RSA Algorithm

↓

Rivest – Shamir – Adleman

– Asymmetric key Algorithm and Block Cipher Algo

③ steps

1. Key Generation
2. Encryption
3. Decryption

* Key Generation

1. Select 2 large prime numbers p and q

↓

for more security

$p = 3$ and $q = 11$

② Calculate $n = p * q \Rightarrow n = 33$

③ Calculate $\phi(n) = (p-1)(q-1)$

$\phi(n) = 2 \times 10 = 20$

④ Choose the value of e such that

$1 < e < \phi(n)$ and $\gcd(\phi(n), e) = 1$

Let $e = 7 \Rightarrow 1 < 7 < 20$ and $\gcd(20, 7) = 1$

$\boxed{e = 7}$

CPPL
Date: / / 2022
Page No.

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

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

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

$7 \times d \bmod 20 = 1$

$\boxed{d = 3}$

(6) Public key $= \{e, n\} = \{7, 33\}$

(7) Private key $= \{d, n\} = \{3, 33\}$

(*) Encryption

$c = m^e \bmod n$

↓ ciphertext

$m \rightarrow$ No. of digits in PT

Assume m

$\boxed{m < n}$

Let $\boxed{m = 31}$

$c = (31)^7 \bmod 33$

$= 4$

$\therefore \boxed{c = 4}$

(*) Decryption

$m = c^d \bmod n$

$= (4)^3 \bmod 33$

$= 64 \bmod 33$

$= 31$

$\boxed{m = 31}$

Diffie Hellman Key Exchange Algorithm

- Not an Encryption / Decryption Algorithm
- Used to exchange keys b/w sender and receiver
- Asymmetric key Cryptography

* Procedure

① Consider a prime number q
Let q = 7

② Select α such that $\boxed{\alpha < q}$ and α is primitive root of q

$\alpha = 3$ and $q = 7$
$3^1 \bmod 7 = 3$
$3^2 \bmod 7 = 2$
$3^3 \bmod 7 = 6$
$3^4 \bmod 7 = 4$
$3^5 \bmod 7 = 5$
$3^6 \bmod 7 = 1$

$= (1, 2, 3, 4, 5, 6)$
$\therefore$ 3 is primitive root of q
Also 5 is primitive root of q.

③ Assume X_A (Private key of A) and $\boxed{X_A < q}$
Calculate $Y_A = \alpha^{**A} \bmod q$
$q = 7$ and $\alpha = 35$
Let $X_A = 3$
$Y_A = (5)^3 \bmod 7 = 125 \bmod 7 = 6$
$\boxed{Y_A = 6}$

classfollow

④ Assume X_B and $X_B < q$
Calculate $Y_B = \alpha^{X_B} \bmod q$

$X_B = 4$

$Y_B = (5)^4 \bmod 7 = 625 \bmod 7 = 2$

$Y_B = 2$

$X_A, X_B = (3, 4)$
$Y_A, Y_B = (6, 2)$

⑤ Calculate secret keys K_A and K_B
$K_A \Rightarrow$ Person A $\quad K_B \Rightarrow$ Person B

$K_A = (Y_B)^{X_A} \bmod q \quad K_B = (Y_A)^{X_B} \bmod q$

$K_A = 2^3 \bmod 7 = 1$
$K_B = 6^4 \bmod 7 = 1$

$K_1 = K_2$
Key exchanged successfully.

MD5 → Message Digest
- developed by Rivest
- fast and produces 128 bit message digest

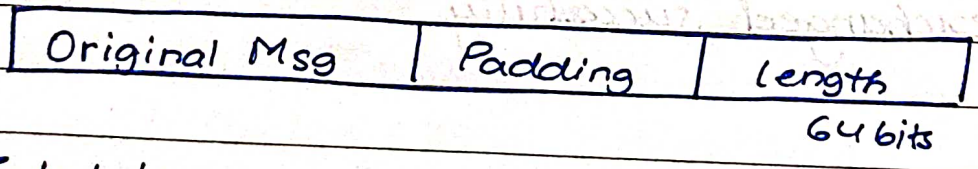
* Working

(1) Padding :-
Original Msg + Padding → extra bits
(So that total length is 64 bit less than exact multiple of 512)

Eg Original msg = 1000 bits
512X2 = 1024
512X3 = 1536 − 64 = 1472 → Total length

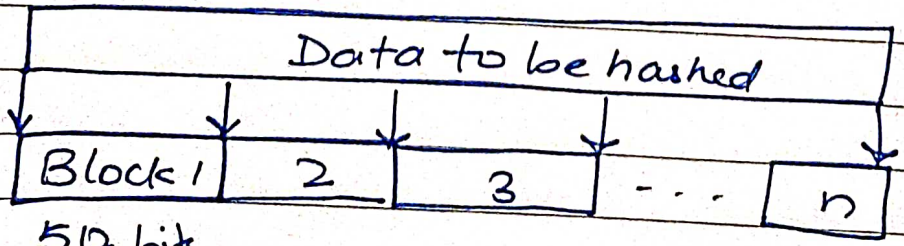
Padding = 472 bits

(2) Appending
Append the original length before padding
~~Calculate length mod 64~~



Total length now becomes a multiple of 512

(3) Divide the input in 512 -bit blocks

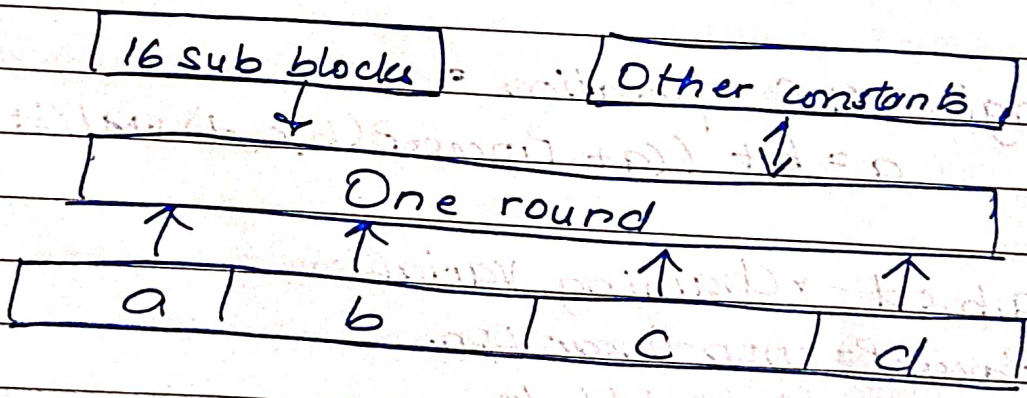


④ Initialise chaining variables.
Four variables → A, B, C, D → each of 32 bits

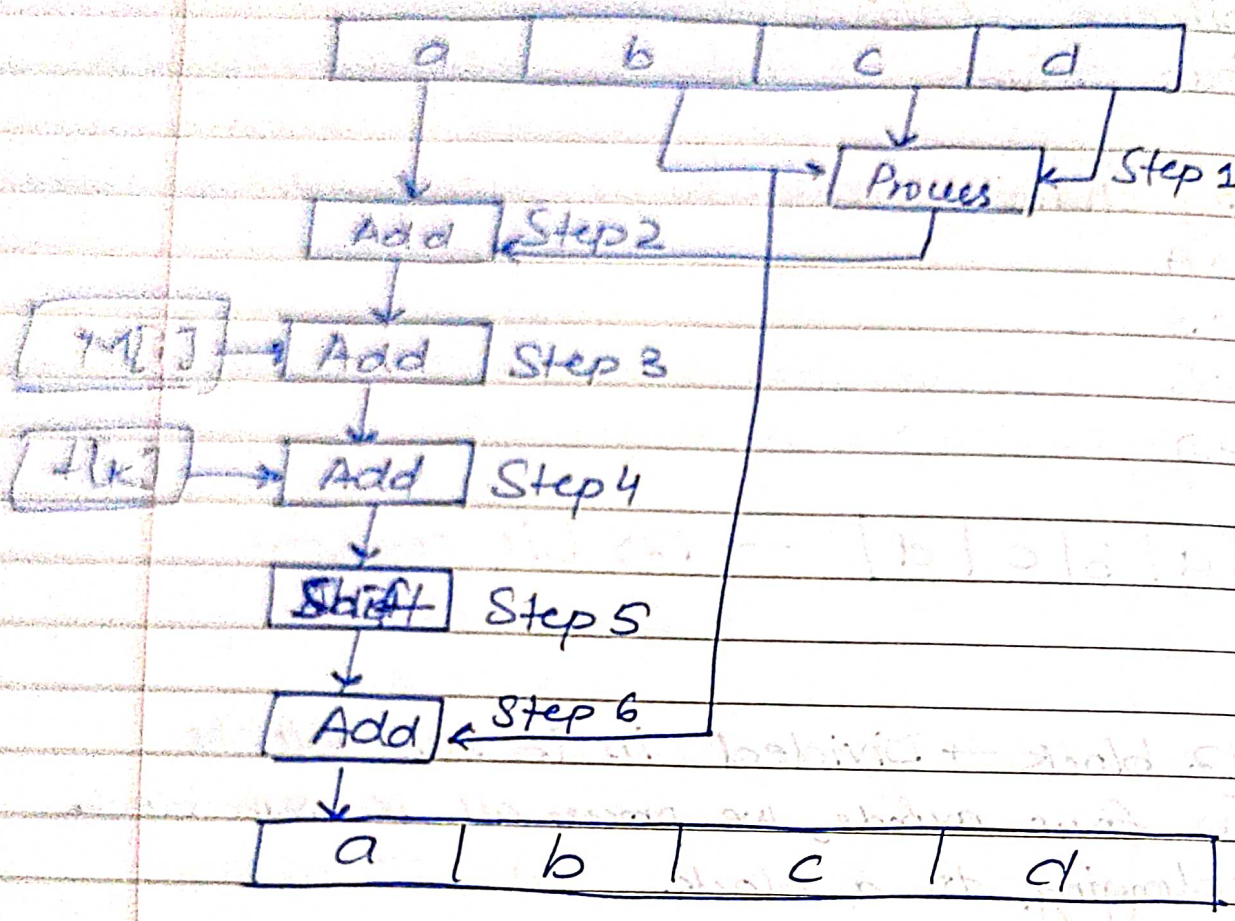
⑤ Copy chaining variables in corresponding variables
a = A
b = B
c = C
d = D

| a | b | c | d | → 128 bit register

512 block → Divided in 16 sub-blocks
In four rounds, we process all 16 sub-blocks belonging to a block.



64 elements are there in constants each of 32 bits.
16 elements are used in each round.



Single MD5 operation :

$a = b + ((a + \text{Process P}(b,c,d) + M[i] + t[k]) <<< s)$

a, b, c, d → chaining variables
Process P → non-linear opn.
M[i] → 16 subblocks
t[k] → Constants
<<< s → Circular shift

Date : / / 2022

Page No.

classfellow

* SHA Algorithm
↓
Secure Hash Algorithm

- Modified version of MD5

In MD5 – length of O/P – 128 bits
In SHA – length of O/P – 160 bits

* Working :-

① Padding → Adding Padding length - 64 bit < 512 multiple

② Appending → Add length to make it a multiple of 64

③ Divide I/P in 512 blocks

④ Initialise 5 chaining variables
A, B, C, D, E

⑤ Process blocks
a=A b=B c=C d=D e=E

- Divide each block in 16 sub blocks of 32 bits each
- Four rounds (each round = 20 steps)

=> Used for authentication.

SHA-512

- Input length = 2^{128} bits
- Output length = 512 bits

1. Padding — 128 bits less than multiple of 1024
2. Append length — to make it a multiple of 1024
3. Divide blocks into 1024-bit blocks
4. Initialise 8 chaining variable
5. Copy variables

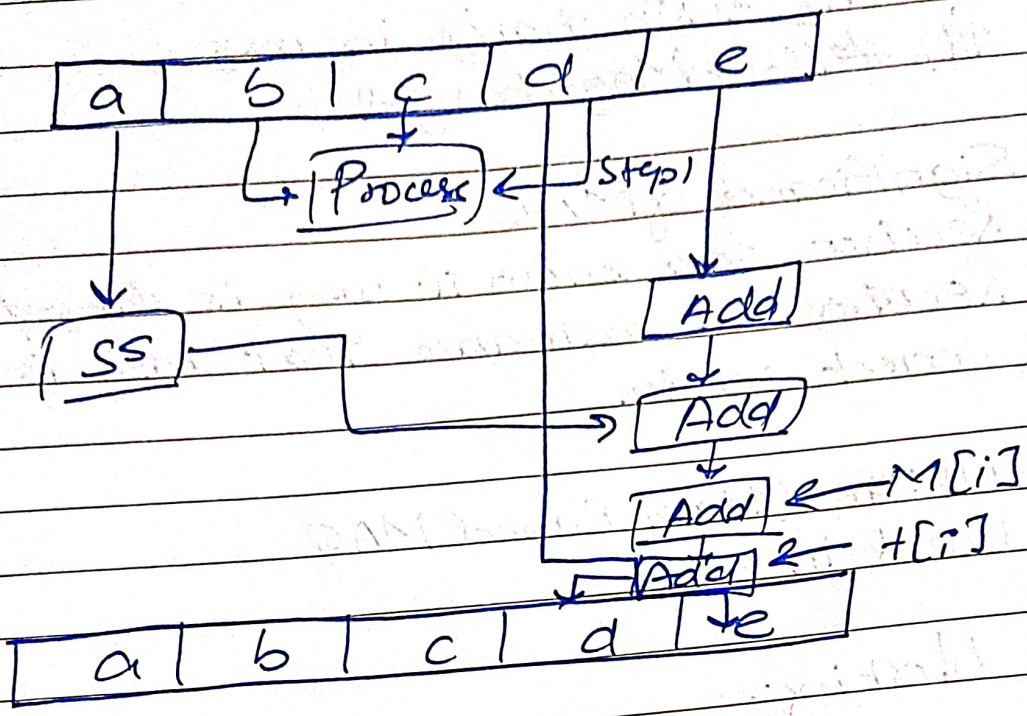
a = A c = C e = E g = G
b = B d = D f = F h = H

Divide 1024 blocks into 16 sub-blocks of 64 bit each

Takes 80 round with 1024 bit block, abcdefgh register and K[t] constants

classfellow

SHA 1 - Process



* ~~HMAC~~ → Msg Authentication Code
- similar to message direct
- symmetric key cryptographic

* Working of MAC
If sender wants to send a msg M

M
↓ key
H1 (MAC code) → cipher text

Now M + H1 → sent to Receiver

↓

Calculate his own MAC with his key (H_2)

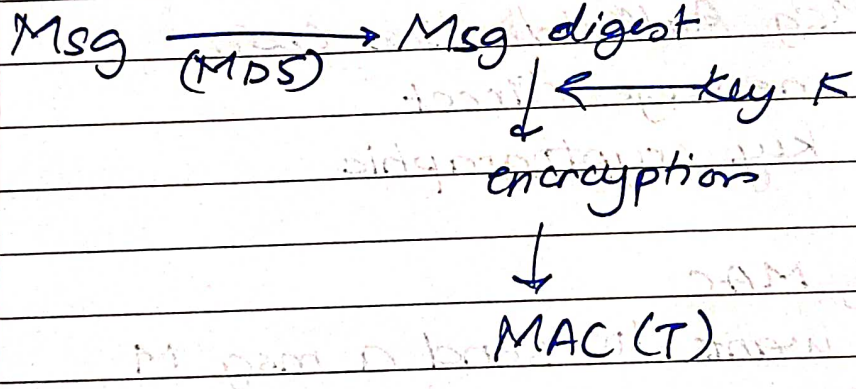
CPPL
Date: / / 2022
Page No.

H_1 and H_2 are compared
$H_1 = H_2$ ⇒ No change in msg
$H_1 \neq H_2$ → Msg changed

* Significance of MAC
— Receiver can know if msg is changed
— Receiver has assurance that msg is from correct sender

* HMAC (Hash Based MAC
— Used in SSL

* Working:



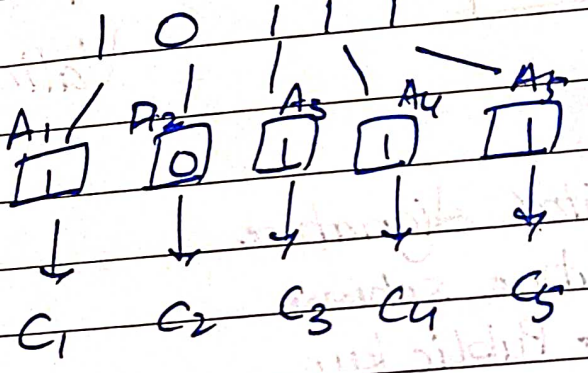
In MAC → direct MAC is generated
In HMAC → MAC is generated with help of msg direct

Date: / / 2022
Page No.

classfellow

* CMAC (Cipher Based MAC)
- has msg size limit
- based on block cipher
- msg is divided into equal no. of blocks and block is encrypted seperately.

Example

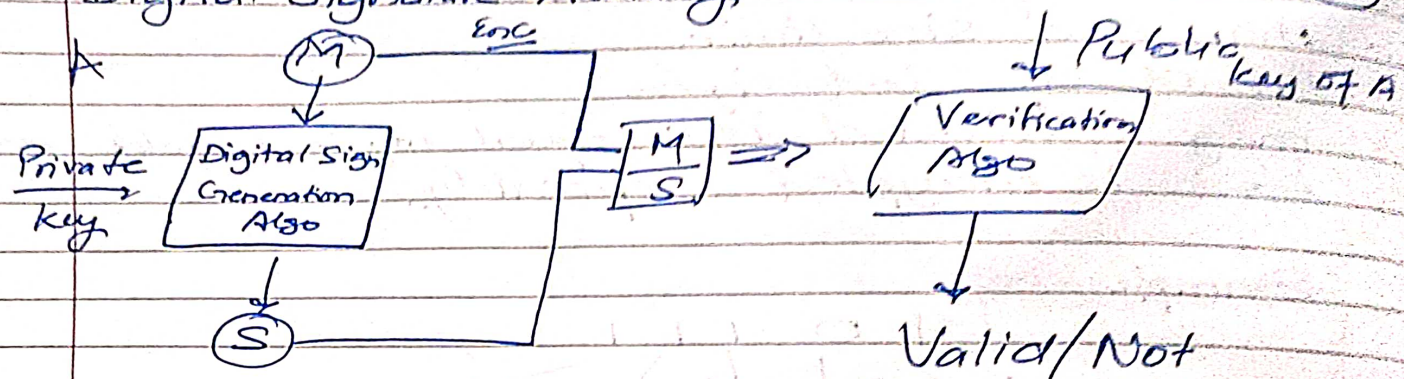


$C_1 = E(A_1, K_1)$
$C_2 = E(K, A_2 \oplus C_1)$
$C_3 = E(K, A_3 \oplus C_2)$
!
!
$C_n = E(K, (A_n \oplus C_{n-1}))$
↓
acts as MAC

classfollow

Date: / / 2022
Page No.

* Digital Signature Working,



— Elgamal Digital Signature
— Digital signature scheme
Encryption → Public key
Decryption → Private key

Working

① Select a prime number (q)

② Select a primitive root (α) of q

③ Generate a random number (X_A)
$1 < X_A < q-1$

④ Compute $Y_A = (\alpha)^{X_A} \bmod q$

⑤ Generate keys for user A
Private → X_A
Public → $\{q, \alpha, Y_A\}$

(6) Generate hash code (m) for PT (M)
$m = H(M)$ $0 \leq m \leq q-1$

(7) Generate random integer k
$1 \leq k \leq q-1$ and $\gcd(k, q-1) = 1$

(8) Now calculate S_1 and S_2
$S_1 = \alpha^k \bmod q$
$S_2 = k^{-1}(m - X_A S_1) \bmod q-1$

(9) Signature pair S_1 and S_2

Now at B's side
Calculate V_1 and V_2

$V_1 = \alpha^m \bmod q$
$V_2 = (Y_A)^{S_1} \cdot (S_1)^{S_2} \bmod q$

If $V_1 = V_2$ → Valid
$V_1 \neq V_2$ → Not Valid