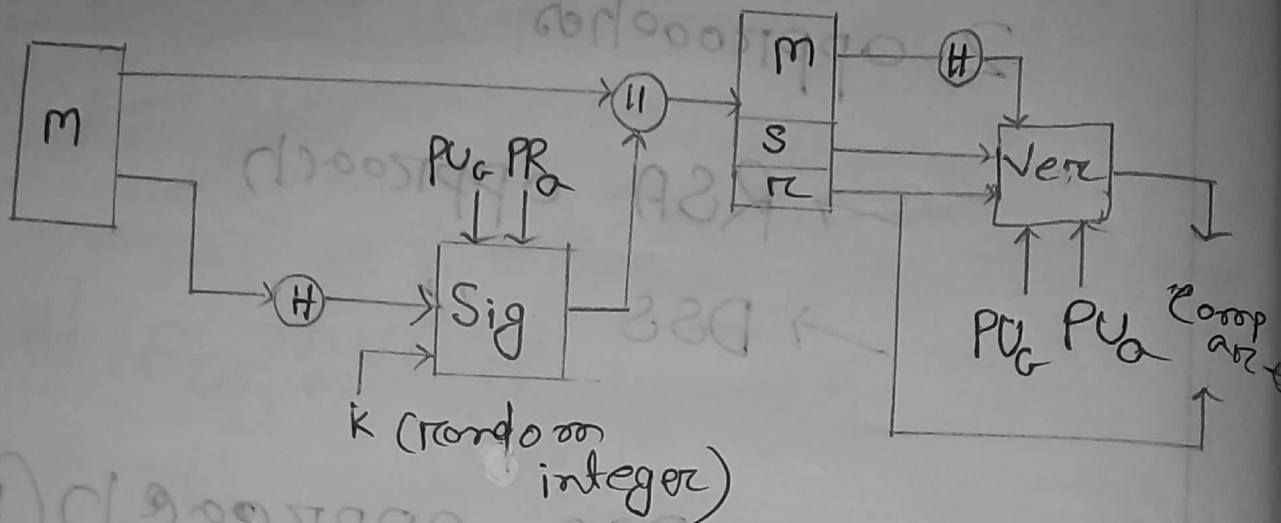
RSA approach:



→ SHA-1

→ SHA-512

DSS approach



→ 2 functions

- Signature function
- Verification "

→ result is a signature

- S
- r

DSA (Digital Signature Algorithm):

- Shared global public key values (p, q, g)
  - → Choose 160-bit prime number q
  - → a large prime p with
    $2^{L-1} < p < 2^L$
    where,
    - ✱ L is 512 to 1024 & is a multiple of 64
    - ✱ such that q is a prime divisor of (p-1)
  - → Choose $g = h^{(p-1)/q}$
    where, $1 < h < p-1$ and
    $h^{(p-1)/q} \bmod p > 1$

User choose private & compute public key:

→ ~~choose random~~

→ private key, $x < q$

↑ randomly chosen

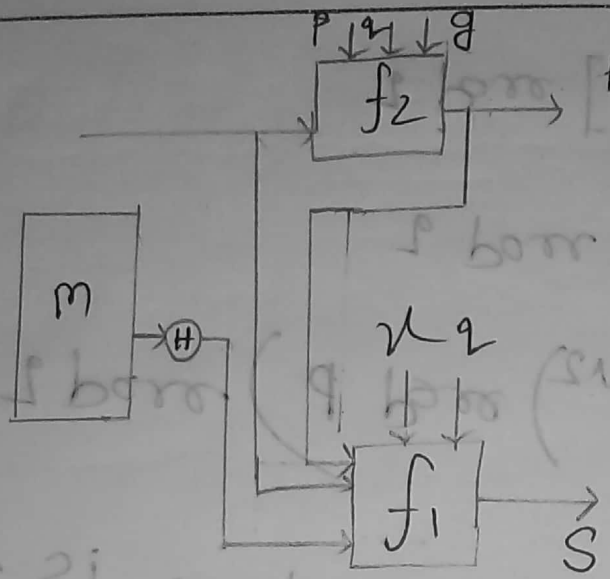
→ public key, $y = g^x \bmod p$

Signing function:

To sign a message m the sender

→ generate a random signature key k; $k < q$

→ k must not be reused



→ Compute signature pair

$$S = f_1(H(m), K, x, r, q) = \left(K^{-1}(H(m) + xr)\right) \bmod q$$

$$r = f_2(K, p, q, g) = (g^k \bmod p) \bmod q$$

→ sends signature (r, s) with message

verifying function:

To verify a signature receiver computes

$$w = f_3(s', q) = (s')^{-1} \bmod q$$

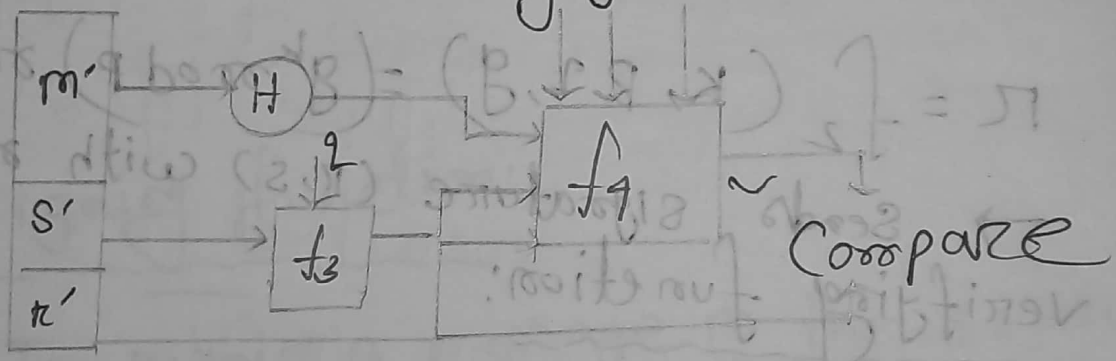
$$v = f_4(w, r', y, q, g, H(m'))$$

$u_1 = [H(m)\, w] \bmod q$

$u_2 = (r\, w) \bmod q$

$v = \left(\left(g^{u_1} y^{u_2} \right) \bmod p \right) \bmod q$

if $v = r$ then signature is verified



Chapter 17
Web Security

Web security consideration

→ the internet is two way

→ The web is increasingly serving as a highly visible outlet (दिखने वाला) for corporate

→ Although web browsers are very easy to use, web servers are relatively easy to configure and manage, and web content is increasingly easy to develop, the underlying software is extraordinarily complex.

→ A web server can be exploited as a launching pad into the corporations or agencys entire computer complex

→ Casual and untrained users are common clients for web-based services.

Types of security attack.

2 types
- active attack
  - altering message in transit
    - between client & server
  - altering information on a web site
  - impersonating another user
- passive attack
  - eavesdropping on network traffic between browser & server
  - gaining access to information on a web site.

in terms of location
3 types
- web server
- web browser
- network traffic between browser & server

→ computer system security
→ web system security

Web security threat:

#Integrity:

→ Modification of user data

→ [illegible] of memory

→ [illegible] of message traffic in transit

→ Trojan horse browser

Consequence:

→ Loss of information

→ vulnerability to all other threat

→ Compromise of machine

Countermeasure:

→ cryptographic checksum

Confidentiality:

threat

→ Eavesdropping on the Net
→ Theft of info. from server
→ " of data " client
→ Info about network configuration
→ " " which client talk to server

consequence:

→ Loss of information
→ Loss of privacy

Counter measure:

→ Encryption.
→ web proxies

Denial of Service:

Threat:

→ killing of user thread

→ Filling up disk of memory

→ Flooding machine with
    → bogus requests

→ Isolating machine by DNS attack

Consequence:

→ Prevent user from getting work done

→ Disruptive

→ Annoying

Counter measure:

* → difficult to prevent

Authentication:

Threats:

→ impersonation of legitimate user
(ছদ্মবেশ)

→ Data forgery
(তথ্য জালিয়াতি)

Consequence:

→ Belief that false information is valid

→ misrepresentation of user

Counter measure

→ cryptographic technique

Location of security facilities in the TCP/IP Protocol Stack:

or web traffic security approaches

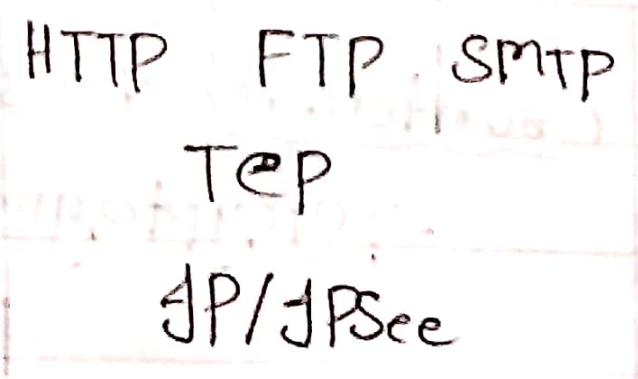
→ Network level

→ transport "

→ Application "

Network level:

→ to provide web security is to use IPSec



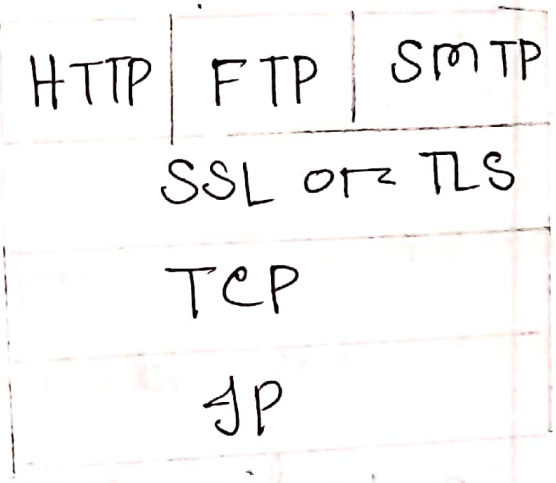
Ad. of IPSec:

→ it is transparent to end user

→ provides a general-purpose solution.

Transport level:

→ implement security just above TCP



Ex:

→ Secure Socket layers (SSL)

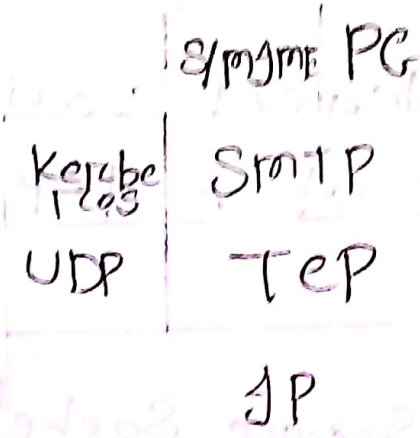
→ transport Layer Security (TLS)

2 implementation Choice

→ SSL or TLS could be provided as part of underlying protocol

→ SSL can be embedded in specific packages

Application level:



Application specific security services are embedded within the particular application

Ad:

service can be tailored to the specific needs of a given application.

Ex:

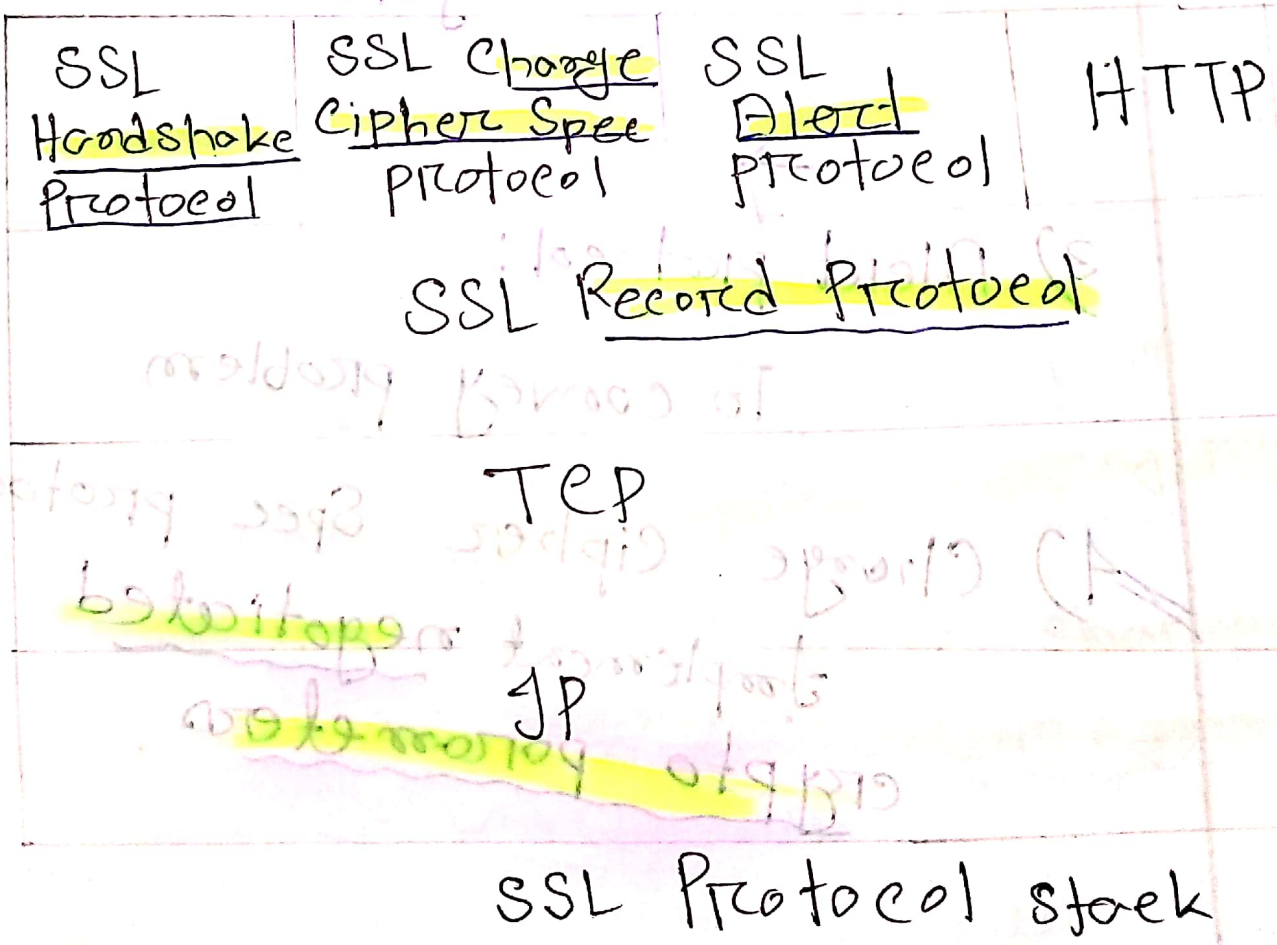
Secure Electronic Transaction (SET)

SSL Architecture:

→ designed to make use of TCP

→ to provide a reliable end-to-end secure service

→ it is two layers of protocol



SSL Protocol stack

4 components in two layers:

1) Record protocol:

Provides encryption & MAC

2) Handshake protocol:

→ Negotiates crypto parameter for a "SSL session"

→

3) Alert protocol:

To convey problem

4) Change cipher Spec protocol:

Implement negotiated crypto parameters

SSL concepts:

2 important SSL concepts:

→ SSL session

→ SSL connection

SSL session:

→ it is an association between a client and a server.

→ created by

→ Handshake Protocol

→ define a set of cryptographic security parameters

→ can be shared among multiple connections

SSL connection:

→ connections are peer to peer relationship

→ connections are transient

→ every connection is associated with one session.

Session state:

A session state is defined by the following parameters:

→ Session identifier

An arbitary byte sequence chosen by the server.

→ Peer certificate

An x500.v3 certificate of the peer.

→ Compression method:

An algorithm used to compress data → prior to encryption

→ Cipher spec:

Specifies the

→ bulk data encryption algorith

→ a hash algorithm

↳ MD5
↳ SHA-1

for MAC calculation.

Also defines cryptographic attribu

↳ hash size

→ master secret:

48-byte secret → shared between the client & server

→ Is resumable

A flag indicating whether the session can be used to initiate new connection

Connection state: (7)

Defined by the following parameter

→ Server and client random:

Byte sequence that are chosen by the server & clien for each connecti

→ Server write MAC secret:

The secret key used in MAC operation → on data sent by the server

→ Client write MAC secret:

The secret key used in MAC operation → on data sent by the client

→ Server write key:

The conventional encryption key for → data encrypted by ser
→ decrypted by cli

→ Client write key:

The conventional encryption key for

→ data encrypted by the Client

→ decrypted by the Server

→ Sequence numbers:

Each party maintain separate sequence number for → transmitted & receiving message for each connection.

☆ may not exceed $2^{64} - 1$

→ Initialization vector:

When block cipher in CBC mode is used an initialization vector (IV) is maintained for each key.

SSL Record Protocol:

Provides two services for SSL:

- **Confidentiality:**

  A shared secret key that is used for conventional encryption of SSL payloads

message Integrity:

A shared secret key that is used to form a MAC.

Operation of SSL record protocol:

→ Application data
→ Fragmentation
→ Compression
→ Add MAC
→ Encrypt
→ Append SSL record header

Fragmentation:

Upper layer message is fragmented into
→ blocks of 2^{14} bytes or less

Compression:

→ optionally applied
→ must be lossless

→ may not increase the content length

→ by more than 1024 bytes

Message Authentication Code (MAC)

The calculation is defined by

hash(MAC_write_secret ||
pad_2 || hash(MAC_write_secret
|| pad_1 || seq_num ||
SSLCompressed.type ||
SSLCompressed.length ||
SSLCompressed.fragment))

where

MAC_write_secret = shared secret key

hash = cryptographic hash algorithm
- SHA-1
- MD5

Pad_1 = the byte 0x36 (00110110)
- repeated 48 times for MD5
- " 40 " " SHA-1

pad_2 = the byte 0x5c (01011100)
- repeated 48 times for MD5
- " 40 " " SHA-1

SEQ_NUM = the sequence number for this message.

Encryption:

→ use symmetric encryption

→ may not increase the content length by more than 1024 bytes

Block Cipher		Stream Cipher	
Algorithm	Key size	Algorithm	key size
AES	128, 256	RC4-40	40
IDEA	128	RC4-128	128
RC2-40	40		
DES-40	40		
DES	56		
3DES	168		
Forte^{22}	80		

→ used in a smart card encryption scheme.

→ ~~HMAC is computed before~~
→ length: 40 bits

Append SSL record header:

Header consisting of following

→ Content Type (8 bits):

The higher layer protocol used to process the enclosed fragment

→ Change Cipher Spec
→ Alert
→ handshake
→ application-data

→ Major version (8 bits):
→ indicates major version of SSL
→ For SSLV3 value is 3

→ Minor version (8 bits):
→ indicates minor version of SSL
→ For SSLV3 value is 0

Compressed length (16 bits):

→ The length in bytes of the plaintext fragment.

Change_Cipher Spec Protocol:

→ consists of a single message

consists of a single byte with the value 1.

1 byte

1

PURPOSE:

to copy the pending state to be copied into the current state.

Alert Protocol:

→ used to convey SSL-related alerts → to the peer entity

1 byte 1 byte

Level Alert

Alert protocol

2 bytes

→ first byte takes the value warning(1) or fatal(2)

→ second byte contains a code that indicates the specific alert.

*Alerts that are always fatal: (5)

→ unexpected_message:
An inappropriate message was received

→ bad_record_mac:
An incorrect MAC was received

→ decompression_failure:
The decompression function received improper input.

→ handshake_failure:
Sender was unable to negotiate → an acceptable set of security parameters given

the option available

→ illegal parameter:

A field in a handshake message was out of range

→ inconsistent with other fiel

* The remainder of the alerts are following: (7)

→ close_notify:

→ Notifies the recipient that the sender will not send any more messages on this connection.

→ Each party is required to send close_notify alert

→ no_certificate:

Sent in response to a cert certificate request if no appropriate certificate is available

→ bad_certificate:

A received certificate was corrupt

→ unsupported certificate:

The type of received certificate is not supported

→ certificate_revoked:

A certificate has been revoked by its signer.

→ certificated_expired:

A certificate has expired

→ certificate_unknown:

Some other unspecified issue arose

↳ in processing the certificate

→ rendering in

Handshake protocol:

→ most complex part of SSL.

* Allows server & client to
  → authenticate each other
  → to negotiate on
    encryption & MAC
  → to " cryptographic keys
    → to be used

→ used before any application data is transmitted

* Handshake protocol

* It consists of a series of messages → exchanged by client & server

* Each message has three fields

1 byte	3 byte	≥ 0 byte
Type	Length	Content

→ Type (1 byte):
Indicates one of 10 messages

→ Length (3 bytes):
Length of the message in byte

→ Content (>=0 bytes):

The parameter associated with this message.

*Handshake protocol message type:

10 types

→ hello_request → null

→ client_hello
→ Server_hello
→ certificate → chain of x509v3 certificate
→ server_key_exchange → parameter signature
→ certificate_request → type, authorities
→ server_done → null
→ certificate_verify → signature
→ client_key_exchange → parameter signature
→ finished → hash value

Handshake Protocol Action:

4 phases

→ Establish Security Capabilities

→ Server Authentication and key Exchange

→ Client " and key Exchange

→ Finish

Phase 1:

Establish Security Capabilities:

→ initiated by the client

→ sends Client_hello_msg

Client_hello_message consists of

→ Version:

The highest SSL version

→ Random:

A client-generated random

Structure

→ 32-bit timestamp

→ 28 bytes generated by a secure number generator

→ Session ID:

** A variable length session identifier.

→ nonzero value:

* client wishes to update the parameter of existing session

* create a new connection on this session

→ Zero value wishes to establish a re connection on a new ses

→ CipherSuite:
List that contains the combination of cryptographic algorithm.
Defines → Key exchange algorit
→ CipherSpec

→ Compression method:
A list of compression methods

→ client wait for server hello message

Version:
→ lower of the version supported by client
→ higher of " " supported by server

random:
→ generated by server

Session ID:
→ same value is used if client session ID is non zero
→ new value if session ID is zero

Compression:
→ compression method selected by server
→ from those proposed by client

CipherSuite:

Single cipher suite selected by the server → from those proposed by client.

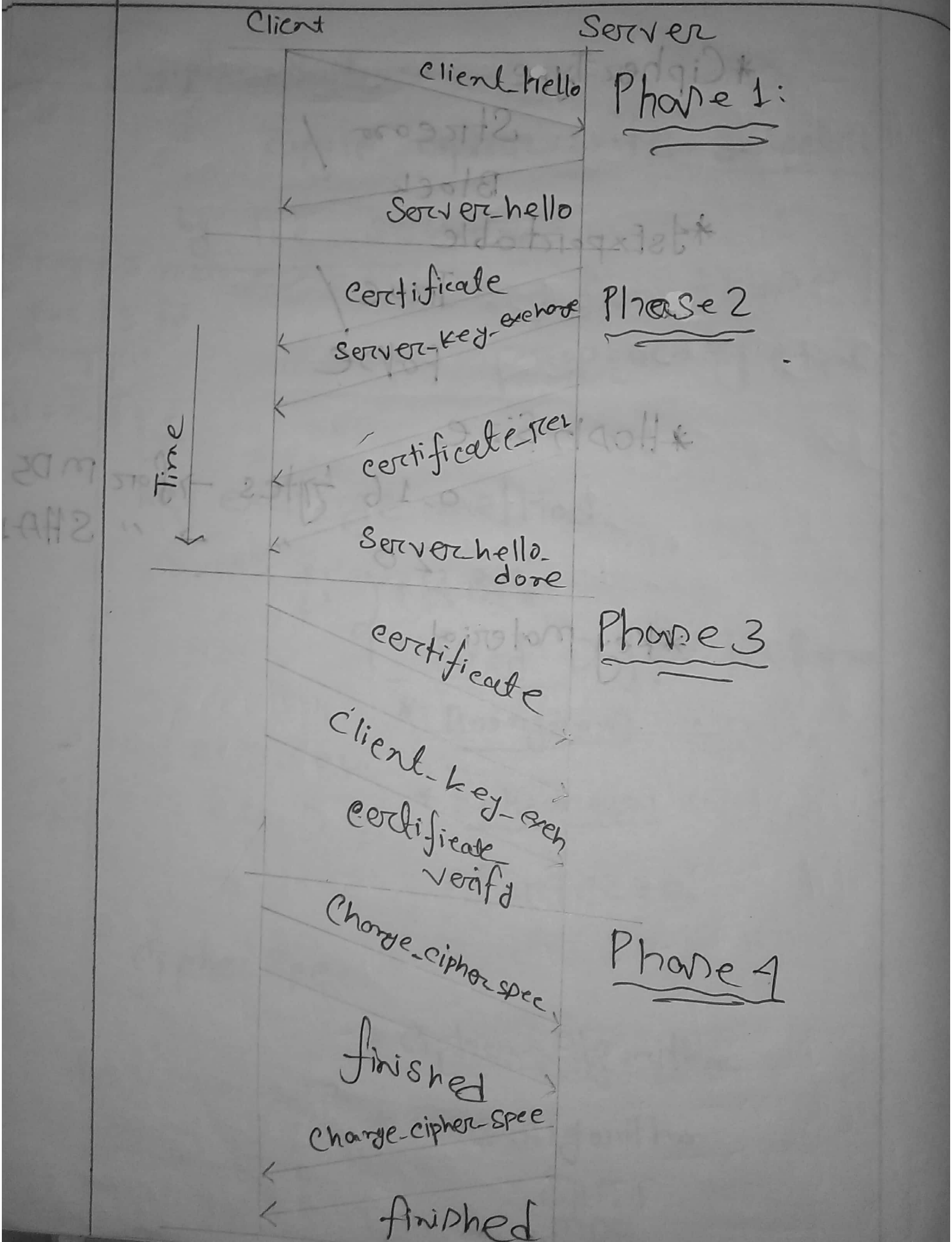
key exchange method:

* RSA
* Fixed Diffie-Hellman
* Anonymous " "
* Ephemeral " "
* Fortezza

CipherSpec:

* Cipher Algorithm
* MAC Algorithm
  SHA-1
  MD5

* Cipher Type
  - Stream /
  - Block
* IsExportable
  - True /
  - False
* Hash Size
  - 0.16 bytes → for MD5
  - 20 " → " SHA-1
* Key Material


Client
Server
client_hello
Phase 1:
Server_hello
certificate
server_key_exchange
Phase 2
certificate_req
Time
Server_hello_done
Phase 3
certificate
client_key_exch
certificate verifd
Change_cipher_spec
Phase 4
finished
change_cipher_spec
finished

Phase 2:

Server Authentication and key Exchange

→ Certificate message:

→ contains one or a chain of X.509 certificate

→ required for any agreed-on key exchange method

→ except anonymous Diffie-Hellman

→ Server_key_exchange message:

→ not required in two instances

if Server has sent a certificate with

* fixed Diffie-Hellman
* RSA

→ required for
* Anonymous Diffie-Hellm
* Ephemeral " "
* RSA,
→ server is using RSA
→ but has a signatur
only
* Fortezza

→ certificate_request_message:
→ a nonanonymous server
can request a certificate
from the client.
→ it includes two parameter
→ certificate_t
→ certificate_
author
ties

→ indicates the public_key algorithm and its use

certificate_type:

* RSA, Signature only
* DSS, signature only
* RSA for fixed Diffie-Hellman
* DSS for fixed Diffie-Hellman
* RSA for ephemeral Diffie
* DSS for "
* Fortezza

→ server_done_message

* sent by the server to indicate the end of the server hello

NB Servers wait for client response

Phase 3:

Client Authentication and key Exchange

* Client should verify that the server provided a valid certificate
* Check that the server_hello-parameters are acceptable

certificate message:

if no suitable certificate is available, the client sends

no_certificate

client_key_exchange_message:

The content of message depends on the type of key exchange.

RSA
generate a 48-byte pre-master secret.

Ephemeral or Anonymous Diffie-Hellman:
Client's public Diffie-Hellman parameters are sent.

Fixed Diffie-Hellman:
null.

Fortezza:
client's Fortezza parameters are sent.

→ client-verify-message:
→ to provide explicit verification of a client certificate.

→ message signs a hash code ~~based on the~~

certificateVerify: signature.md5_hash
MD5(master_secret || Pad_2 ||
MD5(handshake_message ||
master_secret || Pad_1));
certificate: signature.Sha_hash
SHA(master_secret || Pad_2||
SHA(handshake_message ||
master_secret || Pad_1));

Phase 4:

Finish:

→ Client sends a change_cipher_spec message

→ copies the pending CipherSpec into the current CipherSpec

→ then sends the finished message under the new algorithm

finished message:

* MD5(master_secret || Pad2 || MD5(handshake_messages || sender || master_secret || Pad1))
* SHA(master_secret || Pad2 || SHA(handshake_messages || sender || master_secret || Pad1))

Transport Layer Security

goal: to produce an Internet standard version of SSL

→ defined as a Proposed Internet Standard → in RFC 5246

→ RFC 5246 is very similar to SSLv3.

SSL vs TLS

→ Version Number:

For SSLv3 major version is 3
minor " is 0

For current version of TLS major version 3
minor version 1

→ Message Authentication Code (MAC)

* actual algorithm

→ In SSLv3 Padding bytes are connected with secret key

→ In RFC " " connected with XORed with the secret key & padded to the block length

Scope of MAC:

MAC calculation covered all the field covered by SSLv3 calculation plus the field TLSC compressed.

→ Alert Codes:

→ TLS support all alert codes defined by in SSLv3

→ exception of no certificate

→ A number of additional codes are defined in TLS

→ Cipher Suites:

Key exchange:

TLS supports all of the key exchange techniques of SSLv3

with the exception of Fortezza

Symmetric Encryption Algorithm:

TLS includes all of the symmetric encryption algorithm found in SSLv3

with the exception of Fortezza

→ Client Certificate Types:

→ For TLS: rsa_sign,
dss_sign,
rsa_fixed_dh,
dss_fixed_dh

→ For SSLv3,
rsa_ephemeral_dh,
dss_ephemeral_dh,
fortezza_kea
+ certificate type used in TLS

→ certificate_verify & ~~Finished_message~~

→ For MD5 & SHA-1 hashes are calculated only over handshake_messages

→ For SSLv3

MD5, SHA-1 hashes are calculated over handshake-messages master_secret pads

→ Finished Message:

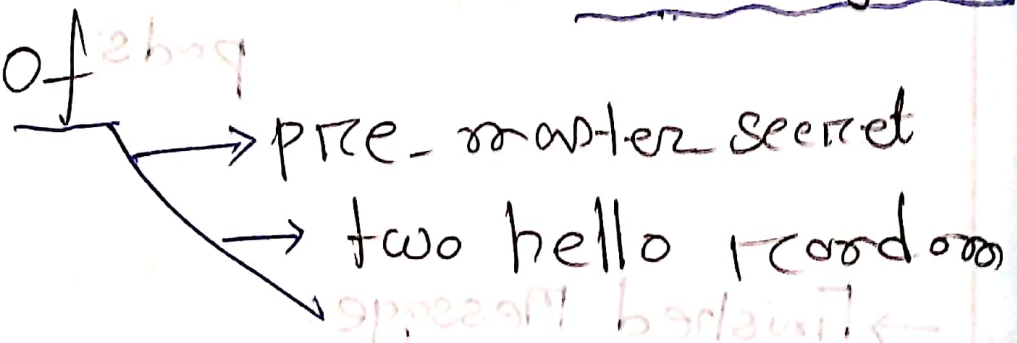
→ for TLS the finished message is a hash based on the

* shared_master_secret
* previous handshake messages
* label

PRF (master_secret, finished_label, MD5 (handshake_message) || SHA-1 (handshake_message))

→ Cryptographic Computation:

master_secret in TLS is calculated as a hash function of



master_secret = PRF (pre_master secret, "master secret", ClientHello.random || ServerHello.random)

→ Padding

* In SSL, minimum amount of data required so that total size of the data to be multiple of the cipher's block length

* In TLS any amount that results in a total that is a multiple of the cipher block length → up to maximum of 255 bytes