MODULE - 2

Public Key Cryptosystem.

~~The~~ f

RSA Operation.

The first step in the RSA is used to generate a pub key & prt key pair.
This is usually a one-time operation unless an individual needs to obtain a fresh one for security reason.

Key Generation Process of RSA.

1] Choose two large prime numbers of same size p and q.
[Typically each p & q has b/w 512 to 2048 bits].

2] Compute $n = p*q$ and $\phi(n) = (p-1)*(q-1)$.

3] Select e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.

4] Compute d such that $1 < d < \phi(n)$ and
$e*d \equiv 1 \bmod \phi(n)$ or
$e*d \bmod \phi(n) = 1$.
knowing $\phi(n)$ makes d easy to compute.

Euler's $\phi(n)$ (totient). is used for a given +ve integer 'n' i.e $\phi(n)$ is the no. of +ve ints less than or equal to n that are co-prime to n.
Eg: $\phi(8) = 1,3,5,7.$ | $\phi(7) = 1,2,3,4,5,6.$

$\phi(\text{prime}) = (\text{prime} - 1)$.

Public key = (e, n).
Private key = (d, n).

Encryption

Let m be a plaintext msg for each block m_i the corresponding Cipher text (C_i) is calculated as:

$$\boxed{C_i = m_i^e \bmod n}.$$

Decryption

Given a block of txt C_i, the corresponding plain text m_i is:

$$\boxed{m_i = C_i^d \bmod n}.$$

Find out the Cipher text & decipher the message "HIDE" using RSA for $p = 3$ $q = 11$. and choose

$p = 3$ $\quad n = 33$ $\qquad$ H I D E
$q = 11$ $\quad \phi(n) = (3-1)(11-1)$ $\qquad$ 7 8 3 4
$= 20$.

$e = 7$

$C_i = 7^7 \bmod 33$ $\qquad 7y \equiv 1 \bmod 20$.
$d = 3$ $\quad = 28$. $\qquad \underline{y = 3}$ (d)

H

$m_i = 28^3 \bmod 33$
$= 21952 \bmod 33$
$= 7$.

I
$C_i = 8^7 \bmod 33$
$= 2097152 \bmod 33$
$= 2$.
$m_i = 2^3 \bmod 33$ $\qquad 8y \equiv$ ~~7 mod 33~~.
$= 8$. (2^3). $\qquad$ ~~= 4~~

$$\left.\begin{aligned} C_i &= 3^7 \bmod 33 \\ &= 2187 \bmod 33 \\ &= 9. \\ m_i &= 9^3 \bmod 33 \\ &= 729 \bmod 33 \\ &= 3. \end{aligned}\right\} D.$$

$$\left.\begin{aligned} C_i &= 4^7 \bmod 33. \\ &= 16384 \bmod 33. \\ &= 16. \\ m_i &= 16^3 \bmod 33. \\ &= 4096 \bmod 33 \\ &= 4. \end{aligned}\right\} E$$

$C_i = m_i^e \bmod n.$ $n = 33$

$= 30^7 \bmod 33.$

$= 30^1 \times 30^2 \times 30^4 \bmod 33$

$C_i = 24.$

H	I	D	E.

7B

0 1 1 1 1 0 1 1

$30^{40}.$

$m_i = 24^3 \bmod 33$

$= 13824 \bmod 33$

$m_i = 30.$

$5^{500} \bmod ~~4~~ 40.$

~~e = 7~~

~~d = 3.~~

$C_i = 5^{500} \bmod 40.$

500.

$$\begin{array}{ccccccccc} ✓ & ✓ & ✓ & ✓ & ✓ & & ✓ & & \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 2^8 & 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \end{array}$$

$5^{2} \bmod 33 = 25.$

$5^4 \bmod 33 = 625 \bmod 33$

$= 31.$

$5^5 = 5^2 . 5^2 . 5^1$

$2^8 = 256.$

$2^7 = 128.$

$2^6 = 64.$

$2^5 = 32$

$2^4 = 8.$

$2^2 = 4.$

Performance of RSA.

Time Complexity

Encryption

$$C_i = m_i^e \bmod n.$$

$$= O(b^2).$$

Decryption

$$m_i = C_i^d \bmod n$$

$$= O(nb^2)$$

▸ Both Encryption & Decryption involves repetitive multiplication of b no. of bits.
▸ Unoptimized multiplication of two b-bit no's & reduction by modulo n (division) which takes $O(b^2)$ time.
▸ The encryption key is usually small integer e relative to n.

▸ The time complexity of encryption is $O(b^2)$.

▸ Decryption on other hand involves raising a b-bit no. to the power of 'd' which implementation of decryption involves d multiplications.

▸ Since d is same order as n the complexity of decryption operation $O(nb^2)$.

Speeding up RSA.

We can speed up the decryption of Cipher Text by computing,

$$m_i = C_i^d \bmod n.$$

$C, C^2, C^3, C^4, C^8, C^{16}, \ldots$ upto the max of d b-bits term.

We multiply elements in this series whose positions corresponds to 1 in the binary representation of the decryption key d.

Ofcourse, each multiplication is mod n multiplication so, the intermediate products are never more than b-bits wide.

This approach with first computes square followed by product is referred as Square and Multiply Technique, which speeds up the decryption concept in RSA.

Eg: Write square & multiplication steps for decryption key = 57.

Dec = 57

$m_i = C_i^{57} \bmod n.$ Bin = 111001

1	1	1	0	0	1
2^5	2^4	2^3	2^2	2^1	2^0

$C^{32} \bmod n \times C^{16} \bmod n \times C^8 \bmod n \times C^1 \bmod n$

$$= C^{32} . C^{16} . C^8 . C^1 \bmod n.$$

~~Applications of RSA~~

$5^{40} \bmod 7$.

40.

1	0	1	0	0	0
5	4	3	2	1	0
32	16	8	4	2	0

~~8~~ 32

$5^{8} \bmod 7 = 4$.

$390625 \bmod 7$

$5^{4} \bmod 7 = 2$.

~~8~~

$5^{16} \bmod 7 = (5^{8})^{2} \bmod 7$.
$= 16 \bmod 7 = 2$.

$5^{32} \bmod 7 = (5^{16})^{2} \bmod 7$.
$= 4 \bmod 7 = 4$.

$5^{40} \bmod 7$.

$= (5^{32} \times 5^{8}) \bmod 7$
$= 4 \times 4 \bmod 7$.
$= 16 \bmod 7$,
$= \underline{\underline{2}}$.

Applications of RSA.

Providing msg confidentiality, msg integrity and authentication.

In summary, the principle drawback of pub key cryptography is speed,
while the principle ~~of~~ drawback of secret key cryptography is key management,
To combine (the speed of secret key cryptography & the convenience of public key cryptography, a session key is used.

Choose a fresh random no. 'S' as the secret key. This is referred to as session key.

The sender:

> Encrypts the msg with session key. $[E_S(m)]$.

> Encrypts the session key with the recipient's public key. $[E_{B.pu}(S)]$.

> Sends the encrypted msg & the encrypted session key in the same msg.

The receiver.

> Uses his pvt key to decrypt the part of the msg containing the encrypted session key.

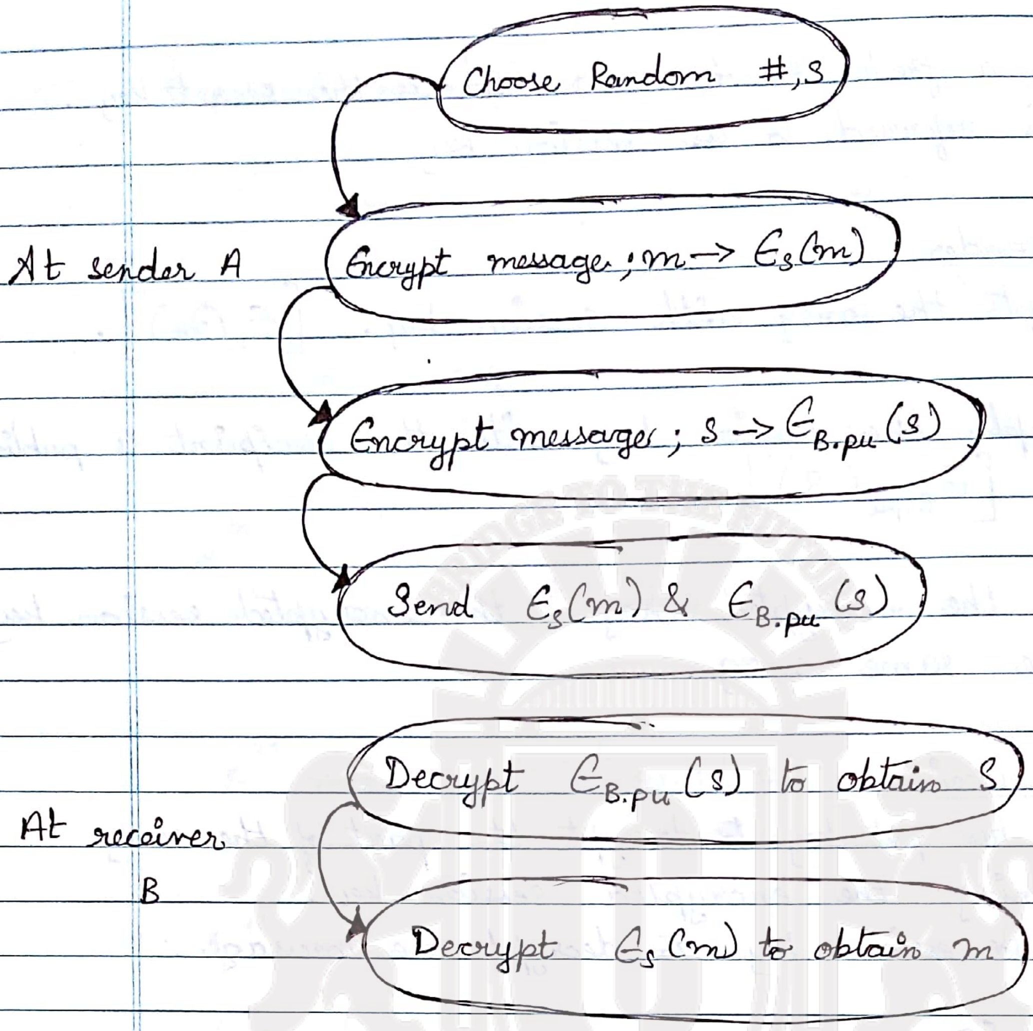
> Uses the session key to decrypt the message.

$$S = [D_{B.pr}(S)]$$

$$m = [D_S(c)]$$

The session key is used to encrypt/decrypt the remaining msg in that session.
The session key is valid for the duration of the session & destroyed thereafter.

Encrypted message with encrypted session key.



Choose Random #, S
At sender A
Encrypt message ; $m \to E_S(m)$
Encrypt message ; $S \to E_{B.pu}(S)$
Send $E_S(m)$ & $E_{B.pu}(S)$
Decrypt $E_{B.pu}(S)$ to obtain S
At receiver B
Decrypt $E_S(m)$ to obtain m

~~35=(1,5,7)~~

Practical issues

i] Generating primes.

Other attacks.

i] Modular Factorisation.

Factoring a no. means representing it as the product of prime no's. A number is said to be factored when all of its prime factors are identified.

As the ~~sig~~ size of the no. increases the difficulty of the factoring ~~s~~ increases rapidly.

Pollard rho algorithm is an algorithm used for factoring no's, other best known factorisation algorithms are
→ Quadratic Sieve → Elliptic Curve → General no. field sieve [GNFS].

Small Exponent attack.

Side - Channel Attack. → time & power

Compute

23/02/18

Compute Inverse (b, c) // compute inverse of c mod b.

```
{
  old1 = 1        new1 = 0
  old2 = 0        new2 = 1
  b' = b          c' = c
  r = 2
  while (r>1)
  { q = b'/c'.
    r = b'%c'.
    t1 = old1 - new1 *q.
    old1 = new1
    new1 = t1
    t2 = old2 - new2 *q.
    old2 = new2.
    new2 = temp2
    b' = c'
    c' = r
    // At this point new1 *b + new2*c = r
  }
  return new2
}
```

Find out the inverse for 12 mod 79.

or Compute gcd (12, 79)

Inverse for 12.

12^{-1} mod 79. [Done before].

$12y \equiv 1 \mod 79$.

Perform gcd on (622, 289).

~~622~~ , ~~289~~

✗. Find gcd of (1070, 1066) using Euclidean algorithm.

1070 1066.

$$\begin{array}{r} 2\,266 \\ 4 \\ \hline 1064 \end{array}$$

$1070 = 1066(1) + 4.$

$1066 = 4(266) + \boxed{2}$

$4 = 2(2) + 0.$

→ An integer n which lies b/w $0 \le n < 210$ satisfies the follg set of congruences.

~~4~~ $n \bmod 5 = 4.$

~~3~~ $n \bmod 6 = 3.$

~~2~~ $n \bmod 7 = 2.$

[CRT]

$n \equiv 4 \bmod 5.$

$n \equiv 3 \bmod 6.$

$n \equiv 2 \bmod 7.$

$M = (5 \times 6 \times 7) = 210.$

$M_1 = \frac{210}{5} = 42.$

$M_2 = \frac{210}{6} = 35$

$M_3 = \frac{210}{7} = 30.$

1] $42y \equiv 1 \bmod 5.$

$2y \equiv 1 \bmod 5.$

M_1^{-1} $\underline{y = 3}.$

2]. $35y \equiv 1 \bmod 6.$

M_2^{-1} $\underline{y = 5}.$

3] $30y \equiv 1 \bmod 7.$

M_3^{-1} $2y \equiv 1 \bmod 7.$

$\underline{y = 4}.$

$n = (4 \times \overset{42}{\cancel{210}} \times 3 + 3 \times \overset{5}{\cancel{35}} \times \overset{5}{\cancel{6}} + 2 \times 30 \times 4) \bmod 210.$

~~$n = (504 + 108 + 240) \bmod 210.$~~

~~$n = 852 \bmod 210.$~~

$n = 1269 \bmod 210.$

$\underline{n = 9}$

$35y \equiv 1 \bmod 6.$

$5y \equiv 1 \bmod 6.$

$\underline{y = 5}.$

Extended Euclidean.

$2 = 1066 - 4 * 266.$

$2 = 1066(1) - (1070 - 1066 * 1) * 266.$

$= 1066 * 267 - 1070 * 266.$

Perform transposition Cipher technique on the plain text "SECURE YOUR NETWORK NOW" by using row major form. (column = 5). By performing row transposition, column transposition & row transposition.

$$\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ \hline S & E & C & U & R \\ E & Y & O & U & R \\ N & E & T & W & O \\ R & K & N & O & W \end{array}$$

Row Transpose. $\begin{bmatrix} S & E & N & R \\ E & Y & E & K \\ C & O & T & N \\ U & U & W & O \\ R & R & O & W \end{bmatrix}$ Colm. Transpose. $\begin{bmatrix} S & E & C & U & R \\ E & Y & O & U & R \\ N & E & T & W & O \\ R & K & N & O & W \end{bmatrix}$

2nd row Transpose. $\begin{bmatrix} S & E & N & R \\ E & Y & E & K \\ C & O & T & N \\ U & U & W & O \\ R & R & O & W \end{bmatrix}$

→ What is the relation b/w RSA encryption & decryption key?

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

→ Find out the value of d if $n = 77$ and $e = 7$.

$n = 7 \times 11 \quad \underline{p = 7} \quad \underline{q = 11}.$

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

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

$p = 7$
$q = 11.$
$\phi(n) = 6 \times 10$
$= 60.$

$7y \equiv 1 \bmod 60.$

60x6
360
350.

$y \equiv 43 \bmod 60.$

$\underline{y = 43}.$

26/02/18 Cryptographic Hash.

A cryptographic hash function $h(x)$ maps a binary string of arbitrary length to a fixed length binary string.

The properties of hash illustrates one way property. Given hash value 'y' it is computationally infeasible to find the input x such that $h(x) = y$.

Weak collision resistance:
Given an i/p value x_1, it is computationally infeasible to find another i/p value x_2 such that $h(x_1) = h(x_2)$. Time Complexity $O(2^n)$.

Strong collision resistance.
It is computationally infeasible to find two input values x_1 & x_2 such that $h(x_1) = h(x_2)$.

Confusion + Diffusion.
If a single bit in the i/p stream is fixed then each bit of the hash value is flipped with probability roughly equal to $\underline{0.5}$.

Challenge

There is a fine difference b/w two collision resistance properties.
In the first, the hash designer chooses x_1 & challenges anyone to find x_2 where the hash values are same $h(x_1) = h(x_2)$.

The attacker tries to find x_1 & x_2 such that $h(x_1) = h(x_2)$
In the 2nd challenge, the attacker has the ability to choose x_1

Side Channel attack in RSA

It is based on montioring of time & power consumption of a cryptographic algorithm on a device.

These attacks are quite successful in leaking sensitive info. such as secret / private keys. especially in the case of embedded device such as smart cards, credit cards, etc...

The attacker induces the card to perform cryptographic tasks involving the stored private key.
It is not possible for the attacker to inspect the contents of register & RAM during smart card operation.
So, there are inexpensive equipments available that enables ~~him~~ the attacker to connect smart card via probes to equipment that can accurately monitor variables such as timing & power consumption.

```
~~For~~ Given d, n, c
~~x=c~~ // want $c^d$ mod n.
        x = c
        for (i = k-2 ; i ≥ 0 ; i--)
          $x = x^2$ mod n
          if ($d_i$ == 1)
            x = x × c mod n
        return (x)
```

SHA-1 [Secured Hash Algorithm]

It is a cryptographic hash function which takes i/p & produces 160 bit o/p.
The hash value known as message digest typically represented in hexadecimal number total 40 digits long.
If a single bit in the msg is flipped, the SHA-1 recomputes 84 bits of 160 bits are flipped for a new hash value.

Attack Complexity.

Weak Collision Resistance.

How long will it take to find input x that hashes to a given value y? [Brute force].

```
do
    { Generate random no. x'
          Compute  h(x')
    }
    while (h(x') != y)
    return (x').
```

Assume that ω is the length of the bits of the string. It follows that the above loop would have to run on the average $2^{\omega-1}$ before finding x'.

Therefore, the brute force attack for one way function property & weak collision resistance takes $O(2^{\omega})$.

Strong Collision Resistance.

Given S is a set of i/p string and hash value pair. [Brute force].

```
notFound = true
while (notFound)
{ generate a random string x'
  search for a pair (x,y) in S where x = x'
  if (no such pair exists in S)
    { compute y' = h(x')
      search for a pair (x,y) in S where y = y'
      if (no such pair exists in x)
          insert (x', y') into S
      else
          notFound = false
    }
return (x and x')
```

Birthday Analogy.

What is the minimum no. of persons require so that the probability of two / more in the group having the same birthday is greater than 50%.

23 persons. $\frac{364}{365} \times \frac{363}{365} \times \frac{362}{365} \times \ldots$

It is known that in a class of 23 individuals there is greater than 50% chance the birthday of atleast two persons coincide. This is birthday paradox.

The random string generated for strong collision resistance is analogous to the random individuals in the birthday paradox.

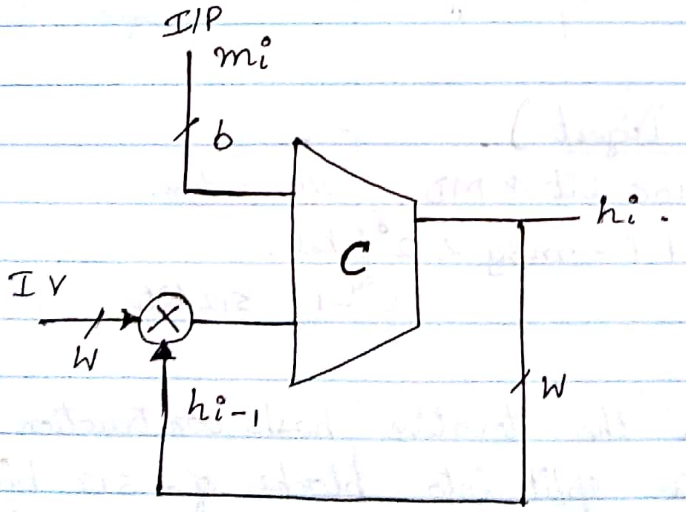
The birthday of randomly chosen individual is analogous to the hash value of randomly chosen string.

Construction of Cryptographic hash

Generic of cryptographic hash.

C is a compression ~~comporsion~~ function, IV represents intialization vector, m_i = i^{th} block of message m, h_i=hash value after i^{th} iteration.

Iterative construction of cryptographic hash.



This was introduced by "Merkel & Damgard."

The i/p to a cryptographic function is a message or document to accomodate i/p's of arbitrary length Hash functions uses iterative construction as shown in the figure.

C

Normally MD5 ~~or~~ & SHA1,

C is a ~~comparison~~ compression box which accepts two binary strings of length of b and ω and produces the output of length ω, where b = block size of the i/p & ω = is the width of the hash digest.

The diagram performs operations and produces operations like $h_1 = C(IV, m_1)$
$h_i = C(IV, m_i)$

During first iteration the multiplexer at the second i/p accepts a predefined IV & the top i/p is the first block of the message.

Subsequently for all iterations the partial hash output is fed back as the second i/p to the C box and the top i/p is derived from the successive blocks of message. This is repeated until the complete blocks ~~in~~ of the message is processed.

MD (Message Digest).

SHA-1. → 160 bit o/p MD. I/P random.

$PT = msg < 2^{64}$ bit.

$(2^{64}-1)$ 512 bits

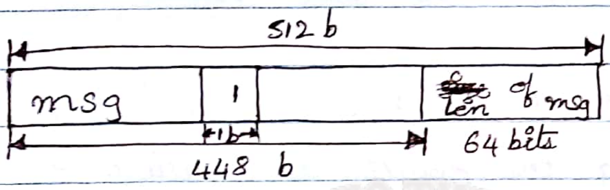
→ SHA-1 uses the iterative hash construction.

→ The msg is split into blocks of 512 bits.

→ Plain txt or msg should be less than 2^{64} bits.

→ The length of the msg is expressed in binary as ~~t~~ a 64-bit number and is appended to the msg.

→ B/W the msg & the length field, a pad is inserted so that the length of the block is a multiple of 512 block size. i.e (msg + pad + length).



Padding is a process of adjusting the message ~~so~~ so its length is (448 mod 512).

Padding bit '1' followed by remaining zeroes.

Description of SHA-1 Algorithm.

Initialize an array such that each block is split into 16 words each of 32 bits.

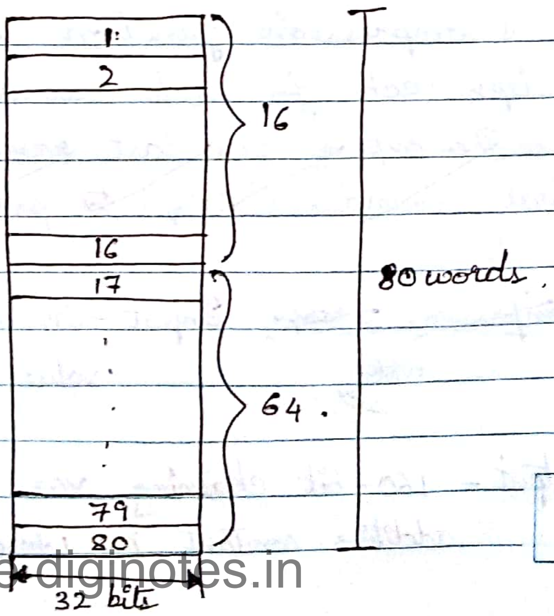
512 / 32 = 16.

These 16 words populate the first 16 positions of an array of 80 words.

The remaining 64 words are obtained from

$\boxed{W_i = W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}}$ where $16 < i \leq 80$.

~~Hash Comp~~



Padding.

Step-1 : append padding bits.

Padding : Given an m-bit message, a single bit '1' is appended as the $m+1^{th}$ bit and then

$(448-(m+1)) \bmod 512$ (b/w 0 & 511) zero bits are appended, making the result as multiple of 512 bits long. (length $\equiv 448 \bmod 512$) The padding pattern is 100.....00.

Step-2 : append length.
A 64-bit length in bits of the original message is appended.

Step-3 : Initialize MD buffer.
A 160-bit buffer is used to hold intermediate and final results of the hash function.
The buffer is represented as (5) 32-bit registers (A, B, C, D, E) initialized to the ~~follg~~ initial integers (hex values).
The values are stored in big-endian order, i.e, the most significant byte of a word in the low address byte position.

Step-4 : process msg in 512 bit (16 word) blocks.

A compression function with 4 rounds of processing of 20 steps each for each round operation.
~~The O/P of the last round is added to the input of the first round.~~ (CV_q) to produce (CV_{q+1}).

~~Compression~~ ~~theory~~ ~~[illegible]~~ so: Input - 512 bit block Y_q, 160-bit buffer value CV_q represented by ABCDE.

Output - 160-bit chaining var. CV_q+1 makes use of additive constant K_t where $0 \leq t \leq 79$.

~~Step.~~ ~~Name~~ ~~Function.~~

1) The o/p of the last round is added to the i/p of the
2) first round.
3)

SHA-1 compression function.

Each round consists of 16 steps operating on the buffer ABCDE with each step of ~~the~~ the form:

$$[(E + f(t,B,C,D) + (A<<5) + W_t + K_t), A, (B<<30), C, D)]$$
$$\downarrow A \quad \downarrow B \quad \downarrow C \quad \downarrow D \quad \downarrow E$$

~~The f 16 words of current block the o/p arising~~

~~8 The overall~~

where,

A, B, C, D, E = the 5 words of the buffer

t = step no, $0 \leq t \leq 79$.

$f(t, B, C, D)$ = primitive logical function for step t.

W_t = a 32-bit word derived from the 512-bit i/p block.

K_t = an additive constant, 4 distinct values are used.

$+$ = addition modulo 2^{32}.

Primitive functions $f(t,B,C,D)$:

Input is 3 32-bit words.

Output is 1 32-bit word.

Each function performs a set of bitwise logical operations as shown below.

Step	Function name	Function value.
$(0 \leq t \leq 79)$	$f_1 = f(t,B,C,D)$	$(B \wedge C) \vee (\overline{B} \wedge D)$
$(20 \leq t \leq 39)$	$f_2 = f(t,B,C,D)$	$B \oplus C \oplus D$
$(40 \leq t \leq 59)$	$f_3 = f(t,B,C,D)$	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
$(60 \leq t \leq 79)$	$f_4 = f(t,B,C,D)$	$B \oplus C \oplus D$

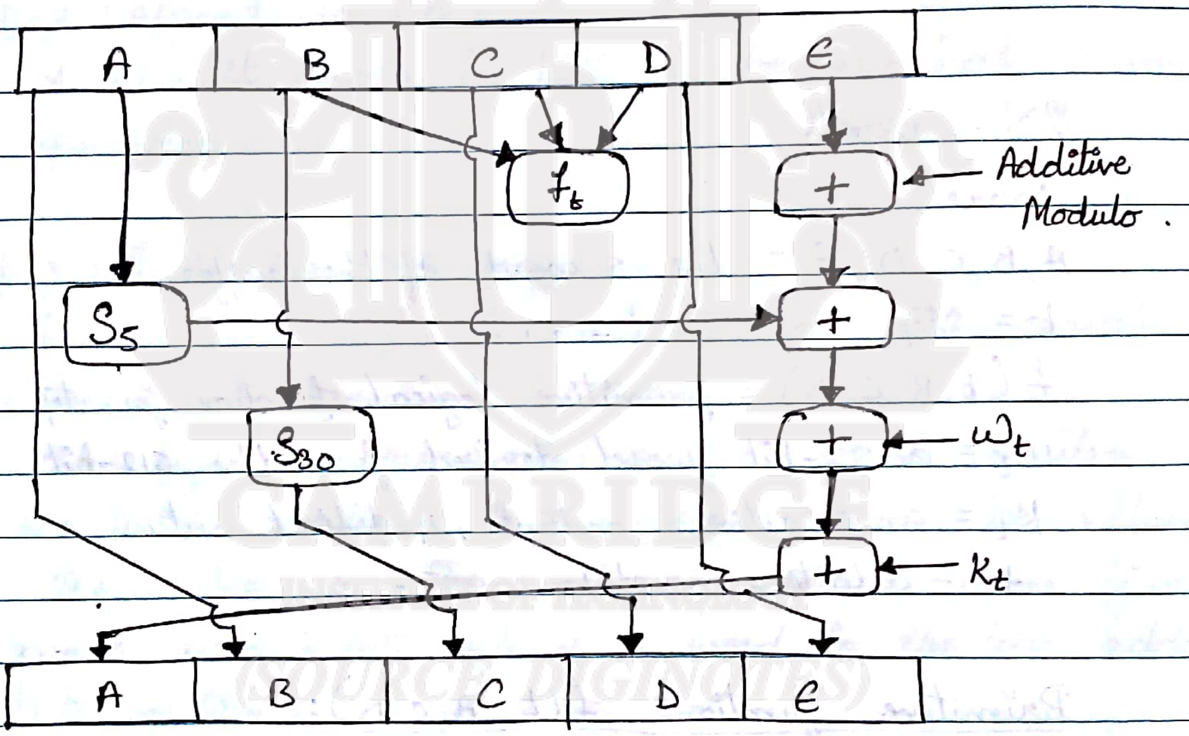
~~(i)~~ ~~[illegible]~~

Derivation of the 32-bit word ω_t from the 512-bit i/p block.

The 1st 16-values of ω_t are taken directly from 16 words of the current block. The remaining values are defined as follows:

$$\omega_t = \omega_{t-16} \oplus \omega_{t-14} \oplus \omega_{t-8} \oplus \omega_{t-3}$$

Overall operation of SHA-1



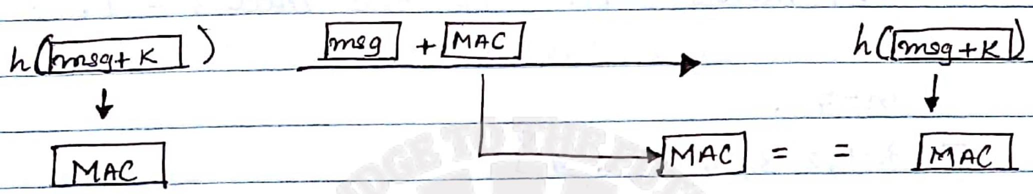
The final value is obtained by adding the initial value with the final value.
(Addition means mod 2^{32}).

2^{32} → because a 32 bit register can store 2^{32} values.

Applications of Hash.

1] MAC. [Message Authentication Code].

Plain text.



MAC is used to provide message integrity as well as message authentication.

The cryptographic hash applied on a message creates a digest or digital fingerprint of that message.

The sender & the receiver share a common secret key 'K'.

The message and the key are concatenated and perform hash operation on that string.

This hash value is the fingerprint of that message (M) and key (K).

That is $MAC = h(M \| K)$

MAC is just a checksum of the message computed and sent to the receiver with the message.

The receiver receives message + MAC, the receiver again computes the MAC with the secret key.

The received MAC and the generated MAC are compared, if its not equal the result is mismatched, he assumes the message is changed.

If the result is matched, the sender of the message is correct i.e source authentication and the message has not been corrupted or tampered within transit which provides message integrity.

$Z = \{1, 3, 5, 7\}$.

$3^1 = 3 \bmod 8 = 3$. $5^1 = 5 \bmod 8 = 5$.

$3^2 = 9 \bmod 8 = 1$. $5^2 = 25 \bmod 8 = 1$.

$3^3 = 27 \bmod 8 = 3$. $5^3 = 125 \bmod 8 = 5$.

$3^4 = 81 \bmod 8 = 1$. $5^4 = 5^4 \bmod 8 = 1$.

$n = 7$.

$Z = \{1, 2, 3, 4, 5, 6, \cancel{7}\}$.

$2 = 2 \bmod 7 = 2$. $3 = 3 \bmod 7 = 3$.

$2^2 = 4 \bmod 7 = 4$. $3^2 = 9 \bmod 7 = 2$.

$2^3 = 8 \bmod 7 = 1$. $3^3 = 27 \bmod 7 = 6$.

$2^4 = 16 \bmod 7 = 2$. $3^4 = 81 \bmod 7 = 4$.

$2^5 = 32 \bmod 7 = 4$. $3^5 = 243 \bmod 7 = 5$.

$2^6 = 64 \bmod 7 = 1$. $3^6 = 729 \bmod 7 = 1$.

$3^7 = 2187 \bmod 7 = 3$.

$\therefore$ 3 is a generator as it contains all elements.

$4 = 4 \bmod 7 = 4$.

$4^2 = 16 \bmod 7 = 2$.

$4^3 = 64 \bmod 7 = 1$.

$4^4 = 256 \bmod 7 = 4$.

$4^5 = 1024 \bmod 7 = 2$.

Diffie Hellman Key Exchange. (DHKE)

DHKE algorithm was invented by Diffie and Hellman in 1976 used to exchange info. b/w two parties shared ~~[illegible]~~ with a secret of a particular time duration, which is a private key and a corresponding public key concept. It is symmetric key.

1] Choose two numbers i.e 'p' and 'g' where 'p' is a prime number and 'g' is a generator of that prime number.

2] It is also known that 'g' acts as base value. 'p' acts as modulus.

Sender Side Key generation (A)

Sender generates ~~a~~ or chooses a random integer a such that a lies b/w $1 < a < p-1$ and computes a partial key.

$$K_A = g^a \bmod p.$$

Receiver Side (B)

Receiver chooses a random integer b such that b lies b/w $1 < b < p-1$ and computes a partial key.

$$K_B = g^b \bmod p.$$

→ A sends the computed partial key ~~to~~ ~~B~~ K_A to B and B sends the computed partial key ~~to~~ K_B to A.

→ On receiving the partial keys, A computes $(K_B)^a \bmod p$. and B computes $(K_A)^b \bmod p$.

→ These both will generate a equal value.

Let $p = 131$ and $g = 2$.
choose random number $a = 24$ $b = 17$.
find K_A & K_B.

$(K_A)^b \bmod p$ $\qquad\qquad$ $(K_B)^a \bmod p.$

$(g^a \bmod p)^b \bmod p.$ $\qquad\qquad$ $(g^b \bmod p)^a \bmod p.$

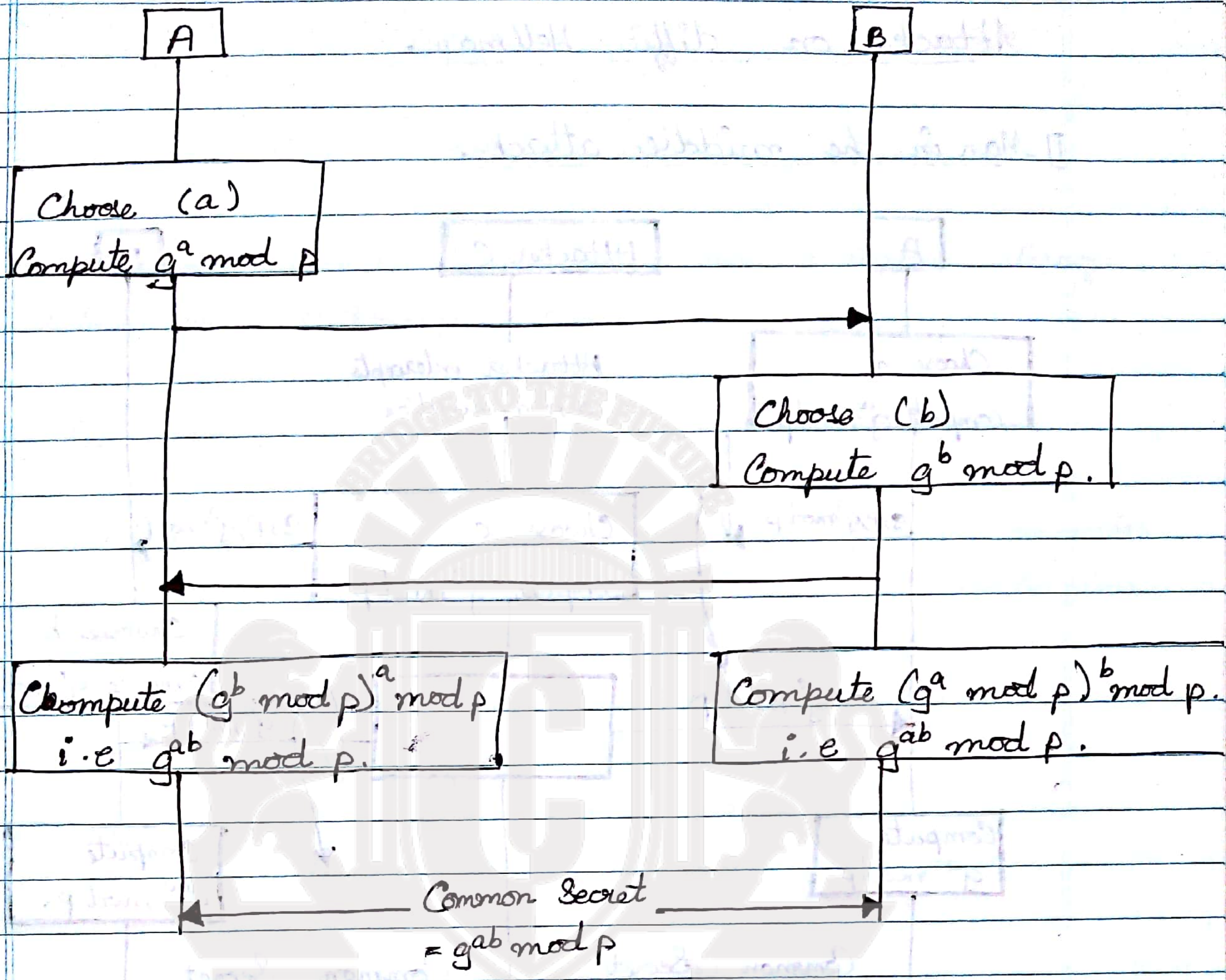
$g^{ab} \bmod p$ $\qquad\qquad$ $g^{ba} \bmod p.$

$g^a \bmod p \Rightarrow 2^{24} \bmod 131$ $\qquad\qquad$ $g^b \bmod p \Rightarrow 2^{17} \bmod 131.$

$= 46$ $\qquad\qquad$ $= 72.$

$(K_A)^b \bmod p$

$= (46)^{17} \bmod 131.$



$p = 11$ $\qquad g^a \bmod p = 5.$ $\qquad y = g^a \bmod p.$

$g = 7$ $\qquad g^b \bmod p = 3.$ $\qquad a = \log_g y \bmod p.$

1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

$x = \log_7$

$7^1 \bmod 11 = 7.$ $\qquad a = \cancel{3}\ 2.$

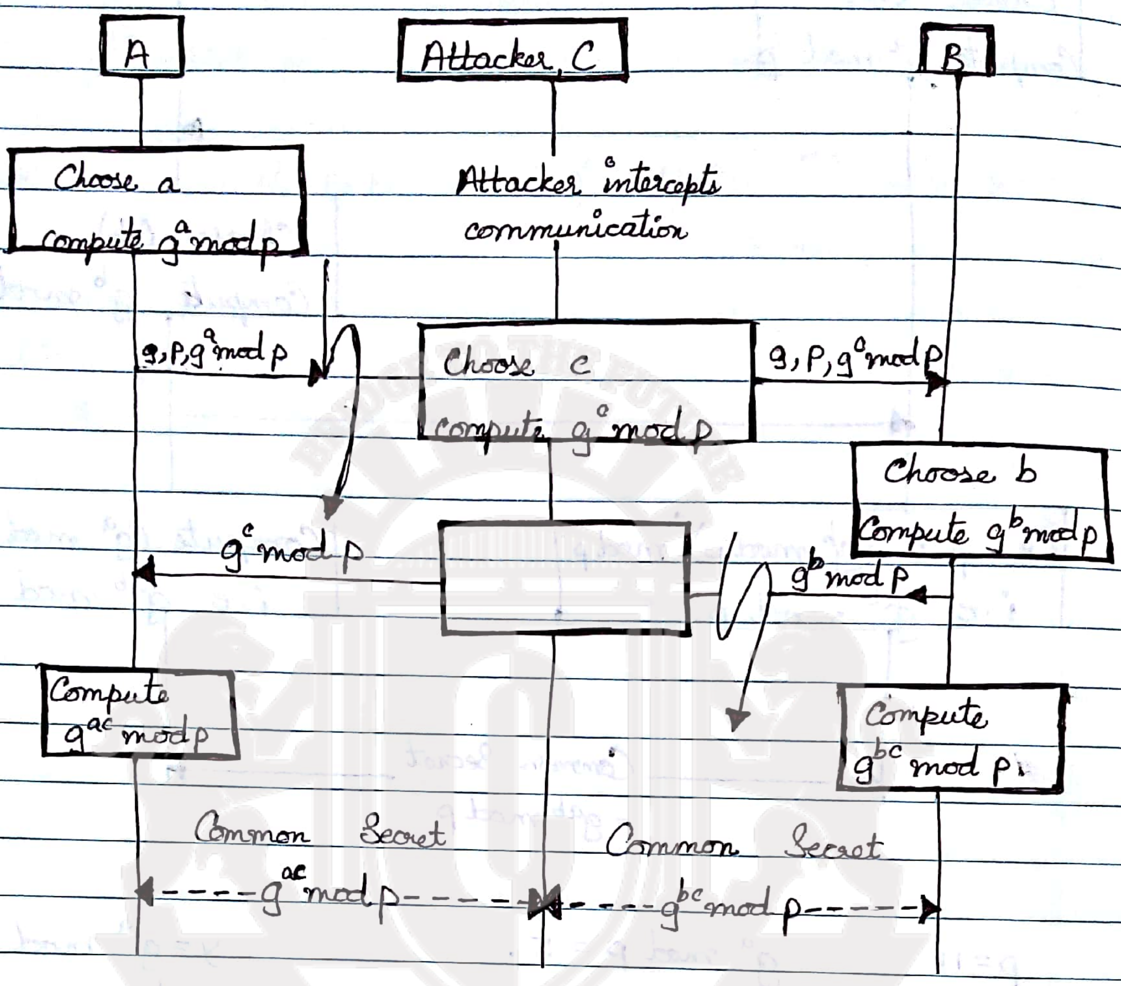
$7^2 \bmod 11 = 5.$ $\qquad b = \cancel{3}\ 4.$

$7^3 \bmod 11 = 2.$

$7^4 \bmod 11 = 3.$

Attacks on diffie Hellman.

1] Man in the middle attack.



* Sender 'A' chooses a random integer 'a' and computes g^a mod p and sends it to the receiver B.
* Attacker 'C' intercepts the communication and chooses a random integer c and computes g^c mod p and sends it to B.
* ~~B~~ Receiver B unaware of C receives g^c mod p and chooses a random integer b then computes g^b mod p and sends it to ~~B~~ A.
* Again Attacker C intercepts the communication chooses an random integer 'c' then computes g^c mod p and sends it to A.
* ~~&~~ A then ~~cor~~ receives g^c mod p then computes g^{ac} mod p
* B also computes g^{bc} mod p.
* Both the keys are secret and are shared between A and B.