UNIT-3

PART-A

Cryptographic Hash Functions.

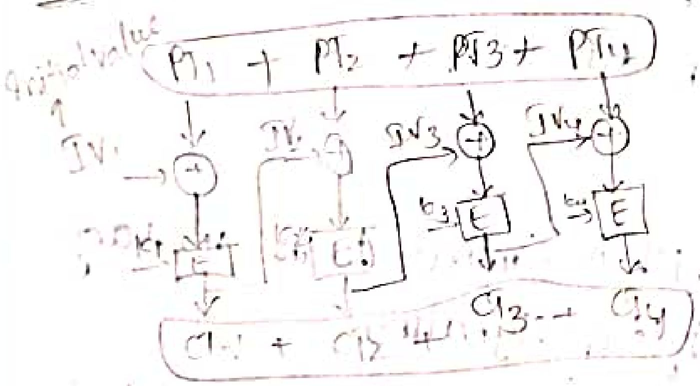
(1) Applications of cryptographic Hash Functions.
(2) Two simple Hash Functions.
(3) Requirements and security. } — 7M
(4) Hash Functions based on cipher Block chaining (CBC)
(5) SHA2 (Secure Hash Algorithm)
(6). SHA-3 } — 7M

PART-B

Digital signature:

(1) ELgamal Digital signature scheme
(2) Schnorr Digital signature scheme
(3) NIST Digital signature Algorithm. } — 7M

CBC



PART-A: Cryptographic Hash Functions.

(1) Applications of cryptographic Hash Functions.

(a) Message authentication.

(1) Message authentication is a mechanism or service used to verify the integrity of a message.

(2) Message authentication assures that data received are exactly as sent (ie, there is no modification, insertion, deletion or replay)

(3) In many cases, there is a requirement that the authentication Mechanism assures that purported identity of the sender is valid.

(4) When a hash function is used to provide message authentication the hash function value is often referred to as a "Message digest".

(5) The essence of the use of a hash function for message integrity is as follows.

(6) The sender computes a hash value as a function of the bits in the message and transmits both the hash value and the message. The Receiver performs the same hash calculation on the message bits and compares this value with the incoming hash value.

(7) If there is a mismatch, The receiver Knows that the message (or possibly the hash value) has been altered.

(b) Digital signatures:

(1) Another important application, which is similar to the message authentication application, is the digital signature.

(2) The operation of the digital signature is similar to that of the MAC

(3) In case of the digital signature, The hash value of a message is encrypted with a user's private Key.

(4) Anyone who knows the user's public Key can verify the integrity of the message that is associated with the digital signature

(5) In This case, an attacker who wishes to alter the message would need to know the user's private key.

(2) Two Simple Hash Functions.

(1) The Basic Hash functions are categorized into two types, They are

(1) Bit-By-Bit XOR

(2) Block-By-Block XOR using CBC.

→ (1) Bit-By-Bit XOR.

(1) A Bit-By-Bit XOR is expressed as $\boxed{C_i = b_{i1} \oplus b_{i2} \oplus \cdots \oplus b_{im}}$

where C_i = i^{th} Bit of Hash code. $(1 \le i \le n)$

m = number of n-bit blocks in the input.

b_{ij} = i^{th} bit in j^{th} Block

$\oplus$ = Exclusive OR (XOR) operation

(2) The above operation produces "simple parity bit" for each bit position.

procedure:

(1) Initially set the n-bit hash value to zero.

(2) process the successive n-bit block of data as

(a) Rotate the current Hash value to the left by 1-bit

(b) XOR the block into hash value.

(Pg-347 : Diagram)

→ (2) Block-By-Block XOR using CBC (cipher block chaining).

(1) In Block-By-Block XOR, it is assumed that the block size is 64-bit. Hence, it is treated as 64-bit-By-64-bit XOR.

(2) Given a message which is denoted by 'M' which consists of a sequence of 64 bit blocks. That are denoted by $X_1, X_2, \ldots X_N$

(3) Define the Hash code as $\boxed{h = H(M)}$ → Message

Hash code / hash value   Hash function

(4) The above hash code is appended as Block-By-Block XOR (64-bit), which results as

$$\boxed{h = X_{N+1} = X_1 \oplus X_2 \oplus \cdots \oplus X_N}$$

(5) Now, we encrypt the entire message with hash code by using CBC (Cipher Block chaining)

3. Requirements and Security.

(1) The Basic Requirements (properties) of a Hash Function are

- (1) Variable Input size : It can be applied to any block of data with any size.
- (2) Fixed output size : The Hash function must produce a fixed length output only.
- (3) Efficiency : $H(x)$ is a relatively easy to compute for any value of 'x' (Message).
- (4) pre-image Resistant (one-way property):

  For any given Hash value (h), it is not possible to compute 'y' where $\boxed{H(y) = h}$
- (5) Second pre-image Resistant (weak collision Resistant)

  For any given block (x), it is possible to compute 'y' $\neq x$. where

  $\boxed{H(y) = H(x)}$
- (6) collision Resistant (strong collision Resistant)

  It is not possible to compute to find a pair (x, y) where $x \neq y$, such that $\boxed{H(x) = H(y)}$
- (7) pseudo Randomness:

  The output of H must meet the standard tests of pseudo-Randomness.

Security:

→ Security must be provided from various attacks.

→ The various security attacks that may effect the Hash function are

(1) Brute-Force Attack
(2) Collision Resistant Attack
(3) pre-image and second pre-image attacks.
(4) CryptAnalysis Attacks.

4. Hash Functions Based on cipher Block Chaining (CBC)

(1) The Hash functions that are based on CBC (Cipher Block chaining) are preffered to avoid "meet-in-the-middle" attack

procedure:

Assume an opponent which intercepts a message with a signature in the form of encrypted hash code which is m-bits long.

conditions:

(1) Calculate the unencrypted hash code 'G'
(2) Construct any desired message in the form $Q_1, Q_2, \dots Q_{N-2}$
(3) Compute the $H_i = E(Q_i, H_{i-1})$ for $1 \le i \le N-2$
(4) Generate $2^{m/2}$ random blocks

for each block X, compute $E(X, H_{N-2})$

for each block Y, compute $D(Y, G)$

(5) With high probability, There will be X and Y such that

$$E(X, H_{N-2}) = D(Y, G)$$

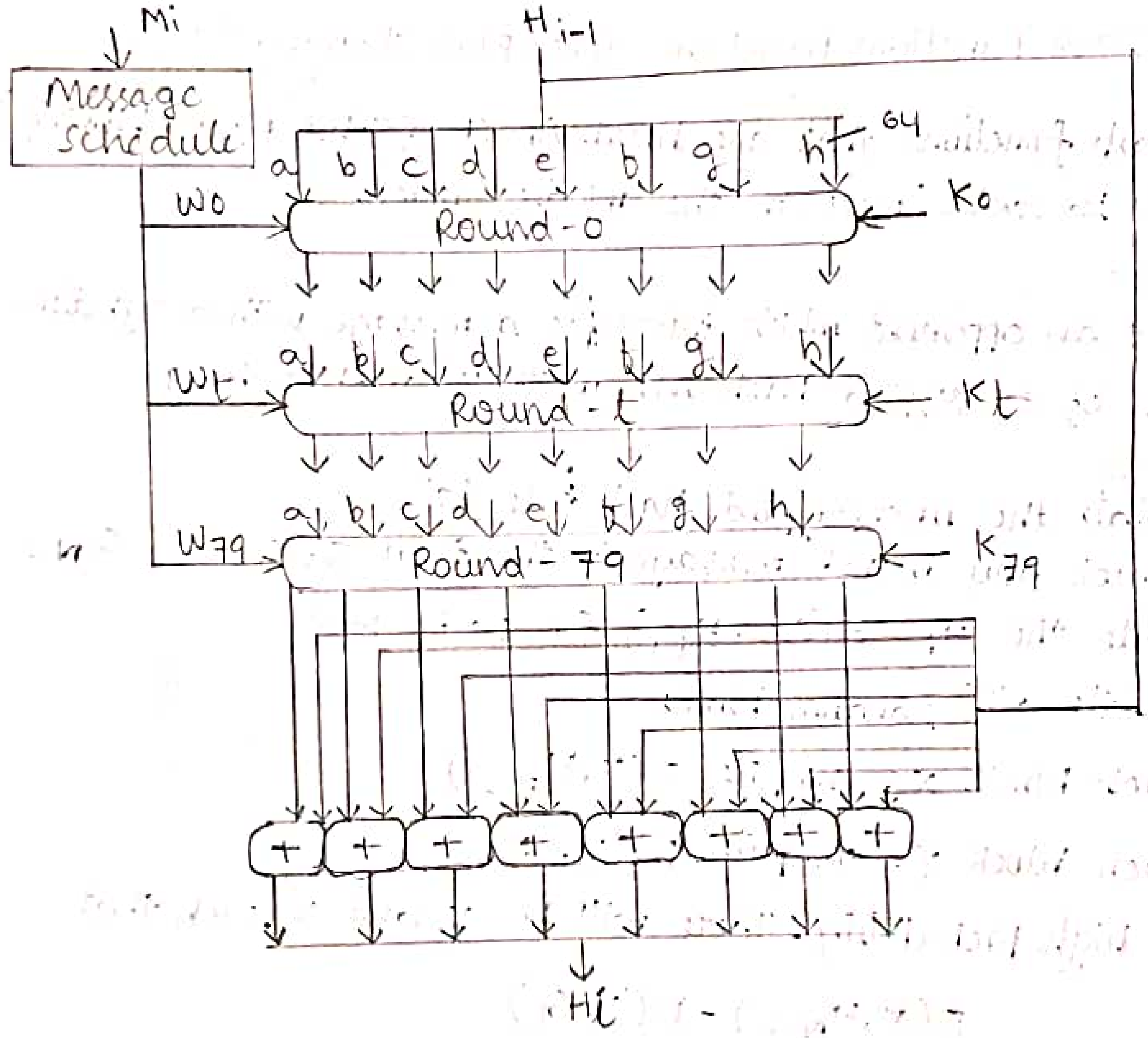
(6) Now, form the message as $Q_1, Q_2, \dots Q_{N-2}, X, Y$.

The above message has hashcode 'G' and Therefore it can be used with intercepted encrypted signature.

5. SHA

→ SHA Stands for Secure Hash Algorithm.
→ The Various SHA algorithms are SHA1, SHA-224, SHA-256, SHA-384, SHA-512 .... etc..,
→ In SHA, The data and the certificates are hashed.

→ SHA is a modified Version of MD-5. (Message Digest)
→ SHA uses bitwise operations mod, modular additions and compression functions.
→ The main goal of SHA is to reduce the input data.



Procedure:

(1) Append padding bits : The message must be padded so that the length is congruent to 896 modulo 1024 ; eventhrough The message is of desired length.

Therefore, The number of padding bits will be in between 1 to 1024.

The padding bits starts with 1 bit followed by necessary number of zeroes.

(2) Append length : A block of 128 bits are appended to the message

∴ The actual message will be in the form of 1024 bit blocks. which are represented as $M_1, M_2, \ldots M_N$, where the total length is represented as $N \times 1024$ bits.

(3) Initialize Hash buffer: A 512-bit buffer is used to hold the intermediate and the final results of Hash function.

Buffer is represented as: 8 registers namely ~~(A, B, C, D, E, F, G, H)~~ a, b, c, d, e, f, g, h which holds 64-bit each

The 8-registers are initialized with hexa-decimal values.

(4) process 1024-bit block:

The SHA has 80 rounds, where each round takes 512-bit buffer values. The intermediate hash value is denoted by H_{i-1}

Each round takes 64-bit (W_i) from the current 1024 bit (M_i) block.

Each round takes additive constant (K_t) which indicates one of the 80 rounds. where $0 \leq t \leq 79$

(5) output:

The output of 80^{th} round is added to the input of first round (H_{i-1}) to produce H_i which must be done independently for each 8 words of buffer.

After all $N \times 1024$ bits blocks have been processed, the output from the N^{th} stage is a 512-bit Hash value.

6. SHA-3

(1) SHA-3 uses "sponge construction scheme".

(2) SHA-3 takes the input messages and partitions it into fixed sized blocks.

(3) Each block is processed and it becomes the input for the next iteration.

(4) The sponge function has 3-parameters, They are.

(1) f = It is the internal function which is used to process each input block.

(2) r = It is the size of the input blocks (bits).

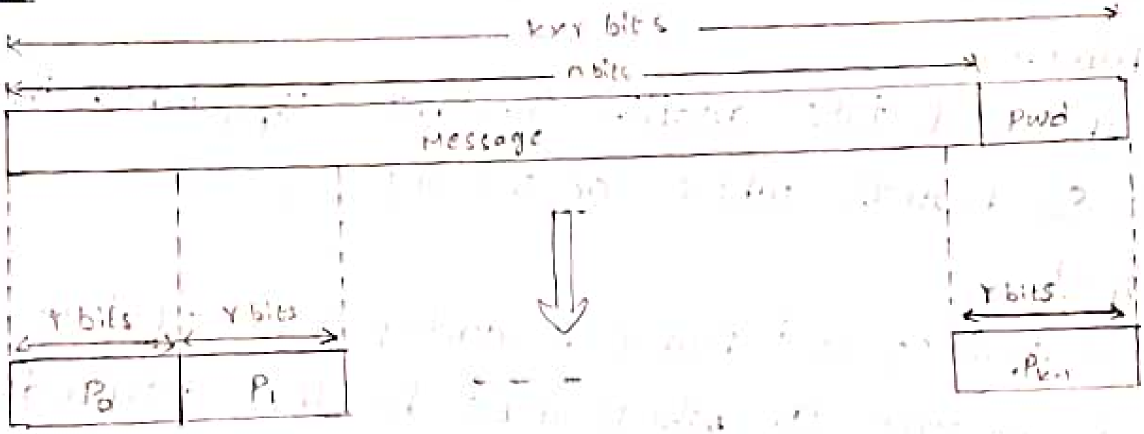
(3) pad = It is the padding algorithm.

(5) The sponge function allows both variable inputs and variable outputs.

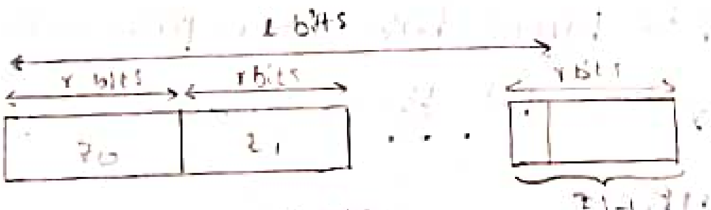
(6) The input message is "n bits" which is partitioned to "k fixed sized blocks" of "r bits each."

(7) The message is padded to achieve the length.

(8) The return partition is the sequence of blocks, $P_0, P_1, \ldots P_{K-1}$ with length <u>K×r</u>.



(a) input



(b) output

Sponge Function Construction:

→ The sponge specification proposes two padding schemes, They are

(1) simple padding

(2) Multirate padding.

(1) Simple padding: It is denoted by pad 10^* which appends single bit '1' followed by minimum number of bits '0'

(2) Multirate padding : It is denoted by pad 10^*1. which appends single bit '1' followed by minimum number of bits '0' and later by single bit '1'.

→ The sponge function consists of two phases. They are

(1) Absorbing stage / phase

(2) Squeezing phase.

(1) Absorbing phase : The input block is padded with zero's which extends 'r' bits to bits.

→ The above content is XOR with message block and 's' is formed.

→ The output is the value of 's' of the next iteration.

(2) Squeezing phase : The desired output denoted by 'l' where $l \leq b$

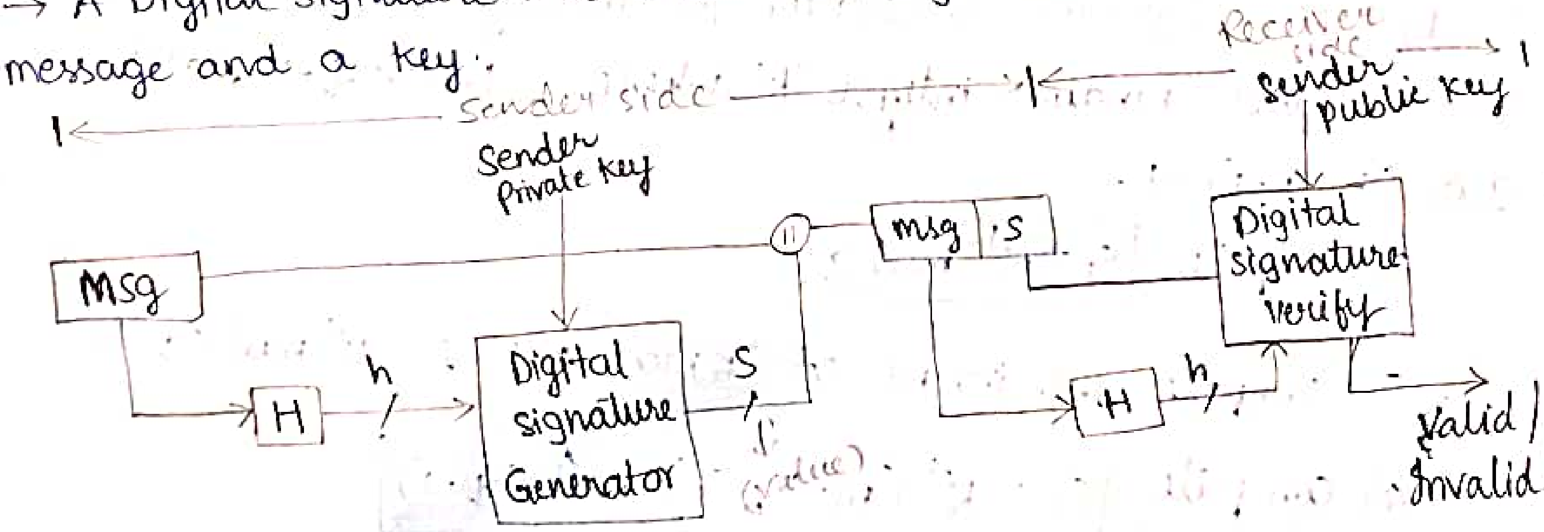
→ After absorbing phase the first 'l' bits of 's' are returned and then the sponge construction gets terminated.

PART-B

Digital Signature

→ A Digital signature is a public key primitive for message authentication.

→ A Digital signature is a value which is generated by combining the message and a key.



1. Elgamal Digital signature scheme.

1) The Elgamal Encryption scheme is designed to enable the encryption by a user's public Key and decryption by user's private key.

2) The Elgamal digital signature scheme involves the use of private key for digital signature generation and uses the public key for digital signature verification.

3) Elgamal digital signature uses primitive ~~rates of~~ in number Theory.

4) For a prime number 'q' if α is a primitive root of 'Q' Then,

$\alpha, \alpha^2, \alpha^3, \ldots \alpha^{q-1}$ are distinct mod q.

Algorithm:

a) private / public Key Generation:

1) Generate a random integer X_A where $1 < X_A < q-1$

where A: Sender, X: private key.

2) Compute $\boxed{Y_A = \alpha^{X_A} \bmod q}$ where A: sender, Y: public key

3) sender private key $= X_A$

sender public key $= Y_A$

b) Digital signature Generation for message M:

1) Choose random integer 'K' where $1 < K < q-1$, and $\gcd(K, q-1) = 1$.

2) compute $\boxed{S_1 = \alpha^K \bmod q}$

3) Compute the inverse of $K \bmod q-1$ ie, $K^{-1} \bmod (q-1)$

4) Compute $\boxed{S_2 = K^{-1}(m - X_A S_1) \bmod (q-1)}$

$\therefore$ Digital signature = pair (S_1, S_2)

c) Digital signature verification:

1) calculate $\boxed{V_1 = \alpha^M \bmod q}$

2) calculate $\boxed{V_2 = (Y_A)^{S_1} (S_1)^{S_2} \bmod q}$

3) If $V_1 = V_2$ then Digital signature is valid otherwise invalid

3) compare the above signatures V_1 and V_2 and if both are same, Digital signature is valid; else, Digital signature is Invalid.

2. Schnorr Digital signature Algorithm.

1) The schonrr algorithm is based on descrete logarithm.
2) schnorr algorithm minimizes the computation time to generate signature
3) It requires multiplying 2n-bit integer with n-bit integer.
4) This algorithm is based on a prime mod p with p-1 having a prime factor of q ie, $\boxed{p-1 \equiv \phi \bmod q}$
5) The use $p = 2^{1024}$ and $q = 2^{160}$ where 1024 and 160 are no. of bits.

Algorithm:

a) private / public Key Generation

1) choose primes 'p' and 'q' where 'q' is a prime factor of 'p-1'.

2) choose an integer 'a' such that, where $a^q \equiv 1 \bmod p$.

3) choose random integer 's' where $0 < s < q$ (s is the private key).

4) calculate $\boxed{v = a^{-s} \bmod p}$

b) Generate Digital signature:

1) choose random integer 'r' where $0 < r < q$ and compute

$$\boxed{x = a^r \bmod p}$$

2) Concatenate the message with 'x' and Hash the result into 'e'.

$$\boxed{e = H(M \| x)}$$

3) Compute $\boxed{y = (r + se) \bmod q}$

4) Hence the Digital signature is the pair (e, y)

c) Digital signature verification:

1) compute $x' = a^y v^e \bmod p$

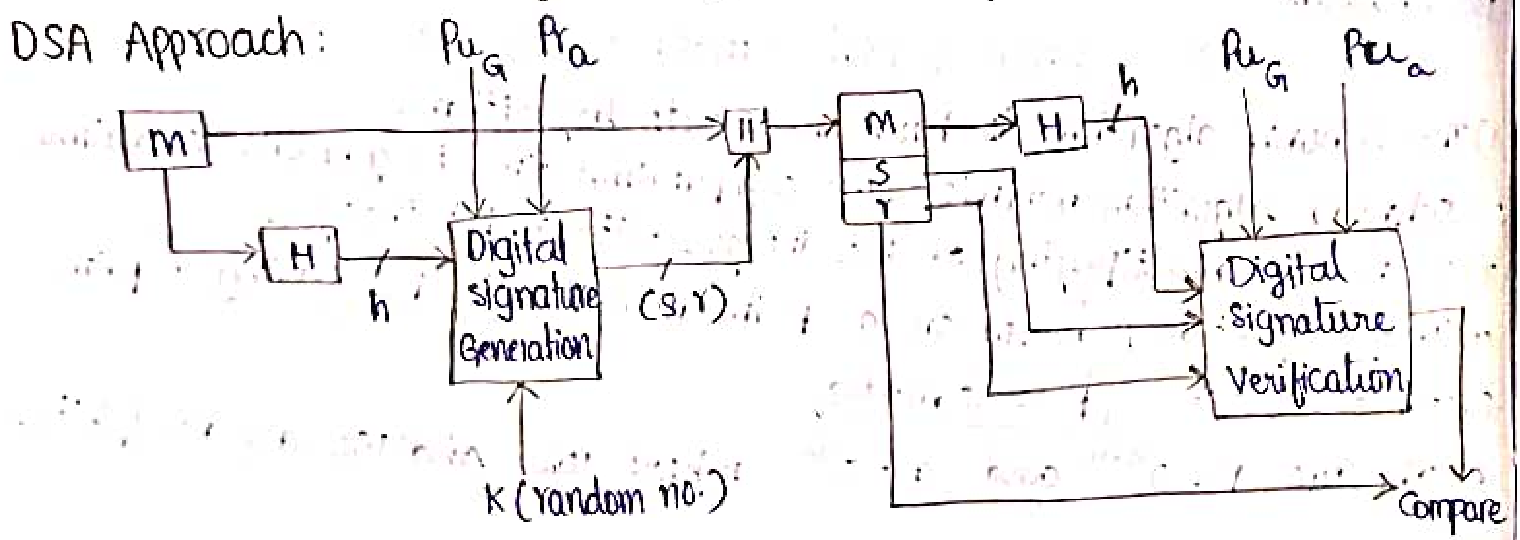
2) verify that $e = H(M \| x')$

3). To verify the digital signature we check $H(M\|x') = H(M\|x)$

4). The above values are same, the digital signature is valid, else invalid.

3. NIST Digital signature Algorithm (DSA)

DSA Approach:



1) NIST stands for National Institute of standards and Technology

2) NIST developed FIPS-186 digital signature, Algorithm (Federal information processing standard)

3) The various versions of FIPS-186; FIPS-186-2, FIPS-186-3, FIPS-186-4.

4) The FIPS-186 can also be called as DSA (digital signature Algorithm)

5) The DSA algorithm uses SHA.

6) The DSA provides only the digital signature function.

Algorithm:

a) public / private Key Generation:

1) choose a random integer, $x, 0 < x < q$ (private key)

2) choose a random integer, $K, 0 < K < q$ (secret number)

3) Generate a public key y; where $\boxed{y = g^x \bmod p}$ (public key)

Here 'p' is a prime number, such that $2^{L-1} < p < 2^L$, where $512 \le L \le 1024$.

$$\boxed{g = h(p-1)/q}$$

b) Digital Signature Generation:

1) Calculate r, where $\boxed{r = (g^K \bmod p) \bmod q}$

2) calculate s, where $\boxed{s = [K^{-1}(H(M) + xr)] \bmod q}$,

3) Hence the digital signature generated is pair (r, s)

c) Digital signature verification:

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

2) calculate $\boxed{U_1 = [H(M')w] \bmod q}$.

3) calculate $\boxed{U_2 = (r')w \bmod q}$

4) calculate $\boxed{V = [(g^{U_1} y^{U_2}) \bmod p] \bmod q}$

Now compare V & r', If $V = r'$, then digital signature is valid, else digital signature is invalid.