Ajay Kumar Garg Engineering College, Ghaziabad

Department of CSE

Model Solution- ODD Semester (2017-18)

Sessional Test -2

Subject Code	:	NIT-701
Subject Name	:	Cryptography and network security
Names of Faculty Teaching with Signature		1. Ms Neeti Pahuja 2. Ms. Nisha Nisha

Name and Signature of Hod: Mamta Bhusry

Prof. Mamta Bhusry

Divya Gupta

Section-A

(1) What are the requirements for Hash functions?

Ans

Requirements	Description
(1) Variable input size	H can be applied to a block of data of any size
(2) Fixed output size	H produces a fixed-length output
(3) Efficiency	$H(x)$ is relatively easy to compute for any given x, making both hardware & software implementations practical
(4) Pre-image resistant	For any given hash value h it is computationally infeasible to find y such that $H(y) = h$.
(5) Second preimage resistant	For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$
(6) Collision resistant	It is computationally infeasible to find any pair (x, y) s.t $H(x) = H(y)$
(7) Pseudorandomness	Output of H meets standard test for pseudorandomness.

(2) What requirements should a digital signature scheme satisfy?

Ans: (1) The signature must be a bit pattern that depends on the message being signed

(2) The signature must use some information unique to the sender to prevent both forgery & denial

(3) It must be relatively easy to produce the digital signature

(4) It must be relatively easy to recognise & verify the digital signature.

(5) It must be practical to retain a copy of the digital signature in storage

(3) Compare & contrast AES and DES for message encryption

Ans

	DES	AES
Developed	1977	2000
Key Length	56 bits	128, 192 or 256 bits
Cipher Type	Symmetric block cipher	Symmetric block cipher
Block Size	64 bits	128 bits
Security	Proven Inadequate	Considered Secure

(4). Find the value of $3^{201} \bmod 11$

Ans $a^{p-1} \equiv 1 \bmod p$

$a^{\phi(n)} \equiv 1 \bmod n$

$\phi(11) = 10$

$3^{10} \bmod 11 = 1$

$201 = 10 \times 20 + 1$

$3^{201} \bmod 11 = ((3^{10} \bmod 11)^{20} \bmod 11 \times 3^{1} \bmod 11) \bmod 11$

$= ((1)^{20} \bmod 11 \times 3 \bmod 11) \bmod 11$

$= ((1 \times 3) \bmod 11)$

$= 3 \bmod 11$

$\boxed{3^{201} \bmod 11 = 3}$

---

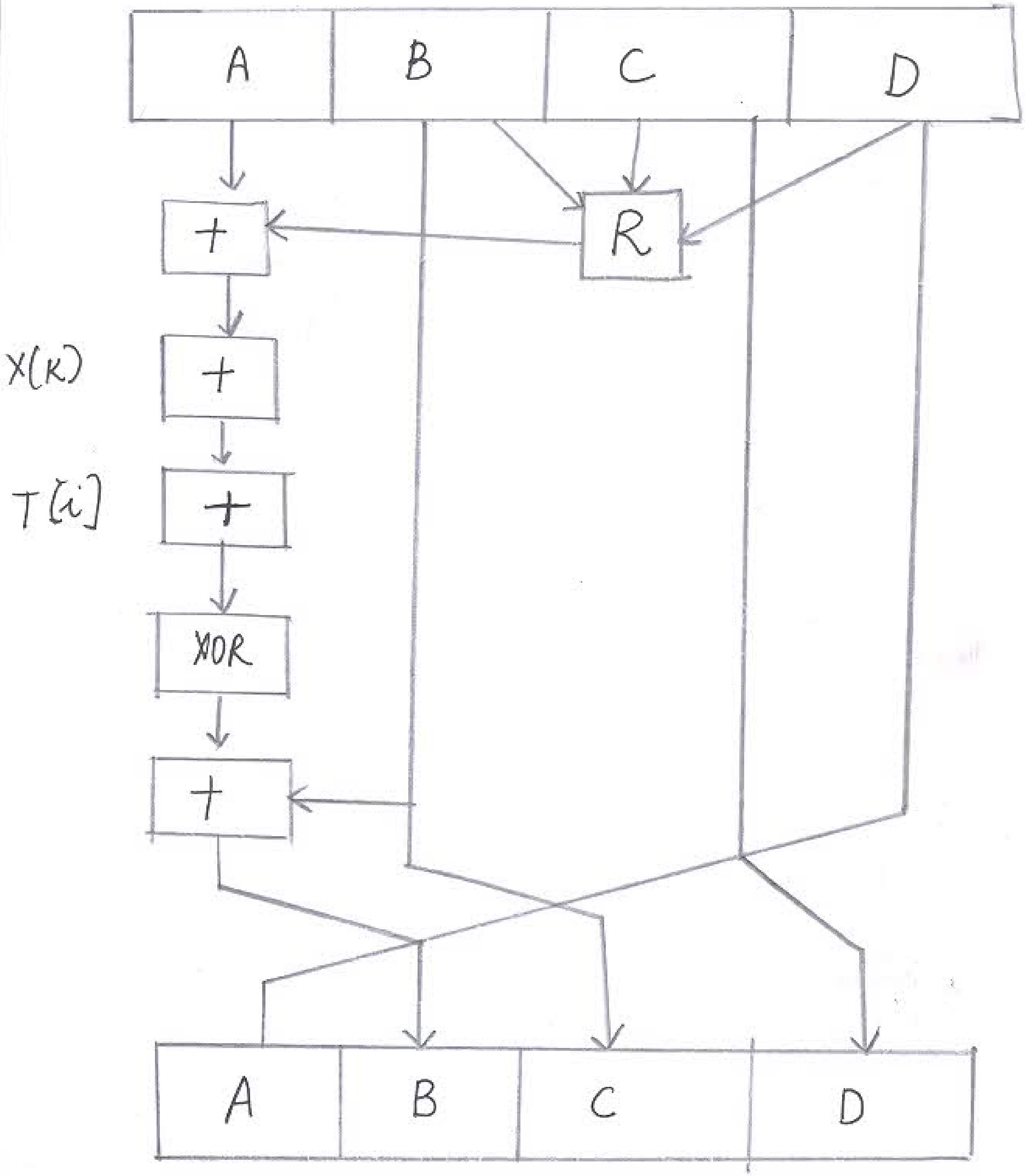
(5) Explain the compression function of MD5 Algorithm for hash calculation.

Ans. Each round has 16 steps of the form

$a = b + ((a + g(b,c,d) + X[K] + T[i]) <<< s)$

- a,b, c, d refers to the 4 words of the buffer, but used in varying permutations where g(b,c,d) is different non-linear function in each round (F, G, H, I)

MD5 Compression function



Section B

(6) State and Prove Euler's Theorem. Compute $\phi(300)$

Ans. Eulers theorem states that for every a and n there are relatively prime :

$$a^{\phi(n)} \equiv 1 (\bmod n) \rightarrow ①$$

Proof Equation ① is true if n is prime, because in that case, $\phi(n) = (n-1)$ and Fermat's theorem holds. However, it also holds for any integer n. Consider the set of such integers,

$$R = \{x_1, x_2 \ldots x_{\phi(n)}\}$$

i.e each element x_i of R is a unique positive integer less than n with $\gcd(x_i, n) = 1$

Now multiply each element by a, modulo n :.

$$S = \{ax_1 \bmod n), (ax_2 \bmod n), \ldots (a x_{\phi(n)} \bmod n)\}$$

The set S is a permutation of R, by following reasons

1. Because a is relatively prime to n and x_i is relatively prime to n, ax_i must also be relatively prime to n.
2. Therefor There are no duplicates in S.

$$\therefore \prod_{i=1}^{\phi(n)} (ax_i \bmod n) = \prod_{i=1}^{\phi(n)} x_i$$

$$\prod_{i=1}^{\phi(n)} ax_i \equiv \prod_{i=1}^{\phi(n)} x_i (\bmod n)$$

$$a^{\phi(n)} \times \left[\prod_{i=1}^{\phi(n)} x_i\right] \equiv \prod_{i=1}^{\phi(n)} x_i (\bmod n).$$

$$a^{\phi(n)} \equiv (\bmod n)$$

which completes the proof.

(b) $\phi(300)$

$$\phi(n) = \prod_{i=1}^{k} (e_k^{i} - e_k^{i-1})$$

$$300 = 3 \times 5^2 \times 2^2$$

$$\phi(300) = (3^1 - 3^0) * (5^2 - 5^1) * (2^2 - 2^1)$$

$$= (3-1)\ (25-5)\ (4-2)$$

$$= (2)\ (20)\ (2)$$

$$= 20 \times 2 \times 2$$

$$\phi(300) = 80$$

---

(7) Explain Euclid Algorithm. Find gcd (1970, 1066) using Euclid's Algorithm.

This Algorithm is used to calculate GCD of two numbers

Euclid (a, b) :

1. $A \leftarrow a$ $\quad B \leftarrow b$
2. if $B = 0$ then gcd (a, b)
3. $R \leftarrow A \bmod B$
4. $A \leftarrow B$
5) $B \leftarrow R$
6) goto step 2.

1. It will store a & b in temporary variables A and B
2. Check if $B = 0$ if Yes, then stop & return A
3. Else find remainder when A is divided by B
4. Replace content of A with B
5. Replace content of B with R
6. Goto check condition again whether $B = 0$ or not

(b)
1 $gcd(1970, 1066) = 1066 \times 1 + 904$
2 $gcd(1066, 904) = 904 \times 1 + 162$
3 $gcd(904, 162) = 162 \times 5 + 94$
4. $gcd(162, 94) = 94 \times 1 + 68$
5. $gcd(94, 68) = 68 \times 1 + 26$
6. $gcd(68, 26) = 26 \times 2 + 16$
7. $gcd(26, 16) = 16 \times 1 + 10$
8. $gcd(16, 10) = 10 \times 1 + 16$
9. $gcd(10, 6) = 6 \times 1 + 4$
10. $gcd(6, 4) = 4 \times 1 + 2$
11. $gcd(4, 2) = 2 \times 2 + 0$
12 $gcd(1970, 1066) = 2$

(8) What are the securities of RSA? Perform encryption and decryption using RSA for $p = 17, q = 11$ if the message $= 88$

Ans: Five possible approaches to attacking the RSA algorithm are.

(1) Brute force: This involves trying all possible private keys

(2) Mathematical attacks: There are several approaches, all equivalent in effort to factoring the product of 2 primes.

(3) Timing attacks: These depend on the running time of the decryption algorithm.

(4) Hardware fault based attack: This involves inducing hardware faults in the processor that is generating digital signatures.

(5) Chosen ciphertext attacks: This type of attack exploits properties of RSA algorithm.

(b) $p = 17, q = 11$

$n = p \times q = 187$

$\phi(n) = \phi(p) * \phi(q)$

$\phi(n) = 16 \times 10 = 160$

$gcd(d, \phi(n)) = 1$

$d = 7$

$d \times e \equiv 1 \bmod \phi(n)$

$7 \times e = 1 \bmod 160$

$e = 23$

Encryption

$C = M^e \bmod n$

$C = 88^{23} \bmod 187$

$C = 88 \bmod 187 = 88$

$= 88^2 \bmod 187 = 77$

$88^4 \bmod 187 = 132$

$88^8 \bmod 187 = 33$

$88^{16} \bmod 187 = 154$

$88^{23} \bmod 187$

$= (154 \times 132 \times 77 \times 88) \bmod 187$

(9) Explain Elgamal scheme of digital signature generation & verification.

Ans. The Elgamal encryption scheme is designed to enable encryption by a user's public key with decryption by the user's private key. For a prime number q, if α is a primitive root of q, then

$$\alpha, \alpha^2, \ldots \alpha^{q-1}$$

are distinct (mod q). It can be shown that if α is a primitive root of q, then

1. For any integer m, $\alpha^m \equiv 1 \pmod q$ if and only if $m \equiv 0 \pmod{q-1}$
2. For any integer, i, j, $\alpha^i \equiv \alpha^j \pmod q$ if and only if $i \equiv j \pmod{q-1}$.

As with Elgamal encryption, the global elements of Elgamal digital signature are a prime number q and a, which is a primitive root of 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; A's public key is $\{q, \alpha, Y_A\}$.

To sign a message M, user A first computes the hash $m = H(M)$, such that m is an integer in the range $0 \leq m \leq q-1$.

A then forms a digital signature as follows.

1. Choose a random integer k such $1 \leq k \leq q-1$ and $\gcd(k, q-1) = 1$
2. Compute $S_1 = \alpha^k \bmod q$. Note it is same as computation of C_1 for Elgamal encryption
3. Compute $k^{-} \bmod (q-1)$. That is compute the inverse of k modulo $q-1$.
4. Compute $S_2 = k^{-}(m - X_A S_1) \bmod (q-1)$.
5. The signature consists of pair (S_1, S_2)

Any user B can verify the signature as follows.

1. compute $V_1 = \alpha^m \bmod q$
2. Compute $V_2 = (Y_A)^{S_1} (S_1)^{S_2} \bmod q$

The signature is valid if $V_1 = V_2$.

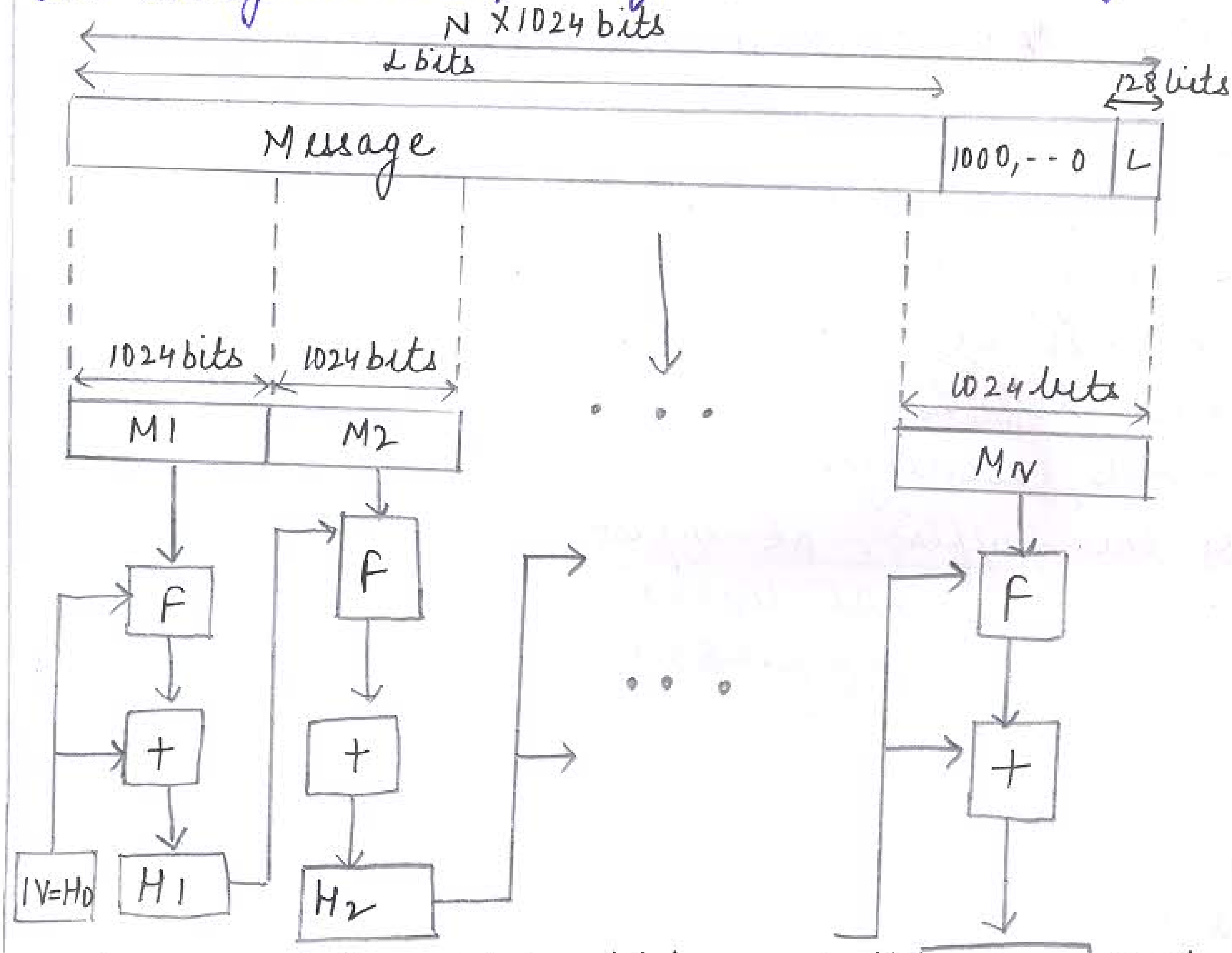
(10) Discuss the logical structure, components and algorithmic steps of SHA-512

Ans. The algorithm takes as input a message with a maximum length of less than 2^{128} bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks.

The process consists of the following steps

STEP 1: Append padding bits. The message is padded so that its length is congruent to 896 modulo 1024. Padding is always added, even if the message is already of the desired length.

STEP 2: Append length. A block of 128 bits is appended to the message. This block is treated as an unsigned 128 bit integer (most significant byte first) and contains the length of the original message (before the padding). The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length.



+ = word by word addition mod 2^{64}

Fig1: Message generation using SHA-512

STEP 3 : Initialize hash buffer. A 512 bit-buffer is used to hold intermediate and final results of hash function. The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h). These registers are initialized as 64 bit integers

These values are stored in big-endian format, which is the most significant byte of a word in the low-address (left most) byte position.

a = 6A09E667F3BCC908   e = 510E527FADE682D1
b = BB67AE8584CAA73B   f = 9B05688C2B3E6C1F
c = 3C6EF372FE94F82B   g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1   h = 5BE0CD19137E2179.

STEP 4 : Process message in 1024-bit (128-word) blocks. The heart of this algorithm is a module that consists of 80 rounds. This is labeled as F.

Each round takes as input the 512 bit buffer value, abcdefgh, and up-dates the contents of the buffer. At input to the first round, the buffer has the value of the intermediate hash value H_{i-1}. Each round t makes use of a 64 bit value W_t, derived from the current 1024-bit block being processed (M_i). The output 18th round is added to the Input to the first round (H_{i-1}) to produce H_i.

STEP 5. Output. After all N 1024 bit blocks have been processed, the output from the Nth stage is 512 bit message digest

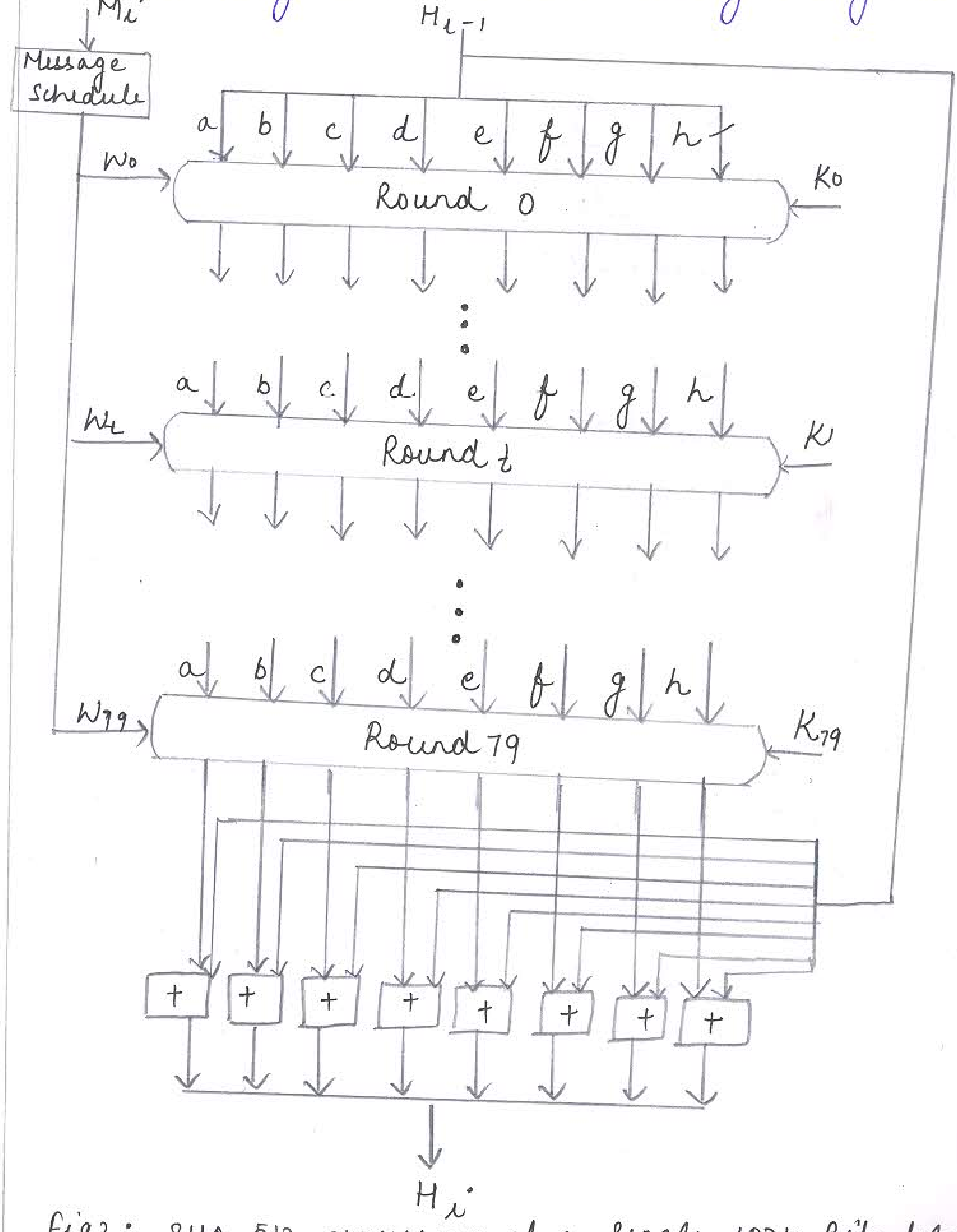


Fig2: SHA-512 processing of a single 1024 Bit block.

so summarizing SHA - 512 as follows

$$H_0 = IV$$
$$H_i = SUM_{64}(H_{i-1}, abcdefgh_i)$$
$$MD = H_N$$

where

IV = initial value of abcdefgh buffer defined in step 3

$abcdefgh_i$ = the output of the last round of the processing of the i^{th} message block

N = the Number of blocks in the message

SUM_{64} = the addition modulo 2^{64} performed separately on each word of the pair of inputs

MD = final message digest value.

SECTION : C

11. Explain Chinese remainder Theorem. Also solve for $x \equiv 2 \bmod 3$, $x \equiv 3 \bmod 5$, $x \equiv 2 \bmod 7$ using chinese remainder Theorem.

Ans. a) If the integers of m_i, where $i = 1,2,3 \ldots n$ are relatively prime in pair, then the congruence $x = a_i \bmod m_i$ where a_i are relatively, integers have one and only one common solution congruent mod.

$$M = \prod_{i=1}^{n} M_i$$

Proof : The given congruences are

$$\left.\begin{aligned} x &\equiv a_1 \bmod m_1 \\ x &\equiv a_2 \bmod m_2 \\ &\vdots \\ x &\equiv a_n \bmod m_n \end{aligned}\right\} \rightarrow I$$

$$M = m_1 \times m_2 \times \ldots \times m_n.$$

$$M_1 = \frac{M}{m_1}, \quad M_2 = \frac{M}{m_2} \quad \ldots \quad M_n = \frac{M}{m_n}.$$

Let us consider the n congruence given by

$$M_i x \equiv 1 \bmod m_i \longrightarrow II$$

in each case gcd $(M_i, m_i) = 1$ because m_i is relatively prime to $m1, m2 \ldots m_n$ & to their product.

Therefore each of congruence has exactly one solution that is

$$x \equiv x_i \bmod m_i \longrightarrow III$$

Now let us consider,

$X = M_1 x_1 a_1 + M_2 x_2 a_2 + \cdots M_n x_n a_n \rightarrow \underline{IV}$

Let us prove that X given by Equation $\underline{IV}$ is a solution of first congruence given by Equation (i)

So $m_1 \mid M_2 x_2 a_2$

Similarly $m_1 \mid M_3 x_3 a_3$

Thus $m_1 \mid M_2 x_2 a_2 + M_3 x_3 a_3 + M_n x_n a_n \rightarrow \underline{IV}$

Again from Eq^n (ii) & (iii).

x_1 is a solution of $M_1 x \equiv 1 \bmod m_1$

Therefore, $M_1 x_1 \equiv 1 \bmod m_1$

multiply both sides by a_1

$M_1 x_1 a_1 \equiv a_1 \bmod m_1$

$\therefore m_1 \mid M_1 x_1 a_1 - a_1 \rightarrow \underline{VI}$

adding Eq(IV) & (VI).

$m_1 \mid M_1 x_1 a_1 + M_2 x_2 a_2 + \cdots M_n x_n a_n - a$

$\therefore m_1 \mid x - a_1$

$\therefore X \equiv a_1 \bmod m_1$

Therefore, X is a solution of first congruence given by Eq (1) since X is a sol. of each congruences given by Eq.

$\therefore X$ is defined in Eq (IV) is a common solution of the congruence given by Eq(1).

Let X & Y be any 2 common solutions of the congruence given by Eq(1).

$X = a_i \bmod m_i$

$Y = a_i \bmod m_i$

$$y = x \bmod m_i$$

$$\frac{y-x}{m_i} = k \text{ (say)}$$

$$y = x + k m_i$$

$$\boxed{y = x + kM}.$$

(b) $x \equiv 2 \bmod 3$

$x \equiv 3 \bmod 5$

$x \equiv 2 \bmod 7$

$M = 3 * 5 * 7 = 105$

$M1 = \frac{M}{m_1} = \frac{105}{3} = 35$

$M2 = \frac{M}{m_2} = \frac{105}{5} = 21$

$M3 = \frac{M}{m_3} = \frac{105}{7} = 15$

$M_1 y_1 = 1 \bmod m_1$

$(35) y_1 = 1 \bmod 3$

$y_1 = (35)^{-} \bmod 3$

$= (2)^{-} \bmod 3$

$y_1 = 2$

$M_2 y_2 \equiv 1 \bmod m_2$

$(21) y_2 \equiv 1 \bmod 5$

$y_2 = (20)^{-} \bmod 5$

$= (1)^{-} \bmod 5$

$y_2 = 1$

$M_3 y_3 = 1 \bmod m_3$

$(15) y_3 = 1 \bmod 7$

$y_3 = (15)^{-} \bmod 7$

$= (1)^{-} \bmod 7$

$y_3 = 1$

$$X = \left(\sum_{i=1}^{3} a_i M_i y_i \right) \bmod M$$

$$X = (2 \times 35 \times 2 + 3 \times 21 \times 1 + 2 \times 15 \times 1) \bmod 105$$

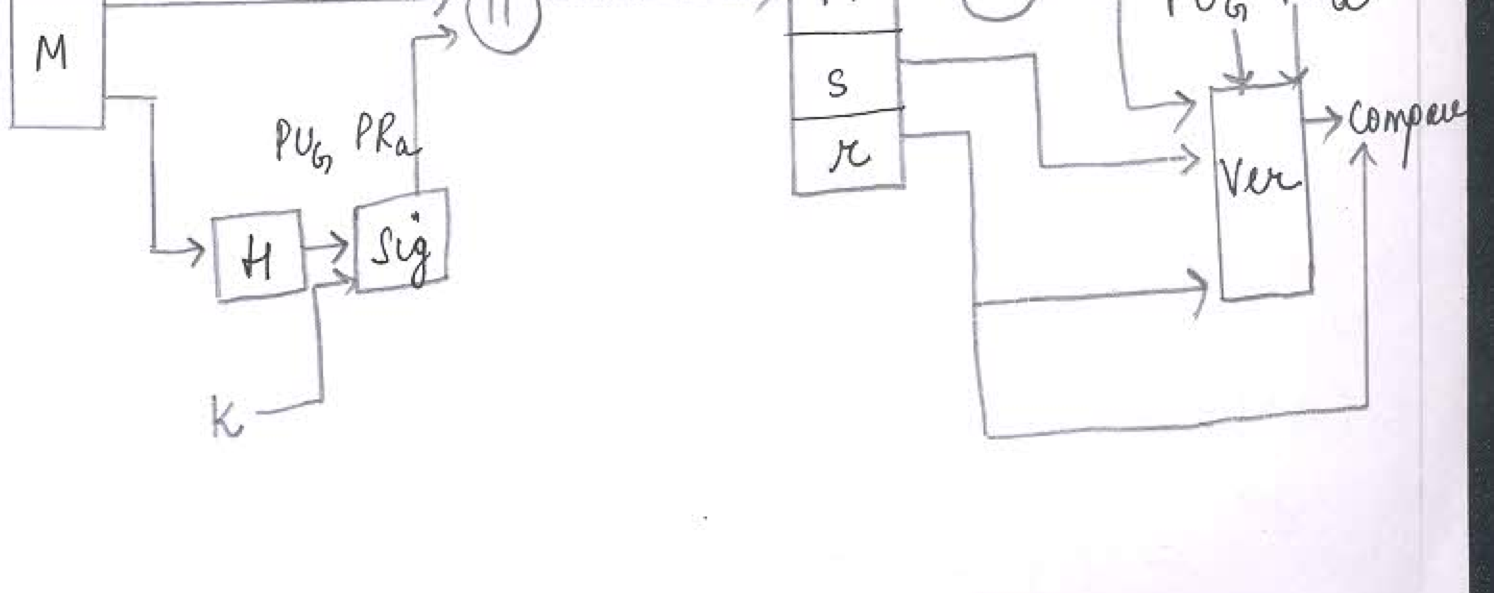
$$= (140 + 63 + 30) \bmod 105$$

$$= 233 \bmod 105 = 23.$$

12 Write signature generation & verification process of digital signature Algorithm of DSS.

Ans: US Govt. approved signature scheme designed by NIST & NSA in early 90's. it published as FIPS 186 in 1991, revised in 1993. It uses hash algorithm. DSA is digital signature only unlike RSA, is a public key technique

DSS Approach



DSA Algorithm

→ It creates a 320 bit signature with 512–1024 bit security. It is smaller and faster than RSA. It is a digital signature scheme only. Its security depends on difficulty of computing discrete logarithms

→ It is a variant of EIGamal & Schnorr schemes

DSA key Generation

→ Let shared global public key values (p,q,g): choose 160-bit prime number q

→ Choose a large prime p with $2^{L-1} < p < 2^L$

- where $L = 512$ to 1024 bits and is a multiple of 64 such that q is a 160 bit prime divisor of $(p-1)$ → choose $g = h^{(p-1)/q}$ where $1 < h < p-1$ and $h^{(p-1)/q} \bmod p > 1$

→ users choose private & compute public key

- choose random private key : $x < q$
- compute public key : $y = g^x \bmod p$

DSA Signature Creation

→ To sign a message M, the sender : generates a random signature key k, $k < q$

→ nb. k must be random, be destroyed after use & never be reused.

Then computes signature pair :

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

→ sends signature (r, s) with message M

DSA Signature Verification

→ having received M & signature (r, s)

→ to verify a signature, recipient computes:

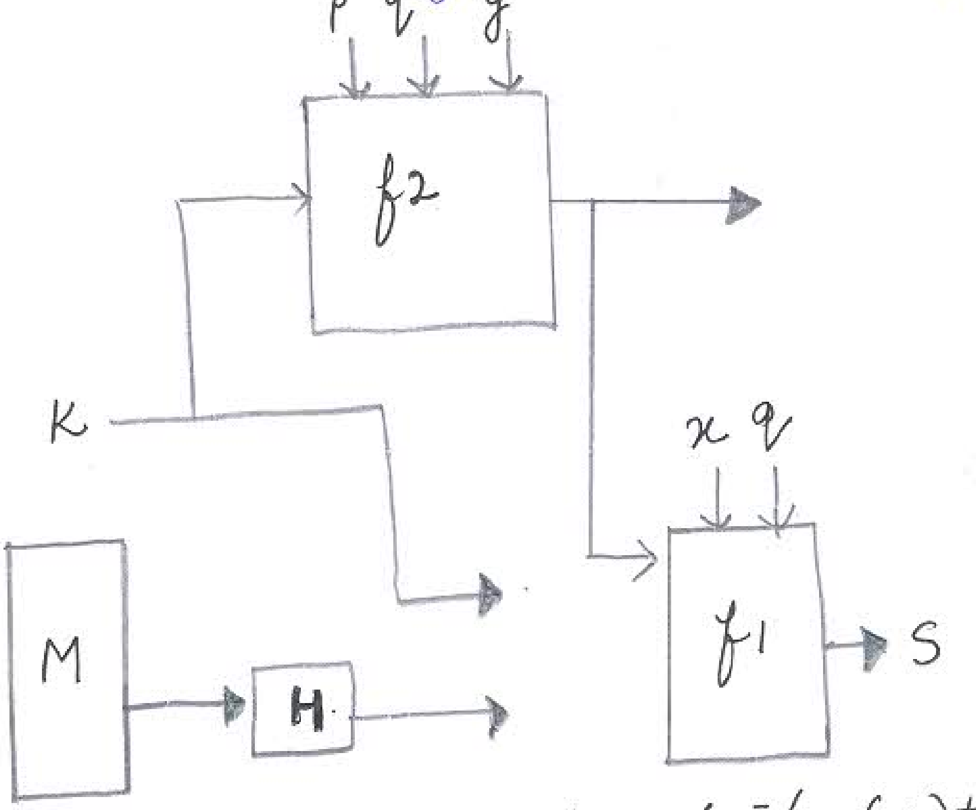
$w = s^{-1} \bmod q$

$u1 = [H(M)\, w] \bmod q$

$u2 = (rw) \bmod q$

$v = [(g^{u1}\, y^{u2}) \bmod p] \bmod q$

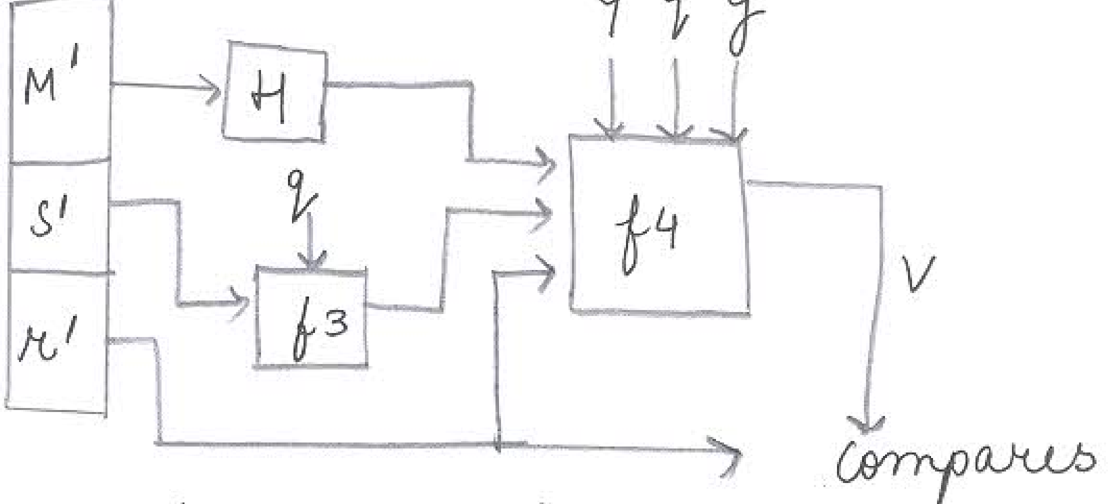
→ if $v = r$ then signature is verified



$s = f1(H(M), K, x, r, q) = (K^{-1}(H(M) + x r)) \bmod q$

$r = f2(K, p, q, g) = (g^K \bmod p) \bmod q$

(a) Signing



$w = f3(s', q) = (s')^{-1} \bmod q$

$v = f4(y, q, g, H(M'), w, r')$

$= ((g^{(H(M')w) \bmod q}\, y^{r'w \bmod q}) \bmod p) \bmod q$

Verifying