Public Key Algorithms

DITISS- 2014

Overview

- Public key algorithms
  - Diffie-Hellman Key Exchange, RSA, Eliptic curve
- Hashing
  - MD5, SHA
- Digital Signature Algorithms
  - DSA

Diffie-Hellman Key Exchange

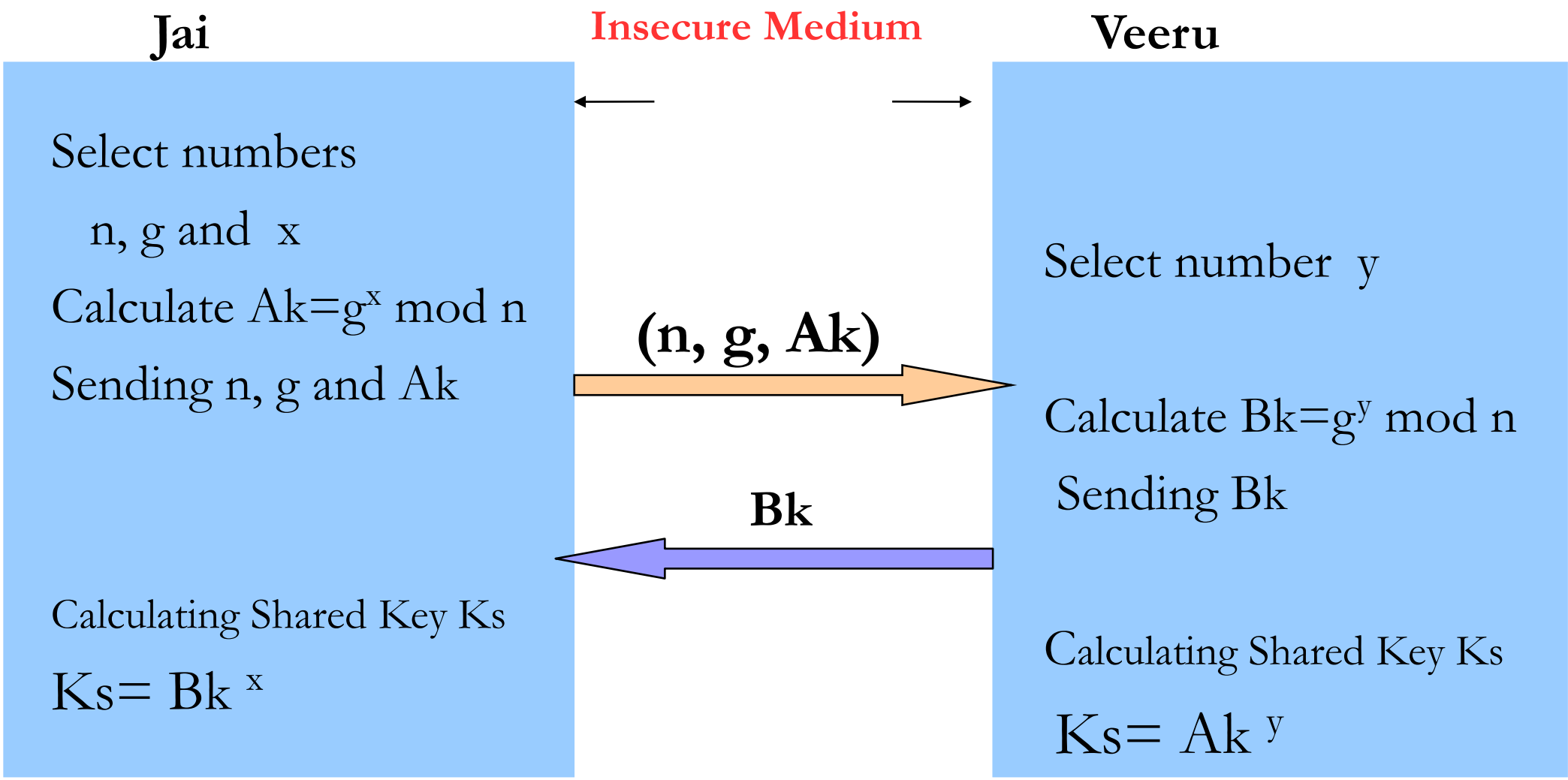
Modular Arithmetic

- If current time is 9 o'clock , after 4 hours the clock will show...
  - It is not at 13 o'clock , but it is 1o'clock.
  - When we reach 12, we start over; 12 is called the modulus.
- Example
  - 10 mod 6=4 5 mod 9 = 5
  - 32 mod 8 = 0

Diffie-Hellman Key Exchange

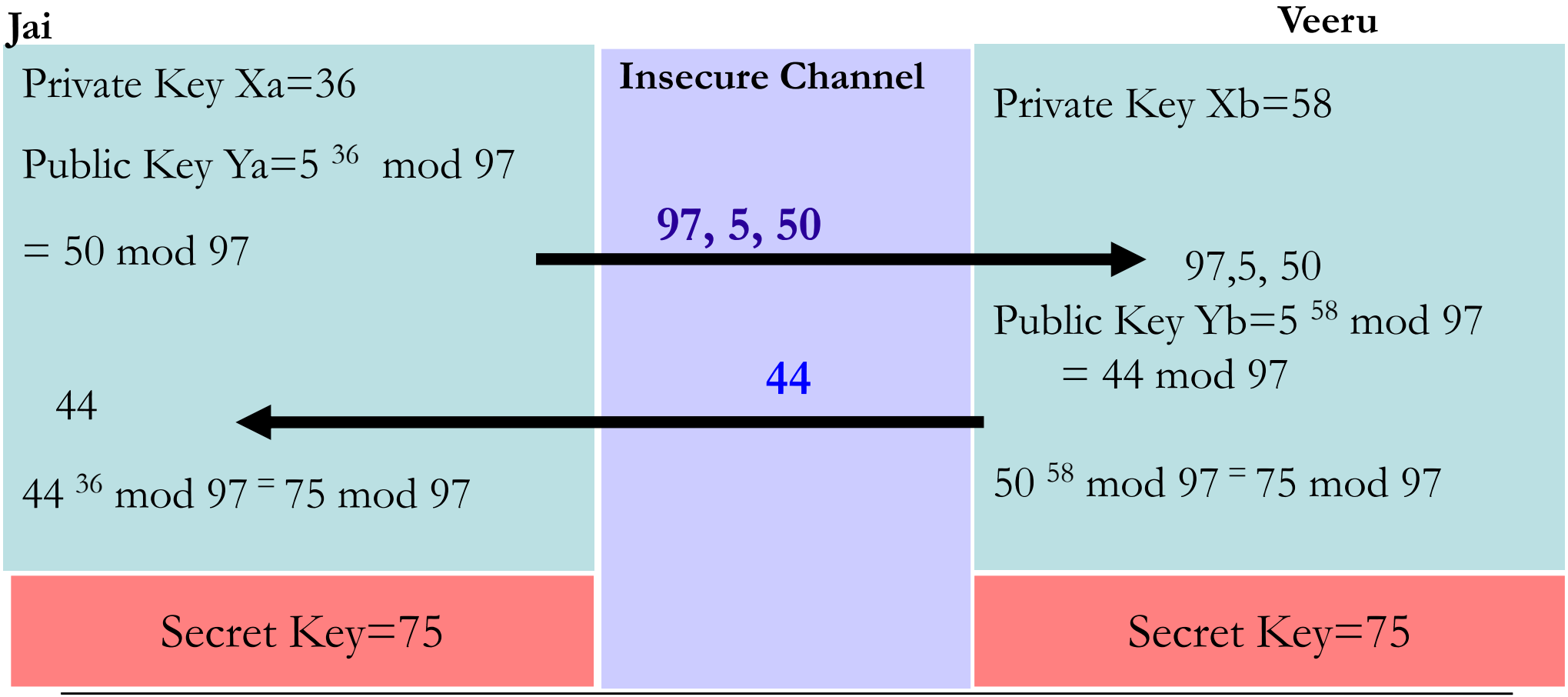
- Public key algorithm for key exchange
- Allows two users to exchange a secret key over an insecure medium without any prior secrets.
  - Functionality limited to key exchange only
- The scheme was first published by Whitfield Diffie and Martin Hellman in 1976.

Diffie-Hellman Key Exchange

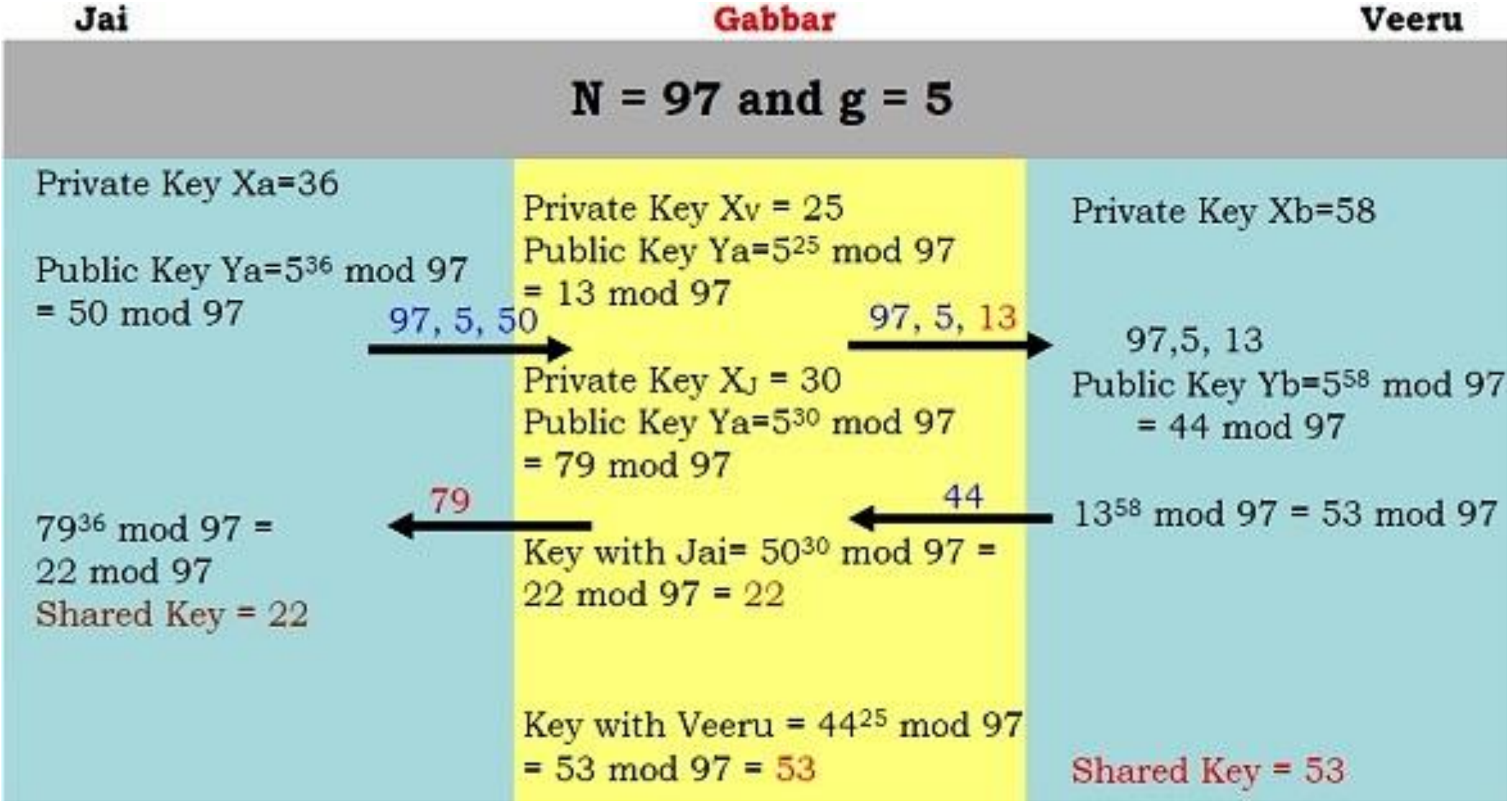


Diffie-Hellman Key Exchange

n=97 g=5



MITM on DHKE


Jai
Gabbar
Veeru
N = 97 and g = 5
Private Key Xa=36
Public Key Ya=5^36 mod 97
= 50 mod 97
Private Key Xv = 25
Public Key Ya=5^25 mod 97
= 13 mod 97
Private Key Xb=58
97, 5, 50
97, 5, 13
97,5, 13
Public Key Yb=5^58 mod 97
= 44 mod 97
Private Key XJ = 30
Public Key Ya=5^30 mod 97
= 79 mod 97
79
44
79^36 mod 97 =
22 mod 97
Shared Key = 22
Key with Jai= 50^30 mod 97 =
22 mod 97 = 22
13^58 mod 97 = 53 mod 97
Key with Veeru = 44^25 mod 97
= 53 mod 97 = 53
Shared Key = 53

Limitation of Diffie- Hellman Algorithm

- ☹Cannot be used for Encryption/Decryption and Digital Signature
- ☹Does not provide authentication to the communication parties.
  - ☹ Vulnerable to man-in the middle attack

RSA

RSA Algorithm

- RSA
  - Developed by Ronald Rivest, Adi Shamir and Leonard Adleman
- Functionality
  - Encryption
  - Decryption
  - Digital signatures
- Based on number theory

RSA Algorithm

- Key Generation Steps
  - Choose two large prime numbers p and q
  - Compute **n** and **z** such that **n=p*q** and **z=(p-1)*(q-1)**
  - Choose a number **d** relatively prime to **z**
  - Compute e such that (e*d)= 1 mod z

Public Key (n,e)

Private Key (n,d)

RSA Algorithm

- Encryption
  - Message m, cipher c
  - $C = m^e \bmod n$
  - where (n,e)receivers public key
- Decryption
  - $m = c^d \bmod n$

  where (n,d) receivers private key

RSA – Key Generation Example

- Select primes p=11, q=3.
- n = p*q = 11*3 = 33
- z = (p-1)(q-1) = 10*2 = 20
- Choose d=7
- Compute ‘e’ such that 7*e = 1 mod 20

  ∴ e=3
- Public key = (n, e) = (33, 3)
- Private key = (n, d) = (33, 7).

RSA – Encryption and Decryption

- Encrypt the message m = 7

  $c = m^e \bmod n = 7^3 \bmod 33 = 343 \bmod 33$

  cipher text c = 13

- Decryption

  $m' = c^d \bmod n = 13^7 \bmod 33 = 7$

Example

- An example of the RSA algorithm.

Plaintext (P)			Ciphertext (C)		After decryption	
Symbolic	Numeric	P^3	P^3 (mod 33)	C^7	C^7 (mod 33)	Symbolic
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E

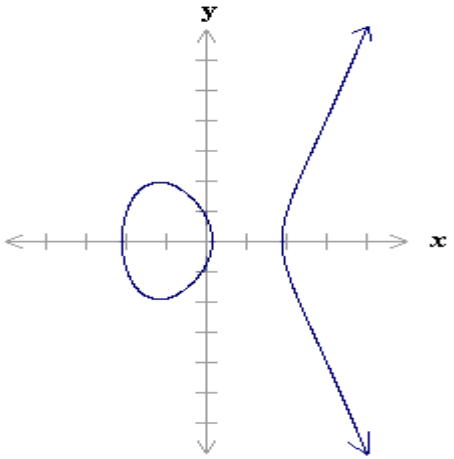
Sender's computation

Receiver's computation

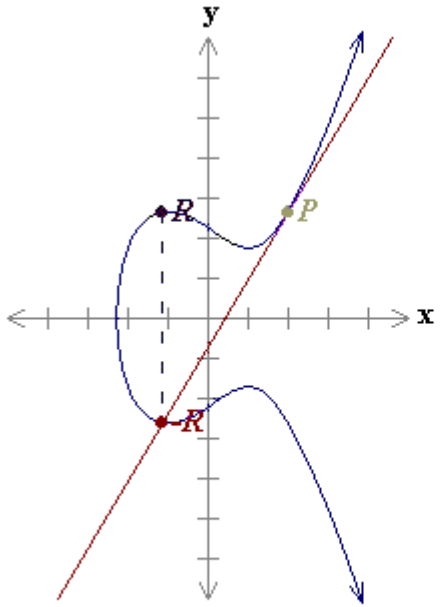
Elliptic Curve

- An elliptic curve over real numbers may be defined as the set of points(x,y) which satisfy an elliptic curve equation $y^2=x^3+ax+b$ where x, y, a and b are real numbers
- Each choice of the numbers a and b yields different elliptic curve
- An elliptic curve group over real numbers consists of the points on the corresponding elliptic curve, together with a special point 'o' called point at infinity

Elliptic Curve



$y^2 = x^3 - 4x + 0.67$



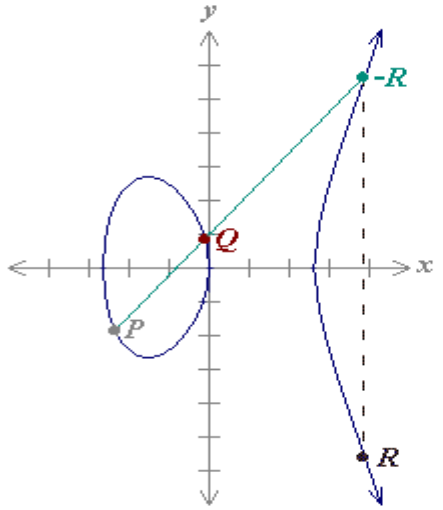
P (2, 2.65)

$-R$ (-1.11, -2.64)

R (-1.11, 2.64)

$2P = R = (-1.11, 2.64)$.

$y^2 = x^3 - 3x + 5$



P (-2.35, -1.86)

Q (-0.1, 0.836)

$-R$ (3.89, 5.62)

R (3.89, -5.62)

$P + Q = R = (3.89, -5.62)$.

$y^2 = x^3 - 7x$

Elliptic Curve Group

- If $x^3 + ax + b$ contains no repeated factors, or equivalently if $4a^3 + 27b^2$ is not 0, then the elliptic curve can be used to form a group.
- A group is simply a set of points on the curve.
- Calculations over the real numbers are slow and inaccurate due to round-off error.
- Elliptic curve groups over the finite fields of Fp is used in practice.
- Field Fp uses the numbers from 0 to p - 1, and the computation end by taking the remainder on division by p

Elliptic Curve Discrete Logarithm

- The trick with elliptic curve cryptography is that if you have a point F on the curve, all multiples of this point are also on the curve.
- Elliptic Curve discrete logarithm problem.
  - Given two points G and Y on an elliptic curve such that Y=k*G. Find 'k'

ECC – Key Generation

- Select a prime number p ≈ 2^{180}
- Define Elliptic group of points $E_p(a,b)$
  - Select elliptic curve parameters 'a' and 'b' for $y^2 = x^3 + ax + b \pmod p$
- Select a generator point G=(x,y) in $E_p(a,b)$
  - The smallest value n for which nG = O be very large prime number.
- $E_p(a,b)$ and G must known to all participants

ECC – Key Generation

- Each user has to generate a public key and private key
- Private key
  - User has to select an integer n_p which is < n
  - n_p is the private key
- Public Key
  - $P_u = n_p * G$ Public Key is a point in E_p (a,b)
  - P_u is the public key

ECC – Key Generation

- Example
  - p = 211 .
  - Elliptic Curve $y^2=x^3-4$ ie $E_p(0,-4)$
  - Generator Point G=(2,2) ie 241 G =O n=241
  - A's private key $n_A=121$
  - A's public Key P_A
    - P_A =121(2,2) = (151,48)
  - B's private key n_B= 203
  - B's Public Key P_B
    - P_B = 203 (2,2) = (130,203)

ECC – Encryption

- Select common parameters
  - Elliptic group Ep (a, b)
  - Generator point G
- Sender A's key
  - Private key nA
  - Public key PA = nA* G
- Receiver B's Key
  - Private key nB
  - Public key PB = nB* G

ECC – Encryption

- To encrypt and send message m, Sender choose a random positive integer k
- Encryption
  - Cipher text C= {kG, (m + kPB) }
    - where PB is receivers public key
- Decryption
  - (m + kPB) – nB (kG)
  - m + k* (nB*G) - nB*(kG) = m
    - where nB is receivers private key

Advantages of ECC

- Using shorter key size can be used to achieve the same security of RSA
- Less storage, memory, power, bandwidth requirement
- Suitable to implement in constraint platform like smart cards, hand-held devices etc..

ECC Vs RSA

ECC Key Size (Bits)	RSA Key size (Bits)	Key Size Ratio
163	1024	1:6
256	3072	1:12
384	7680	1:20
512	15360	1:30

Reference : NIST Recommendation for Key Management – Part 1: General (Revised)

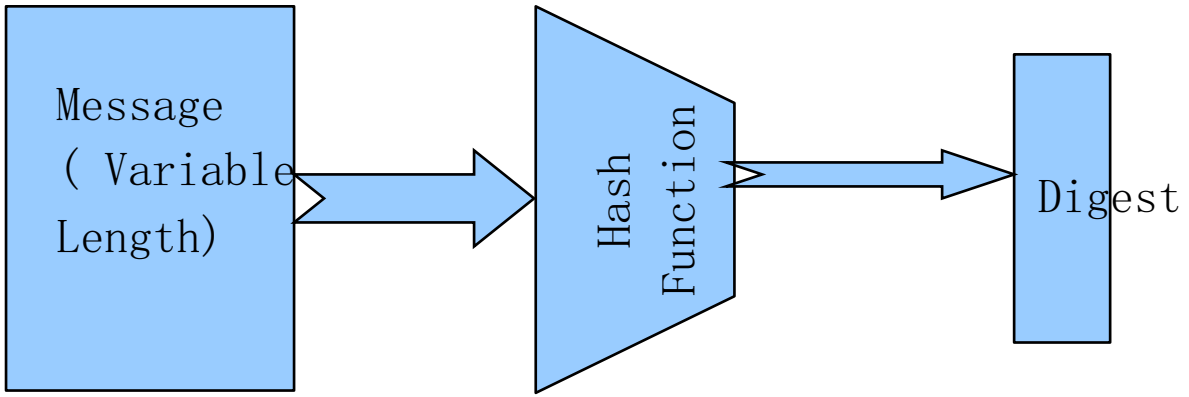
Hash Functions

Hash Function

- A hash function is an algorithm
  - which creates a digital representation or "fingerprint" in the form of a "hash value" or "hash result" of a standard length
  - Hash value is usually much smaller than the message but nevertheless substantially unique to it.
  - Any change to the message invariably produces a different hash result when the same hash function is used
  - It is a one way function – Easy to calculate hash value from message but difficult to generate message from hash value

29

Hash Function



Message Digest Algorithm (MD5)

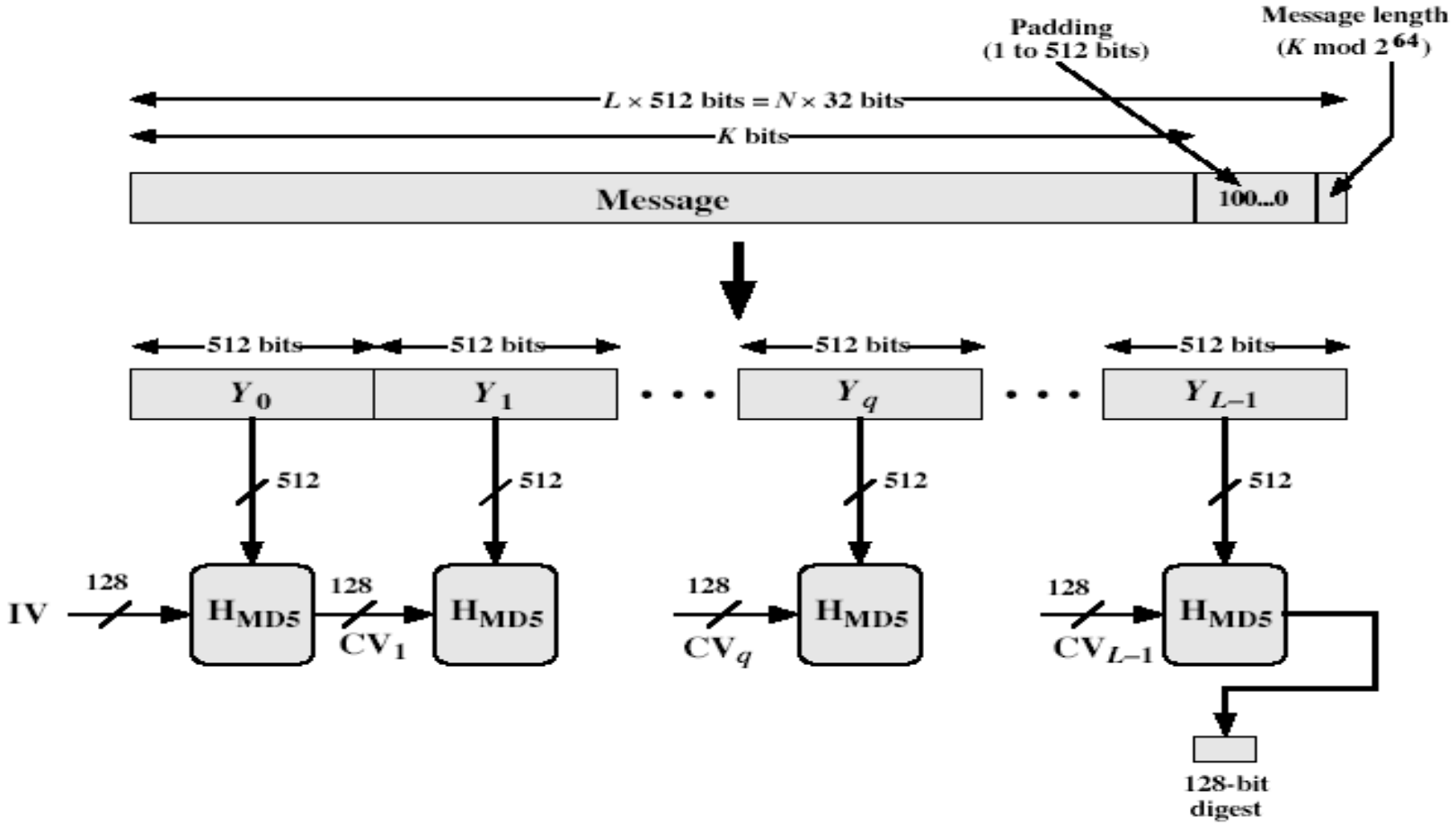
- Message Digest Hashing Algorithm
- Developed by Ronald Rivest
- Processing input text in 512 bit block
- Output size 128 bit
- Specified as Internet standard RFC1321

MD5 Algorithm

Message Digest Algorithm - Steps

1. Pad message so its length is 448 mod 512
2. Append a 64-bit length value to message
3. Initialise 4-word (128-bit) MD buffer (A,B,C,D)
4. process message in 16-word (512-bit) blocks:
   - Compression function consists of 4 rounds of 16 steps on message block & buffer
   - add output to buffer input to form new buffer value
5. output hash value is the final buffer value

MD5 Algorithm



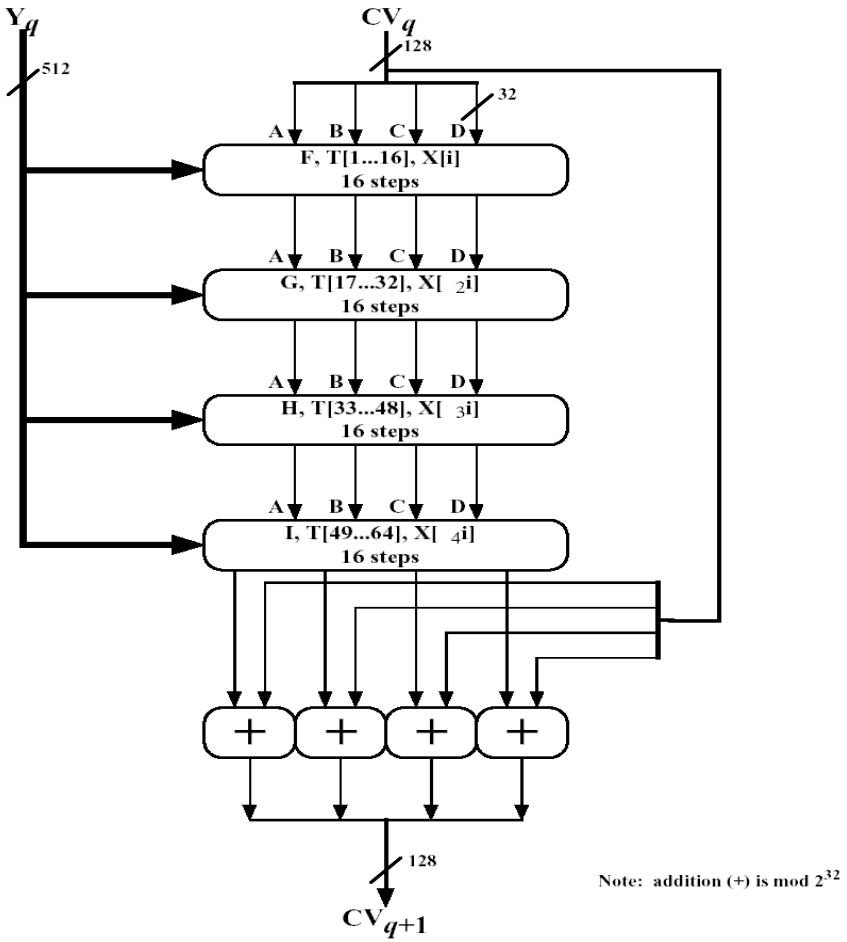
Message Digest Generation using MD5

Reference: Cryptography and Network Security -Stallings

MD5 compression

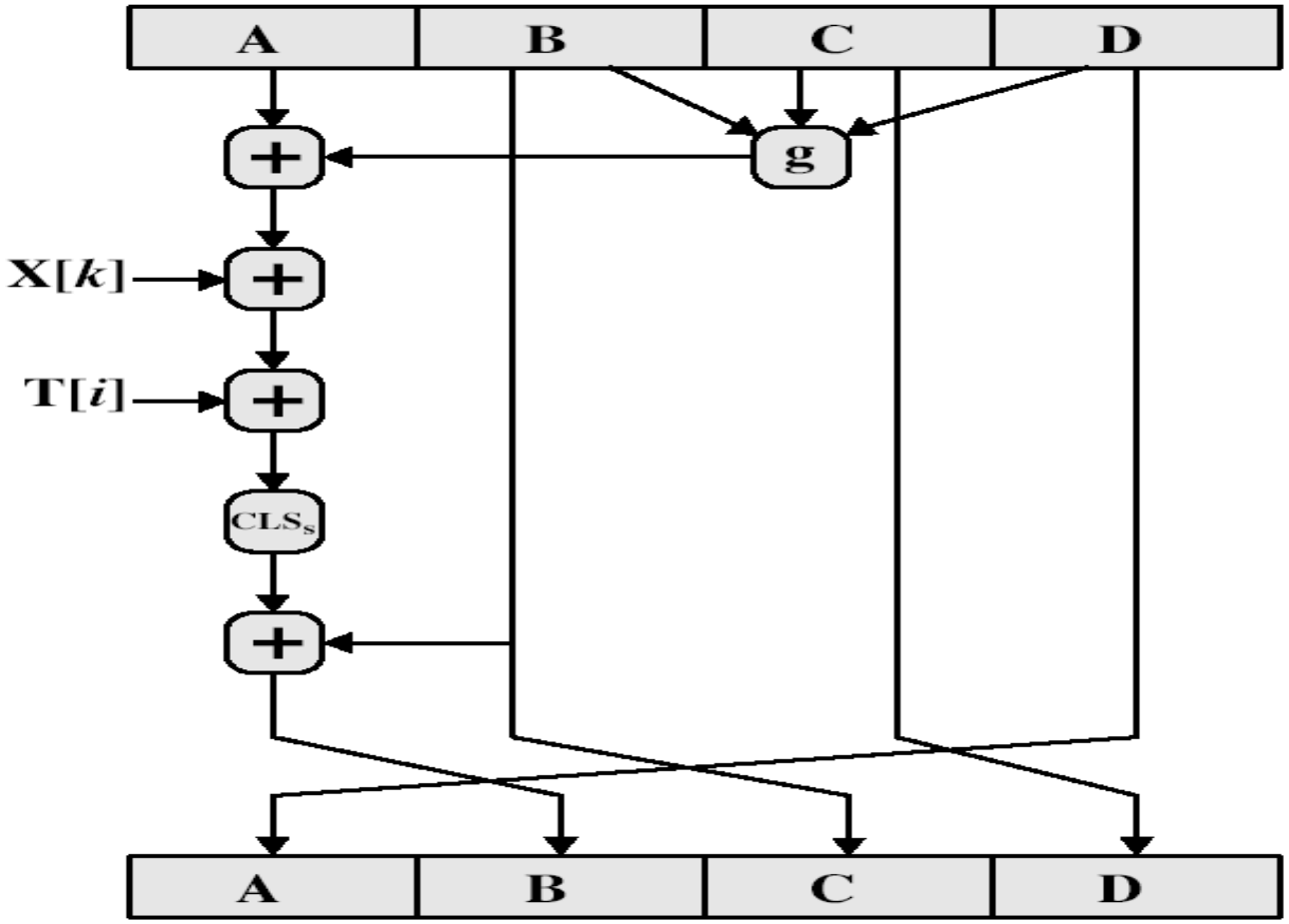
- Each round has 16 steps of the form:
  - a = b+((a+g(b,c,d)+X[k]+T[i])<<<s)
- a,b,c,d refer to the 4 words of the buffer, but used in varying permutations
  - After 16 steps each word is updated 4 times
- where g(b,c,d) is a different nonlinear function in each round (F,G,H,I)
- X[k]=M[q*16 + k] =k th 32 bit word in q th 512 bit block of message
- T[i] is a constant value derived from sine function

MD5 Single block Processing



Reference: Cryptography and Network Security -Stallings

Elementary MD5 operation



Reference: Cryptography and Network Security -Stallings

Secure Hash Algorithm (SHA)

- SHA was designed by NIST & NSA in 1993, revised 1995 as SHA-1
- US standard for use with DSA signature scheme
  - standard is FIPS 180-1 1995, also Internet RFC3174
  - the algorithm is SHA, the standard is SHS
- produces 160-bit hash values
- now the generally preferred hash algorithm

SHA - steps

1. Pad message so its length is 448 mod 512
2. Append a 64-bit length value to message
3. Initialize 5-word (160-bit) buffer (A,B,C,D,E) to
   (67452301,efcdab89,98badcfe,10325476,c3d2e1f0)
4. Process message in 16-word (512-bit) chunks:
   - expand 16 words into 80 words by mixing & shifting
   - use 4 rounds of 20 bit operations on message block & buffer
   - add output to input to form new buffer value
5. Output hash value is the final buffer value

SHA Vs MD5

- SHA
  - Brute force attack is harder (160 Vs 128 bits for MD5)
  - Not vulnerable to any known cryptanalytic attacks (compared to MD4/5)
  - A little slower than MD5 (80 Vs 64 steps)
- Both work well on a 32-bit architecture
- Both designed as simple and compact for implementation

Digital Signature Standards (DSS)

- DSS is a standard for Digital Signature
- The DSS defines a public key cryptographic system for generating and verifying digital signatures
- Digital signature is represented as a string of bits
- Digital signature(DS) is computed using set of rules and set of parameters such that the identity of the signatory and integrity of the data can be verified
- DS can be generated for stored data and programs . So integrity of the data can be verified in later time

Digital Signatures (DS)

- DS contains two process
  - Signature Creation
  - Signature Verification
- Signature Creation
  - Performed by sender
  - Create the hash result (digest) of the message
  - Transform hash result into digital signature using senders private key

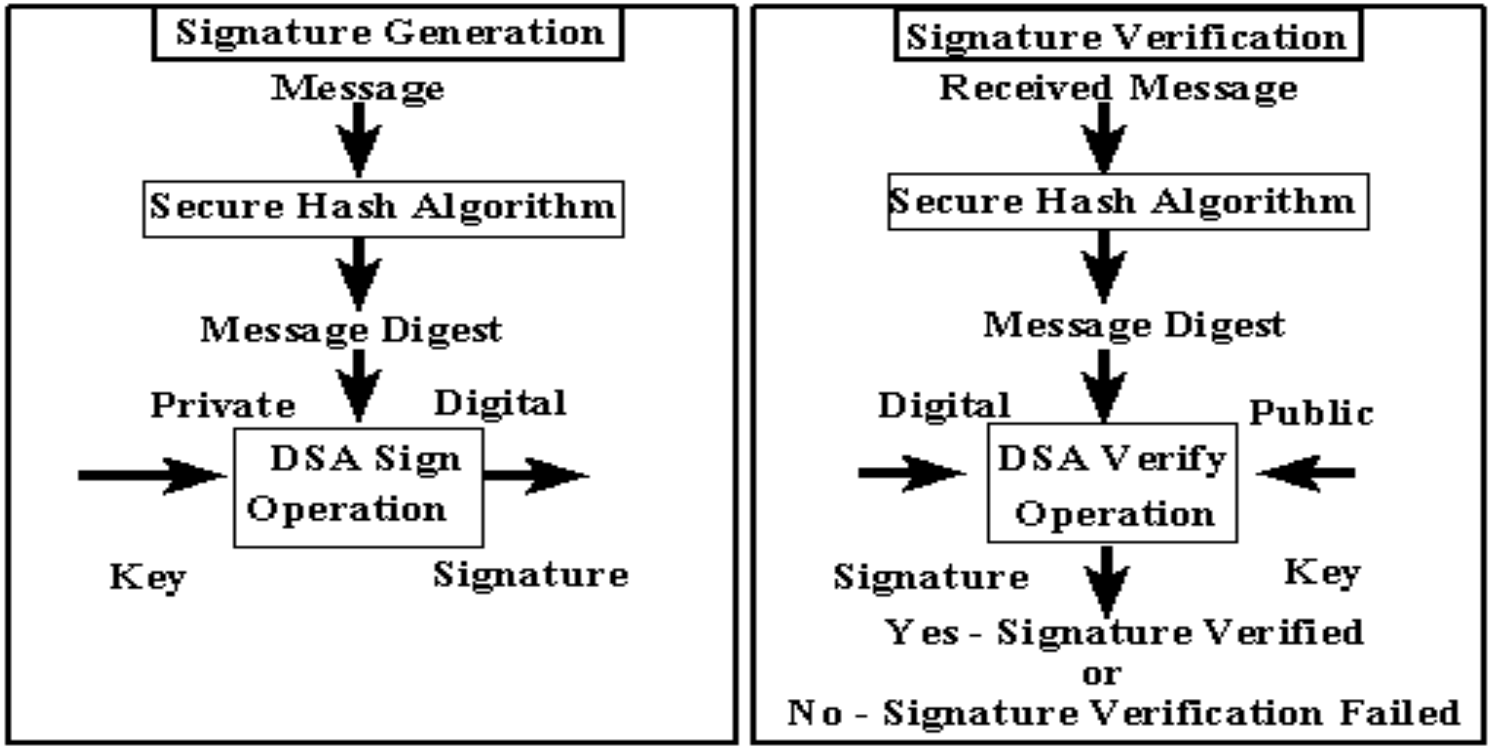
Digital Signatures

- Signature Verification
  - Performed by receiver
  - Verify the signature using senders public key
  - Calculate the hash of received message using same hash function used by the sender

Digital Signature Algorithm

- Digital Signature Algorithm is defined in Digital Signature Standards (DSS)
- DSA provides the capability to generate and verify signatures
- A hash function is used in the signature generation process to obtain a condensed version of data, called a message digest .
- The message digest is then input to the DSA to generate the digital signature

Digital Signature Algorithm



Reference:http://www.itl.nist.gov/fipspubs/fip186.htm

References

- Cryptography and Network security – principles and practice : William Stallings
- Applied Cryptography, Second Edition: Protocols, Algorthms, and Source Code in C : Bruce Schneier
- www.certicom.com/index.php/ecc-turorial
- http://campustechnology.com/articles/39190_2
- http://csrc.nist.gov/
- http://www-fs.informatik.uni-tuebingen.de/~reinhard/krypto/English/english.html