CHAPTER 3

UNIT III

Public Key and Management

Syllabus :

At the end of this unit, you should be able to understand and comprehend the following syllabus topics :

- Public Key Cryptography
  - RSA Algorithm
    - Working
    - Key length
    - Security
  - Key Distribution
    - Diffie-Hellman Key Exchange
  - Elliptic Curve
    - Arithmetic
    - Cryptography
    - Security
- Authentication methods
- Message Digest
- Digital Signatures
  - Implementation
  - X.509 Authentication service
  - Algorithms
  - Standards (DSS)
- Kerberos
- Authentication Protocol

3.1 Modular Arithmetic

✍ ***Definition :*** *The modular arithmetic deals with operations on integers specifically around remainders from division.*

– Let's take a few examples.

Dividend	Divisor	Quotient	Remainder (Modulus)
15	5	3	0
15	4	3	3
15	3	5	0
15	2	7	1
15	1	15	0

– So, for example, number 15 when divided by 2, gives you Quotient of 7 and Remainder of 1. This can be mathematically written as 15 *mod* 2 = 1

Note : Before you proceed, try finding out a few mods (remainders from division) for numbers of your choice.

3.1.1 Congruence Property

– Two numbers are said to be in congruence modulo, if they give out the same mod.

– For example,

15 mod 2 = 1

17 mod 2 = 1

– Hence, 15 is congruent to 17 *modulo* 2 i.e. 15 and 17 when undergo mod operation with 2, they both give same result of 1. They can be mathematically denoted as

$$15 \equiv 17 \pmod 2$$

1. Modular Addition

(A + B) mod C = (A mod C + B mod C) mod C

Let A = 12, B = 15 and C = 5

Left Hand Side	Right Hand Side
= (12 + 15) mod 5 = 27 mod 5 = 2	= (12 mod 5 + 15 mod 5) mod 5 = (2 + 0) mod 5 = 2 mod 5 = 2

2. Modular operation on Negative numbers

– If you come across modular operation on a negative number, make the number positive by repetitively adding mod until it becomes positive.

– For example, if you have to find -13 mod 5, keeping adding 5 to –13 until you get a positive number. So,

$$-13 + 5 = -8 + 5 = -3 + 5 = 2$$

– Now, do mod on the positive number you got. In this case, –13 mod 5 becomes 2 mod 5. Hence, the answer is 2.

3. Modular Subtraction

(A – B) mod C = (A mod C – B mod C) mod C

Let A = 12, B = 15 and C = 5

Left Hand Side	Right Hand Side
= (12 – 15) mod 5 = – 3 mod 5 = 2	= (12 mod 5 – 15 mod 5) mod 5 = (2 – 0) mod 5 = 2 mod 5 = 2

4. Modular Multiplication

(A * B) mod C = (A mod C * B mod C) mod C

Let A = 12, B = 15 and C = 5

Left Hand Side	Right Hand Side
= (12 * 15) mod 5 = 180 mod 5 = 0	= (12 mod 5 * 15 mod 5) mod 5 = (2 * 0) mod 5 = 0 mod 5 = 0

5. Modular Inverse

- Modular arithmetic does not have division operation. However, it has inverse. Inverse of a general number is 1 divided by that number. For example, inverse of 2 is ½. In other words, a number when multiplied by its inverse would give 1. So, 2 multiplied by its inverse ½ would give $2 \times ½ = 1$.
- So, modular inverse of A mod C is the value of B such that when A is multiplied by B and the mod C operation is carried out, it gives 1. Mathematically, it can be written as

$$A \times B \equiv 1 (\text{mod } C)$$

Note : To understand the above, refer to the congruency equation. mod C operation on A*B should be same as mod C operation on 1.

- Let's take an example.
- Suppose you have to find modular inverse of 12 mod 5. Here, A = 12 and C = 5. Let's assume value of B from 0 onwards until we find (A* B) mod C = 1.

Value of B	Operation	Result
0	(12 * 0) mod 5	0
1	(12 * 1) mod 5	2
2	(12 * 2) mod 5	4
3	(12 * 3) mod 5	1

- Hence, modular inverse of 12 mod 5 is 3.

3.2 Arithmetic in Cryptography

Cryptography heavily relies on various complex mathematical calculations and operations. Let's learn some of the arithmetic operations used in cryptography.

Prime Numbers

These are whole numbers which are greater than 1 and are only divisible by 1 and itself. For example, 2, 3, 5, 7, 11 and various others.

2. Coprime Numbers

Two integers a and b are said to be relatively prime, mutually prime, or coprime (also written co-prime) if the only positive integer (factor) that divides both of them is 1. For example, 5 and 7 are coprime because their common factor is only 1 whereas 14 and 18 are not coprime because their common factor can also be 2.

3. Discrete Logarithm

- If a is an arbitrary integer relatively prime to n and g is a primitive root of n, then there exists exactly one number μ such that $a = g^{\mu} \pmod n$.
- The number μ is then called the discrete logarithm of a with respect to the base g modulo n and is denoted

  $\mu = \text{ind}_g a \pmod n$.
- Discrete logarithms are quickly computable in a few special cases. However, no efficient method is known for computing them in general. Several important algorithms in public-key cryptography use discrete logarithms.

4. Greatest Common Divisor (GCD)

- Greatest Common Divisor (GCD) of two positive integers is the largest integer that can fully divide both the integers.
- For example, GCD (5, 10) is 5. GCD of (11, 13) is 1. GCD is usually found out by finding the factors of the respective integers and then choosing the common highest factor.
- For example, to find GCD (24, 70)
  - Factors of $24 = 2 \times 2 \times 2 \times 3$: Factors could be 2, 3, 4, 6, 8, 12, 24
  - Factors of $70 = 2 \times 5 \times 7$: Factors are only 2, 5, 7
- Hence, the largest common factor is 2. Hence, GCD (24, 70) = 2.

3.2.1 Euclid's or Euclidean Algorithm

- Finding GCD for smaller numbers is quite straight forward. But, when it comes to finding GCD of large numbers, it might be a complex task. This is precisely where Euclid's (or Euclidean) algorithm helps.
- Euclid's algorithm states that,

$$\gcd(a, b) = \gcd(a \bmod b, b) \quad \text{if } a > b$$

$$\gcd(a, b) = \gcd(a, b \bmod a) \quad \text{if } b > a$$

Ex. 3.2.1 : Find gcd(50, 65) using Euclidean algorithm.

Soln. :

$\gcd(50, 65) = \gcd(50, 65 \bmod 50)$ [because 65 is greater than 50]

$= \gcd(50, 15)$

$= \gcd(50 \bmod 15, 15)$ [because 50 is greater than 15]

$= \gcd(5, 15)$

$= \gcd(5, 15 \bmod 5)$ [because 15 is greater than 5]

= gcd (5, 0) [stop here once the mod of a term becomes 0]

gcd (50, 65) = 5 [is the gcd (50, 65)]

Ex. 3.2.2 : Find gcd(464, 238) using Euclidean algorithm.

Soln. :

gcd (464, 238)

= gcd (464 mod 238, 238)

= gcd (226, 238)

= gcd (226, 238 mod 226)

= gcd(226, 12)

= gcd (226 mod 12, 12)

= gcd(10, 12)

= gcd (10, 12 mod 10)

= gcd(10, 2)

= gcd (10mod 2, 2)

= gcd(0, 2)

gcd (464, 238) = 2

Ex. 3.2.3 : Find gcd(105, 80).

Soln. :

gcd(105, 80)

= gcd (105 mod 80, 80)

= gcd (25, 80 mod 25)

= gcd (225 mod 5, 5)

= gcd (0, 5)

= 5

3.2.2 Extended Euclidean Algorithm

- The Extended Euclidean Algorithm can be used to find the gcd of two numbers, and also to simultaneously express the gcd as a linear combination of these numbers. It helps to find values of coefficients x and y such that to satisfy the following equation:

$$ax + by = gcd(a, b)$$

- In the extended algorithm, the computation involves several entities. First, let's define them:
  - i = index = This would just be used to iterate (repeat) the process.
  - Here if b is greater than a, then assume a = b and b = a. Swap the values to avoid negatives
  - r = remainder = temporary placeholder variable for a and b

- r would be calculated as $r_{i+1} = r_{i-1} - q_i r_i$
- q would be calculated as $q_{i+1} = r_{i-1} / r_i$
- s = temporary placeholder for values while deriving coefficient x
- s would be calculated as $s_{i+1} = s_{i-1} - q_i s_i$
- t = temporary placeholder for values while deriving coefficient y
- t would be calculated as $t_{i+1} = t_{i-1} - q_i t_i$
- x would be $s_{i+1} = \frac{b}{\gcd(a, b)}$
- y would be $t_{i+1} = \frac{a}{\gcd(a, b)}$

Ex. 3.2.4 : For a = 161 and b = 42, calculate gcd(a, b) and also the values of x and y to satisfy the extended Euclidean algorithm.

Soln. :

$r_0 = 161$ and $r_1 = 42$

i starts with 0

First draw the initial table.

Index i	quotient q for i	Remainder r for i	S for i	t for i
0		161	1	0
1		42	0	1

Start steps :

Index i	quotient q for i	Remainder r for i	S for i	t for i
0		161	1	0
1		42	0	1
2	161 / 42 = 3	161 - 3 * 42 = 35	1 - 3 * 0 = 1	0 - 3 * 1 = -3

- For the highlighted row,
  - i = 1 (we are just doing the calculation for 2nd row, hence the value of i is still 1)
  - q_1 can we written as q_i where i = 1

    So, $q_1 = r_{i-1} / r_i = r_0 / r_1 = 161 / 42 = 3$
  - r_2 can be written as r_{i+1} where i = 1
    1. So, $r_2 = r_{i-1} - q_i r_i$ which means $r_2 = r_0 - q_1 * r_1$
    2. $r_2 = 161 - 3 * 42$
       a. $r_{i-1} = r_0$ which is 161
       b. $r_i = r_1$ which is 42

- Similarly, s_2 can be written as s_{i+1} where i = 1

  So, $s_2 = s_{i-1} - q_i s_i = s_0 - q_1 s_1 = 1 - 3*0 = 1$

- Similarly, t_2 can be written as t_{i+1} where i = 1

  So, $t_2 = t_{i-1} - q_i t_i = t_0 - q_1 t_1 = 0 - 3*1 = -3$

Similarly proceed to the next step.

Index i	quotient q for i	Remainder r for i	S for i	t for i
0		161	1	0
1		42	0	1
2	161 / 42 = 3	161 - 3 * 42 = 35	1 - 3 * 0 = 1	0 - 3 * 1 = -3
3	42 / 35 = 1	42 - 1 * 35 = 7	0 - 1 * 1 = -1	1 - 1 * -3 = 4

Here again:

- $q_i = q_2$ = where i = 2

  So, $q_2 = r_{i-1} / r_i = r_1 / r_2 = 42 / 35 = 1$

- $r_3 = r_1 - q_2 * r_2 = 42 - 1 * 35 = 7$
- $s_3 = s_1 - q_2 * s_2 = 0 - 1 * 1 = -1$
- $t_3 = t_1 - q_2 * t_2 = 1 - 1 * -3 = 4$

Index i	quotient q for i	Remainder r for i	S for i	t for i
0		161	1	0
1		42	0	1
2	161 / 42 = 3	161 - 3 * 42 = 35	1 - 3 * 0 = 1	0 - 3 * 1 = -3
3	42 / 35 = 1	42 - 1 * 35 = 7	0 - 1 * 1 = -1	1 - 1 * -3 = 4
4	35 / 7 = 5	35 - 5 * 7 = 0	Do not calculate	Do not calculate

- $q_i = q_3$ where i = 3

  So, $q_3 = r_2 / r_3 = 35 / 7 = 5$

- $r_4 = r_2 - q_3 * r_3 = 35 - 5 * 7 = 0$
- Do not calculate s and t once you get r = 0
- Last calculated s and t become x and y respectively
- Last calculated r becomes gcd

So, according to extended Euclidean algorithm, for numbers 161 and 42

gcd(161, 42) = 7

– In the equation

$$ax + by = \gcd(161, 42)$$

$$x = -1$$

$$y = 4$$

– You can verify the answer by putting the values in the equation.

Left-hand side:

$$ax + by = 161 * -1 - 42 * 4$$

$$= -161 + 168$$

$$= 7 \quad \text{[which is equal to gcd(161, 42)]}$$

Ex. 3.2.5 : For a = 256 and b = 5004, calculate gcd (a, b) and also the values of x and y to satisfy the extended Euclidean algorithm.

Soln. :

– First note that b > a. Hence, let's swap the values to make the calculations simple. We would re-swap them in the final answer.

– So, assume a = 5004 and b = 256

Index i	quotient q for i	Remainder r for i	S for i	t for i
0		5004	1	0
1		256	0	1

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		5004	1	0
1		256	0	1
2	5004 / 256 = 19	5004 – 19 * 256 = 140	1 – 19 * 0 = 1	0 – 19 * 1 = –19
3	256 / 140 = 1	256 – 1 * 140 = 116	0 – 1 * 1 = –1	1 – 1*-19 = 20
4	140 / 116 = 1	140 – 1 * 116 = 24	1 – –1 * 1 = 2	–19 –1*20 = –39
5	116 / 24 = 4	116 – 24 * 4 = 20	–1 – 4 * 2 = – 9	20 – –39*4 = 176
6	24 / 20 = 1	24 – 20 * 1 = 4	2 – –9 * 1 = 11	–39 – 1*176 = –215
7	20 / 4 = 5	20 – 4* 5 = 0	Do not calculate	Do not calculate

gcd(256, 5004) = 4

We originally swapped the value. So, re-swap it.

Hence, x = – 215 and y = 11

Putting it in the equation, you get,

Left-Hand Side

$= ax + by$

$= 256 * -215 + 5004 * 11$

$= -55{,}040 + 55{,}044$

$= 4$ [which is equal to right hand side = gcd (256, 5004)]

Ex. 3.2.6 : For a = 86 and b = 14, calculate gcd(a, b) and also the values of x and y to satisfy the extended Euclidean algorithm.

Soln. :

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		86	1	0
1		14	0	1
2	86 / 14 = 6	86 – 6*14 = 2	1 – 6*0 = 1	0 – 6*1 = -6
3	14 / 2 = 7	14 – 2*7 = 0	Do not calculate	Do not calculate

gcd(86, 14) = 2

x = 1, y = – 6

To verify, let's put the above values in the equation:

$ax + by = 86 * 1 - 14 * 6$

$= 86 - 84$

$= 2$ [this is the gcd value that we got f or 86, 14]

Ex. 3.2.7 : For a = 999 and b = 9, calculate gcd(a, b) and also the values of x and y to satisfy the extended Euclidean algorithm.

Soln. :

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		999	1	0
1		9	0	1
2	999 / 9 = 111	999 – 111*9 = 0	Do not calculate	Do not calculate

gcd (999, 9) = 9

x = 0, y = 1

Let's put those values in the equation :

$ax + by = 999 * 0 + 1 * 9$

$= 9$ [which matches the gcd value we got f or 999, 9]

Tip : It would perhaps be easy for you to calculate the first two columns q and r until you get 0 in r. That way you are focusing on one set of calculation at a time. Once you have q and r computed in the first two columns, calculate s and t by substitution the values of q and r in the respective equations.

3.2.3 Multiplicative Inverse using extended Euclidean Algorithm

- If you recall our discussion from the previous section on modular inverse, you understand what inverse operation is. One application of the extended Euclidean Algorithm is to find out multiplicative inverse.

Definition : A modular multiplicative inverse of an integer a is an integer x such that the product ax is congruent to 1 with respect to the modulus m

- In modular arithmetic, it can be written as $ax \equiv 1 \pmod m$
- According to the extended Euclidean Algorithm,

$$ax + my = \gcd(a, m) = 1$$

$$ax + my = 1$$

$$ax - 1 = (-y)m$$

- Dividing both sides by (mod m)

$$ax \pmod m - 1 \pmod m = (-y)m \pmod m$$

$$ax \pmod m - 1 \pmod m = 0$$

$$ax \equiv 1 \pmod m$$

Note : The multiplicative inverse of a modulo m exists if and only if a and m are coprime (i.e., if gcd(a, m) = 1)

- So, if you come across a question where it is asked to calculate multiplicative inverse such that gcd(a, m) is not 1, do not attempt to solve the problem. Just calculate the gcd and show that the numbers are not coprime and hence the multiplicative inverse does not exist.

Ex. 3.2.8 : Find multiplicate inverse of 24140 mod 40902.

Soln. :

$$\begin{aligned}
\gcd(24140, 40902) &= \gcd(24140, 40902 \bmod 24140) \\
&= \gcd(24140, 16762) \\
&= \gcd(24140 \bmod 16762, 16762) \\
&= \gcd(7378, 16762) \\
&= \gcd(7378, 16762 \bmod 7378) \\
&= \gcd(7378, 2006) \\
&= \gcd(7378 \bmod 2006, 2006) \\
&= \gcd(1360, 2006) \\
&= \gcd(1360, 2006 \bmod 1360) \\
&= \gcd(1360 \bmod 646, 646)
\end{aligned}$$

= gcd (68, 646 mod 68)

= gcd(68 mod 34, 34)

= gcd(0, 34)

gcd (24140, 40902) = 34

– Here you find that gcd(24140, 40902) = 34. Hence, multiplicative inverse of 24140 mod 40902 does NOT exist.

Note : You can also calculate gcd using tabular method as you learnt in the extended Euclidean algorithm section to avoid repeating the gcd steps to calculate x and y if gcd = 1 does exist.

Ex. 3.2.9 : Find multiplicate inverse of 8 mod 11.

Soln. :

– Since 8 < 11, let's swap the values for simplicity.

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		11	1	0
1		8	0	1

– Calculate next steps.

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		11	1	0
1		8	0	1
2	11 / 8 = 1	11 – 8*1 = 3	1 – 1*0 = 1	0 – 1*1 = –1
3	8 / 3 = 2	8 – 3*2 = 2	0 – 2*1 = –2	1 – 2*–1 = 3
4	3 / 2 = 1	3 – 2*1 = 1	1 – 1*–2 = 3	–1 – 1*3 = – 4
5	2 / 1 = 2	2 – 1*2 = 0	Do not calculate	Do not calculate

– Let's re-swap the values.

Hence, x = – 4 and y = 3

– Putting the values in the extended Euclidean algorithm, you get

$ax + by = 1$

$8(-4) + 11(3) = 1$

– Since, you have to find multiplicative inverse in mod 11, divide both sides by mod 11.

$8(-4) \bmod 11 + 11(3) \bmod 11 = 1 \bmod 11$

$8(-4) \bmod 11 + 0 = 1$

– Recall our discussion on calculating mod for negative numbers. You need to keep adding mod until the number turns positive and then calculate mod on the positive number you got.

$-4 + 11 = 7$

$7 \bmod 11 = 7$

Hence, $8(7) \bmod 11 = 1$

- So, multiplicative inverse of 8 mod 11 is 7.

Note : You can also test your solution. If there are mistakes, re-visit the steps you took. In this example,

$8 \times 7 = 56$ and $56 \bmod 11 = 1$. Hence, you find that 7 is indeed multiplicative inverse of 7 in mod 11.

Ex.3.2.10 : Find multiplicate inverse of 1234 mod 4321.

Soln. :

- Since 1234 < 4321, let's swap the values for simplicity.

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		4321	1	0
1		1234	0	1
2	4321 / 1234 = 3	4321 – 3*1234 = 619	1 – 3*0 = 1	0 – 3*1 = –3
3	1234 / 619 = 1	1234 – 1*619 = 615	0 - 1*1 = –1	1 – 1*-3 = 4
4	619 / 615 = 1	619 –1*615 = 4	1– 1*–1 = 2	–3 – 1*4 = – 7
5	615 / 4 = 153	615 – 4*153 = 3	–1 – 153*2 = –307	4 –153*-7 = 1075
6	4 / 3 = 1	4 – 1*3 = 1	2 – 1*–307 = 309	–7 – 1*1075 = –1082
7	3 / 1 = 3	3 – 3*1 = 0	Do not calculate	Do not calculate

- Let's re-swap the values.
- Hence, $x = -1082$ and $y = 309$
- Putting it in the equation,

$$ax + by = 1$$

$$1234\,(-1082) + 4321\,(309) = 1$$

Dividing both sides by mod 4321, you get

$$1234(-1082) \bmod 4321 + 4321(309) \bmod 4321 = 1 \bmod 4321$$

$$1234(-1082) \bmod 4321 + 0 = 1$$

Convert -1082 to positive

$$-1082 + 4321 = 3239$$

$$3239 \bmod 4321 = 3239$$

Hence,

$$1234\ (3239) \bmod 4321 = 1$$

Or 3239 is multiplicative modular inverse of 1234 in mod 4321.

3.2.4 Chinese Remainder Theorem

- The Chinese Remainder Theorem (CRT) helps to solve a system of simultaneous linear congruences.
- Let $m_1, m_2, \ldots, m_r$ be a collection of pairwise relatively prime integers. Then the system of simultaneous congruences

$$x \equiv a_1 \pmod{m_1}$$
$$x \equiv a_2 \pmod{m_2}$$
$$\ldots$$
$$x \equiv a_r \pmod{m_r}$$

- It has a unique solution modulo $M = m_1 m_2 \ldots m_r$ for any given integers $a_1, a_2, \ldots, a_r$.

Ex. 3.2.11 : Find the value of x using CRT when $x \equiv 2 \bmod 7$, $x \equiv 2 \bmod 7$, $x \equiv 3 \bmod 9$

Soln.:

- Since, $x \equiv 2 \bmod 7$ is repeated twice, consider it just once else gcd(7, 7) would not be 1 and hence you would not be able to solve this problem.
- Here,

$$a_1 = 2, a_2 = 3$$
$$m_1 = 7, m_2 = 9$$

- According to CRT,

$$x = (m_1 x_1 a_1 + m_2 x_2 a_2) \bmod M$$

Step 1 : Calculate the product of all mod

$$M = m_1 * m_2 = 7 * 9 = 63$$
$$M_1 = M / m_1 = 63 / 7 = 9$$
$$M_2 = M / m_2 = 63 / 9 = 7$$

Step 2 : Calculate inverse modulo for each congruence

$$M_1 X_1 \equiv 2 \pmod 7$$
$$9X_1 \equiv 2 \pmod 7$$

Value of X1	Operation	Result
0	(9 * 0) mod 7	0
1	(9 * 1) mod 7	2
2	(9 * 2) mod 7	4
3	(9 * 3) mod 7	6
4	(9 * 4) mod 7	1

- Hence, modulo inverse of $9X1 \equiv 2 \pmod 7$ is 4. Hence, X1 = 4
- Similarly,

$$M_2X_2 \equiv 3 \pmod 9$$

$$7X_2 \equiv 3 \pmod 9$$

Value of X_2	Operation	Result
0	(7 * 0) mod 9	0
1	(7 * 1) mod 9	7
2	(7 * 2) mod 9	5
3	(7 * 3) mod 9	3
4	(7 * 4) mod 9	1

Hence, $X_2 = 4$

Putting the values in the CRT equation, you get

$$x = (m1\ x1a1 + m2x2a2) \bmod M$$

$$x = (9 * 4 * 2 + 7 * 4 * 3) \bmod 63$$

$$x = (72 + 84) \bmod 63$$

$$x = 156 \bmod 63$$

$$x = 30$$

So, $30 \equiv 2 \bmod 7$,

$30 \equiv 2 \bmod 7$,

$30 \equiv 3 \bmod 9$

You can also verify your answer.

30 mod 7 = 2 [which aligns with equation 1 and 2]

30 mod 9 = 3 [which aligns with equation 3]

Ex. 3.2.12 : In a school picnic,

1. If the children were arranged in the group of 3, 2 children were left out.
2. If the children were arranged in the group of 4, 3 children were left out.
3. If the children were arranged in the group of 5, 4 children were left out.

Find out the minimum number of children that could be in the school picnic.

Soln. :

- Assume that the number of children in the school picnic is x. The above information can be written as below:

$x \equiv 2 \bmod 3$ [as x mod 3 would give 2 as per the given information]

$x \equiv 3 \bmod 4$ [as x mod 4 would give 3 as per the given information]

$x \equiv 4 \bmod 5$ [as x mod 5 would give 4 as per the given information]

gcd(3, 4) = 1 gcd(3, 5) = 1 gcd(4, 5) = 1

- Hence, all the given mod are coprime and there exists a solution x. Let's continue.

$a_1 = 2$ $a_2 = 3$ $a_3 = 4$

$m_1 = 3$ $m_2 = 4$ $m_3 = 5$

- Calculate product of all mod

$$M = m_1 * m_2 * m_3$$

$$M = 3*4*5 = 60$$

$$M_1 = 60/3 = 20$$

$$M_2 = 60/4 = 15$$

$$M_3 = 60/5 = 12$$

- Calculate inverse modulo for each congruence

$$M_1X_1 \equiv 2 \bmod 3$$

$$20X_1 \equiv 2 \bmod 3$$

Value of X_1	Operation	Result
0	(20 * 0) mod 3	0
1	(20 * 1) mod 3	2
2	(20 * 2) mod 3	1

Hence, $X_1 = 2$

$$M_2X_2 \equiv 3 \bmod 4$$

$$15X_2 \equiv 3 \bmod 4$$

Value of X_2	Operation	Result
0	(15 * 0) mod 4	0
1	(15 * 1) mod 4	3
2	(15 * 2) mod 4	2
3	(15 * 3) mod 4	1

Hence, $X_2 = 3$

$$M_3X_3 \equiv 4 \bmod 5$$

$$12X_3 \equiv 4 \bmod 5$$

Value of X_3	Operation	Result
0	(12 * 0) mod 5	0
1	(12 * 1) mod 5	2
2	(12 * 2) mod 5	4
3	(12 * 3) mod 5	1

Hence, $X_3 = 3$

Putting the above values in the CRT equation, you get

$$x = (M_1X_1a_1 + M_2X_2a_2 + M_3X_3a_3) \bmod M$$

$$x = (20*2*2 + 15*3*3 + 12*3*4) \bmod 60$$

$$x = (80 + 135 + 144) \bmod 60$$

$$x = 359 \bmod 60$$

$$x = 59$$

- Therefore, there are 59 children in the school picnic.
- You can also verify your answer.

$59 \bmod 3 = 2$

$59 \bmod 4 = 3$

$59 \bmod 5 = 4$

Ex. 3.2.13 : Find the value of x using CRT when $x \equiv 10 \bmod 3$, $x \equiv 11 \bmod 4$, $x \equiv 12 \bmod 5$

Soln. :

$\gcd(3, 4) = 1$ $\gcd(3, 5) = 1$ $\gcd(4, 5) = 1$

- Hence, all the given mod are coprime and there exists a solution x. Let's continue.

$a_1 = 10$ $a_2 = 11$ $a_3 = 12$

$m_1 = 3$ $m_2 = 4$ $m_3 = 5$

- Calculate product of all mod

$$M = m_1 * m_2 * m_3$$

$$M = 3*4*5 = 60$$

$$M_1 = 60/3 = 20$$

$$M_2 = 60/4 = 15$$

$$M_3 = 60/5 = 12$$

- Calculate inverse modulo for each congruence

$$M_1X_1 \equiv 10 \bmod 3$$

$20X_1 \equiv 10 \bmod 3$

Value of X_1	Operation	Result
0	(20 * 0) mod 3	0
1	(20 * 1) mod 3	2
2	(20 * 2) mod 3	1

Hence, $X_1 = 2$

$M2X_2 \equiv 11 \bmod 4$

$15X_2 \equiv 11 \bmod 4$

Value of X_2	Operation	Result
0	(15 * 0) mod 4	0
1	(15 * 1) mod 4	3
2	(15 * 2) mod 4	2
3	(15 * 3) mod 4	1

Hence, $X_2 = 3$

$M_3X_3 \equiv 12 \bmod 5$

$12X_3 \equiv 12 \bmod 5$

Value of X_3	Operation	Result
0	(12 * 0) mod 5	0
1	(12 * 1) mod 5	2
2	(12 * 2) mod 5	4
3	(12 * 3) mod 5	1

Hence, $X_3 = 3$

– Putting the above values in the CRT equation, you get

$$x = (M_1X_1a_1 + M_2X_2a_2 + M_3X_3a_3) \bmod M$$

$$x = (20 * 2 * 10 + 15 * 3 * 11 + 12 * 3 * 12) \bmod 60$$

$$x = (400 + 495 + 432) \bmod 60$$

$$x = 1327 \bmod 60$$

$$x = 7$$

– Therefore, value of x is 7.

- You can also verify your answer.

  7 mod 3 = 1 [which is same as 10 mod 3]

  7 mod 4 = 3 [which is same as 11 mod 4]

  7 mod 5 = 2 [which is same as 12 mod 5]

3.2.5 Fermat's Theorem

- Fermat's theorem, also known as Fermat's little theorem or Fermat's primality test, states that for any prime number p and any integer a such that p does not divide a (the pair are relatively prime), p divides exactly into $a^p - a$.
- This can be expressed as

$$a^p \equiv a \pmod{p}$$

- Another variant of this theorem is when a is not divisible by p.

$$a^{p-1} \equiv 1 \pmod{p}$$

Proof :

Let $a = 2$ and $p = 7$

$a^7 = 2^7 = 128$

$a^7 - a = 128 - 2 = 126$

$126 = 7 * 18$ and no remainder

- The second variant can be similarly proved.

$$a^{7-1} = a^6 = 2^6 = 64$$

- Now, 64 mod 7 = 1
- Hence, Proved.

Ex. 3.2.14 : Find $2^{16} \pmod{17}$

Soln. :

- You can re-write $2^{16} \pmod{17}$ as

$$2^{17-1} \pmod{17}$$

- According to Fermat's theorem

$$a^{p-1} \equiv 1 \pmod{p}$$

$$2^{17-1} \equiv 1 \pmod{17}$$

- Hence, $2^{16} \pmod{17} = 1$

Ex. 3.2.15 : Find $2^{50} \pmod{17}$

Soln. :

- You can re-write $2^{50} \pmod{17}$ as

$$[(2^{16})^3 * 2^2] \pmod{17} = [(2^{17-1} \pmod{17})^3 * 4 \pmod{17}]$$

$[(2^{16})^3 * 2^2] \pmod{17} = 1^3 * 4 \pmod{17}$ [because $2^{16} \pmod{17} = 1$ according to Fermat's theorem]

$= 4 \bmod 17$

$= 4$

Hence, $2^{50} \pmod{17} = 4$

3.2.6 Euler's theorem

- Euler's theorem (also known as the Fermat–Euler theorem or Euler's totient theorem) states that if *n* and *a* are coprime positive integers, then

  $a^{\Phi(n)} \equiv 1 \pmod{n}$ where $\Phi(n)$ is Euler's totient function.

- Euler's totient function counts the positive integers up to a given integer *n* that are relatively prime to *n*. It is denoted by the Greek letter phi $\Phi(n)$. These coprime numbers are also called as totatives.

Ex. 3.2.16 : Find Φ(n) where n = 5.

Soln. :

- Numbers greater than or equal to 1 and less than 5 are 1, 2, 3 and 4.
- Each pair is a coprime with 5 because
  - gcd(1, 5) = 1
  - gcd(2, 5) = 1
  - gcd(3, 5) = 1
  - gcd (4, 5) =1
- Hence, $\phi(5) = 4$ [that is there 4 coprime numbers with respect to 5].

Ex.3.2.17 : Find totatives where n = 10.

Soln. :

- Numbers greater than or equal to 1 and less than 10 are 1, 2, 3, 4, 5, 6, 7, 8 and 9
- Find out coprime pairs with 10
  - gcd(1, 10) = 1
  - gcd(2, 10) = 2
  - gcd(3, 10) = 1
  - gcd(4, 10) = 2
  - gcd(5, 10) = 5
  - gcd(6, 10) = 2
  - gcd(7, 10) = 1
  - gcd(8, 10) = 2
  - gcd(9, 10) = 1

Out of the above, only 1, 3, 7 and 9 are coprime with number 10 (since their gcd is 1). Hence, 1, 3, 7 and 9 are the totatives of number 10. Also, $\phi(10) = 4$ [total count of totatives].

Ex. 3.2.18 : Find the value of $7^{10} \pmod{10}$ using Euler's theorem.

Soln. :

- According to Euler's theorem,

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

$$\phi(10) = 4 \text{ [as you calculated from the previous practice question example]}$$

- So,

$$a^4 \equiv 1 \pmod{n}$$

$$7^4 \equiv 1 \pmod{10}$$

So, $7^4 = 1$

- So, $7^{10} \pmod{10}$ can be re-written as,

$$= (7^4 * 7^4 * 7^2) \pmod{10}$$

$$= 7^4 \pmod{10} * 7^4 \pmod{10} * 7^2 \pmod{10}$$

$$= 1 * 1 * 49 \pmod{10}$$

$$= 49 \bmod 10$$

$$= 9$$

- Hence, $7^{10} \pmod{10} = 9$.

$x^{\phi(n)} = 1 \bmod n$

↓ if n is prime

$x^{n-1} = 1 \bmod n$

$x^{\phi(n)+1} = x \bmod n$

$x^{(n-1)+1} = x \bmod n$

$x^n = x \bmod n$

3.3 Public Key Cryptography

The public key cryptography is based on the asymmetric keys that you learnt earlier in this chapter. Recall and revise that section before you proceed.

3.3.1 Principles of Public Key Cryptosystems

- Following are some basic principles of public key-based cryptosystems.

1. Public key cryptosystems require the use of two keys – a public key and a private key.
2. Public keys are widely known.
3. Private key is kept secret with its owner.
4. The two keys are mathematically related and form a key pair.
5. One key in the key pair cannot be used to derive the other key in the key pair.
6. Any key in the key pair can be used for encryption. The other key then must be used for decryption.
7. Sender and the receiver both must have their own key pairs.

- In this section, you are going to learn about various asymmetric key based algorithms.

3.4 RSA

- RSA, named after its inventors Ron Rivest, Adi Shamir and Leonard Adleman, is an asymmetric key based algorithm. As you understand, RSA, or any other asymmetric key based algorithms can be used for confidentiality [encryption, decryption], authentication and non-repudiation.
- RSA is based on finding prime factors for very large numbers. The length of numbers that we are referring to here is around 500 digits!

RSA Key Length	Number of digits
1024-bit	309
2048-bit	617
4096-bit	1233

- Let's understand how RSA derives public and private keys and how does encryption and decryption process work based on the derived keys.

1. Choose two random large prime numbers, p and q
2. Multiply the numbers. $n = p*q$
3. Choose a random integer to be encryption key e such that e and $(p-1)(q-1)$ are relatively prime.
4. Decryption key is computed as $d = e^{-1} \bmod ([p-1]*[q-1])$
5. The public key = (n, e)
6. The private key = (n, d)
7. For encrypting message M with public key (n, e), you get ciphertext $C = M^e \bmod n$
8. For decrypting ciphertext with private key (n, d), you get plaintext $M = C^d \bmod n$

Ex. 3.4.1 : Perform encryption and decryption using RSA algorithm with p = 7, q = 11, e = 17 and M = 8.

Soln. :

$n = p * q$

$n = 7 * 11 = 77$

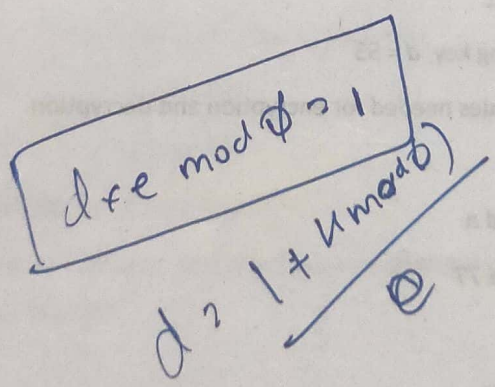
$r = (p-1) * (q-1)$

$r = 6*10 = 60$

$d = e^{-1} \bmod r$

$ed \equiv 1 \bmod 60$

$17d \equiv 1 \bmod 60$



- Let's calculate modulo inverse using extended Euclidean algorithm (swapping 17 and 60)

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		60	1	0
1		17	0	1
2	60 / 17 = 3	60 – 3*17 = 9	1 – 3*0 = 1	0 – 3*1 = –3
3	17 / 9 = 1	17 – 1*9 = 8	0 – 1*1 = –1	1 – 1*-3 = 4
4	9 / 8 = 1	9 – 1*8 = 1	1 – 1*–1 = 2	–3 – 1*4 = –7
5	8 / 1 = 8	8 – 1*8 = 0	Do not calculate	Do not calculate

- Let's re-swap the values.

$$x = -7$$

$$y = 2$$

- Putting the values in the extended Euclidean algorithm, you get

$$ax + by = 1$$

$$117(-7) + 60(2) = 1$$

- Since, you have to find multiplicative inverse in mod 60, divide both sides by mod 60.

$$17(-7) \bmod 60 + 60(2) \bmod 60 = 1 \bmod 60$$

$$17(-7) \bmod 60 + 0 = 1$$

- Recall our discussion on calculating mod for negative numbers. You need to keep adding mod until the number turns positive and then calculate mod on the positive number you got.

$$-7 + 60 = 53$$

$$53 \bmod 60 = 53$$

- Hence, $17(53) \bmod 60 = 1$
- Hence, value of decrypting key, $d = 53$
- Now, you have all the values needed for encryption and decryption.
- As per RSA,

$$C = M^e \bmod n$$

$$C = 8^{17} \bmod 77$$

$$C = 57$$

$$M = C^d \bmod n$$

$$M = 57^{53} \bmod 77$$

$$M = 8$$

Ex. 3.4.2 : Given two Prime Numbers P = 17 and Q = 29 find out N, E and D in an RSA encryption process.

Soln. :

$$N = p * q = 17 * 29 = 493$$

$$r = (p-1) * (q-1)$$

$$r = 16*28 = 448$$

- Now, choose a random integer to be encryption key e such that e and r are relatively prime.
- Let e = 3. 3 is coprime with 448.

$$d = e^{-1} \bmod r$$

$$ed \equiv 1 \bmod 448$$

$$3d \equiv 1 \bmod 448$$

- Let's calculate modulo inverse using extended Euclidean algorithm (swapping 3 and 448)

Index i	quotient q for i	Remainder r for i	s for i	t for i
0		448	1	0
1		3	0	1
2	448 /3 = 149	448 – 149 *3 = 1	1 – 0*149 = **1**	0 – 1*149 = **–149**
3	3 / 1 = 3	3 – 1* 3 = 0	Do not calculate	Do not calculate

Let's re-swap the values.

$$x = -149$$

$$y = 1$$

Putting the values in the extended Euclidean algorithm, you get

$$ax + by = 1$$

$$3*(-149) + 448*(1) = 1$$

Since, you have to find multiplicative inverse in mod 448, divide both sides by mod 448.

$$3*(-149) \bmod 448 + 448*(3) \bmod 448 = 1 \bmod 448$$

$$3*(-149) \bmod 448 + 0 = 1 \bmod 448$$

$$3*(-149) \bmod 448 = 1 \bmod 448$$

Recall our discussion on calculating mod for negative numbers. You need to keep adding mod until the number turns positive and then calculate mod on the positive number you got.

$$-149 + 448 = 299$$

$$299 \bmod 448 = 299$$

- Hence, multiplicative inverse is 299.
- Therefore, the value of decrypting key, $d = 299$.

- Now, you have all the values needed for encryption and decryption. Let's assume that you want to encrypt a m
  M = 10.
- As per RSA,

$$C = M^e \bmod n$$
$$C = 10^3 \bmod 493$$
$$C = 14 \text{ [encrypted message]}$$
$$M = C^d \bmod n$$
$$M = 14^{299} \bmod 493$$
$$M = 10 \text{ [decrypted message]}$$

3.4.1 Attacks on RSA

1. **Brute-force attack**
   Here the attacker tries to find factors of *n* by trying out various possibilities.
2. **Common modulus**
   To avoid generating a different modulus $n = p*q$ for each user one may wish to fix *n* for all the users. It might s like deriving the decrypting key *d* is not possible for every encrypting key *e* by any other user since encrypting ke a randomly chosen value. But, the problem with this approach is that a particular user who knows her pair of e can successfully use her own pair to find the factors for common modulus. Once the factors are known, since encrypting key *e* is known to everyone (because it is public key), the decrypting key *d* could be found out. Hence should not be using common modulus to generate keys for multiple users.
3. **Choosing smaller numbers**
   The security of the algorithm comes from the fact that factoring large numbers is computationally inten Sometimes, to improve system performance, smaller numbers can be chosen which can significantly enhance performance but at the cost of making the algorithm weaker. Hence, you should always choose large numbers maintain the strength of the algorithm.
4. **Man in the middle attack**
   The attacker could collect all the ciphertext coming out from the user's system (that is encrypted with her private ke and try to find the private key. The information known to the attacker is the ciphertext and public key.

3.5 Diffie-Hellman Key Exchange Algorithm

- If you recall, one of the challenges with using symmetric key algorithms was key distribution. How could sender a the receiver agree upon the key that would be used for encryption and decryption (recall that the symmetric ke based algorithms use the SAME key for both encryption as well as decryption)? I asked you this question in th symmetric key section and here I am again to help you with the answer. Diffie-Hellman algorithm is one of th answers to the question.

Definition : *The Diffie-Hellman algorithm provides a way of generating a shared secret between the send and the receiver in such a way that the secret need not be exchanged or transferred over the communicat medium.*

- Basically, the sender and the receiver create the key together at their respective ends at the same time. The key that they create at their respective ends is mathematically computed to be the same.
- Hence, the key distribution need not happen, and the sender and the receiver can confidentially communicate using the key they created. You should note here that the Diffie-Hellman algorithm is NOT used for actual encryption and decryption process. It is used only for key generation.
- Let's understand the steps involved in generating the shared key. Assume that there are two users Alex and Bobby who need to generate a shared key for securely communicating with each other.
  1. Alex chooses two prime numbers g and p and also a secret number a. He calculates value of A such that $A = g^a \bmod p$. He then sends g, p and A to Bobby. Note here that Alex does not share the secret number a with Bobby.
  2. Similarly, Bobby chooses a secret number b and computes the value of B such that $B = g^b \bmod p$. She then sends B to Alex.
  3. Alex computes the shared key at his end as Shared Key, $S = B^a \bmod p$
  4. Bobby computes the shared key at her end asShared Key, $S = A^b \bmod p$
- The values of the shared key, S, derived in the step 3 and 4 are equal due to mod operation.

$$(g^a \bmod p)^b \bmod p = g^{ab} \bmod p$$

$$(g^b \bmod p)^a \bmod p = g^{ba} \bmod p$$

- It does not matter which step you do earlier. Both the keys created would be equal and can be used with any of the algorithms such as DES and AES to encrypt the information and communicate confidentially.
- Let's understand the algorithm by solving a question.

Ex. 3.5.1 : Calculate shared key between two users if the initial chosen prime numbers are 5 and 7.

Soln. :

$g = 5$

$p = 7$

- Assume that the user Alex chooses a secret number a = 2

$A = g^a \bmod p$

$A = 5^2 \bmod 7$

$A = 25 \bmod 7$

$A = 4$

- Alex sends g, p and A to Bobby.
- Assume that the user Bobby chooses a secret number b = 3

$B = g^b \bmod p$

$B = 5^3 \bmod 7$

$B = 125 \bmod 7$

$B = 6$

- Bobby sends B to Alex.
- Now, both the users compute the shared key, S, at their respective ends.
- Alex calculates it as

$$S = B^a \bmod p$$

$$S = 6^2 \bmod 7$$

$$S = 36 \bmod 7$$

$$S = 1$$

- Bobby calculates it as

$$S = A^b \bmod p$$

$$S = 4^3 \bmod 7$$

$$S = 64 \bmod 7$$

$$S = 1$$

- So, the shared key, that can be used between Alex and Bobby, is $S = 1$.

Ex. 3.5.2 Find the key exchanged between Alok and Bobby considering following data.

(i) n =11

(i) g =5

(iii) X = 2, Y =3

Find value of A, B and secret key K. **SPPU – March 19(In Sem.), 5 M**

Soln. :

$$g = 5$$

$$n = 11$$

- For user A,

$$A = g^x \bmod n$$

$$A = 5^2 \bmod 11$$

- Hence, the public key of user A is 3.
- For user B,

$$B = g^y \bmod n$$

$$B = 5^3 \bmod 11$$

- Hence, the public key of user B is 4.
- Now, both the users compute the shared key, S, at their respective ends.
- User A calculates it as

$$S = B^x \bmod n$$

$$S = 4^2 \bmod 11$$

$$S = 5$$

- User B calculates it as

$$S = A^Y \bmod n$$

$$S = 3^3 \bmod 11$$

$$S = 5$$

- Hence, the shared key, S between User Alok and User Bobby is 5.

Note : The example we took here involves very small numbers to make it easy for you to understand the steps involved in the key generation. Practically, the numbers used in the shared key generation are extremely large, possibly containing around 500 digits!

3.6 Elliptic Curve Arithmetic and Cryptography

Q. Discuss elliptic curve cryptography in detail. SPPU – May 19 (May 19, 5 Marks)

- Computers are getting faster. Algorithms such as RSA might be unusable in coming years because the existing key-lengths might not provide adequate operation.
- Continuously increasing the key-length might not be a possible solution because increasing key-lengths also significantly increases time it takes to encrypt and decrypt the information.

Definition : *Elliptic Curve Cryptography (ECC) is a public-key cryptography system which is based on discrete logarithms structure of elliptic curves over finite fields.*

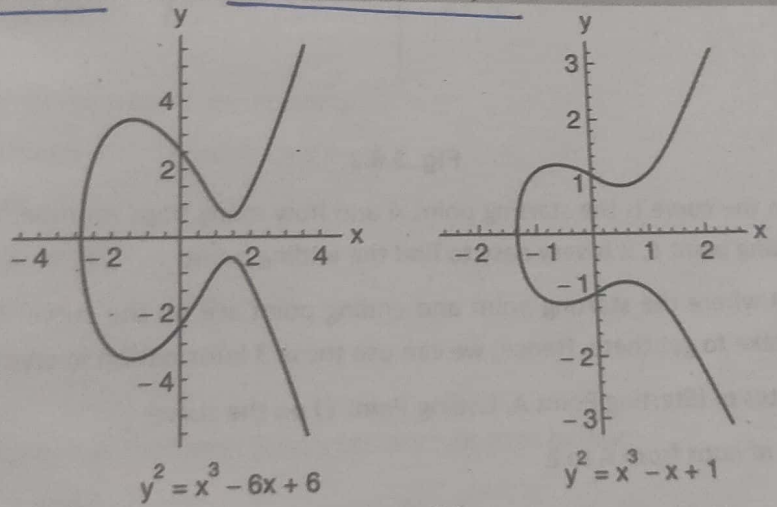


Fig. 3.6.1

ECC uses an elliptic curve over a finite field (p) of the form $y^2 = x^3 + ax + b \pmod p$

The curve defines a finite field consisting of points that satisfy this equation along with infinity (∞) as the identity element. The value of a and b determines the shape of the curve. Only those curves which don't have repeated factors for $x^3 + ax + b$ are used in cryptography.

Given are the two plots for a = – 6 and –1 and b = 6 and 1 respectively.

3.6.1 How does it work?

- ECC uses a trapdoor function. The trapdoor function is similar to a mathematical game of pool. You start with a point on the curve and using the dot function you find a new point on the curve. You keep repeating the dot function from point to point until you reach the desired point on the curve.
- So, based on the diagram,
  - You start from point A and go to B. Reflect the point B in the opposite axis.
  - It reflects at point C.
  - From C you go to D and reflect again cross X-axis as E
  - You keep repeating it until you reach the desired point

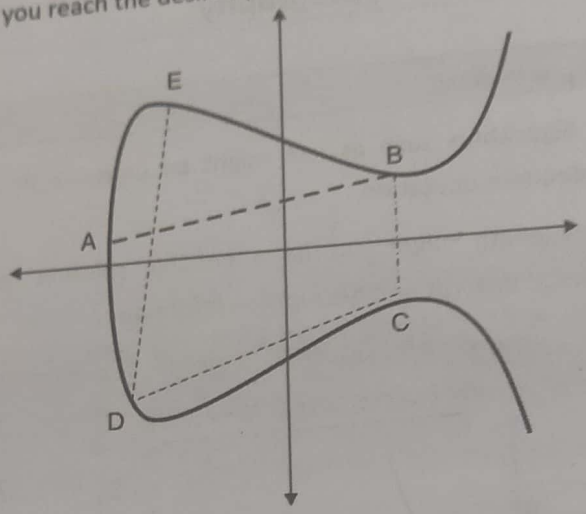


Fig. 3.6.2

- So, if you know where on the curve is the starting point *A* and how many hops (number of dot functions needed) are required to reach the ending point *E*, it is very easy to find the ending point.
- But, if you just know that where the starting point and ending point are on the curve, it is nearly impossible to know how many hops it would take to get there. Hence, we can use these 3 information in cryptography as below:
  - Public Key : Coordinates of (Starting Point A, Ending Point E) on the curve
  - Private Key : Number of hops from A to E

Advantages of ECC

1. **Smaller key sizes :** ECC requires smaller key sizes for similar protection compared to RSA. For example, 256-bit key is considered to be equivalent of 3072-bit RSA key. Following is a comparison chart for key sizes in bits.

Symmetric Key Size	RSA Key Size	ECC Key Size
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

2. **Faster operation :** Due to smaller key sizes and the mathematical algorithm, ECC is faster than RSA.

Disadvantages of ECC

1. Implementing the algorithm securely is difficult
2. Evolutionary standard : Not all parts of the standard are thoroughly adopted in the industry.

Usage of ECC

1. Elliptic Curve Diffie–Hellman (ECDH) key agreement
2. Elliptic Curve based encryption e.g. Elgamal
3. Elliptic Curve Digital Signature Algorithm (ECDSA) for digital signature and authentication

3.7 ElGamal Curve Arithmetic and Cryptography

- ElGamal is a public key algorithm that can be used for digital signatures, encryption, and key exchange. It is based on Diffie-Hellman algorithm. Unlike other asymmetric algorithms that are based on factoring large prime numbers, it is based on calculating discrete logarithms in a finite field. Although, El Gamal provides the same type of functionality as most of the other asymmetric algorithms, its main drawback is its slow performance.
- ElGamal encryption consists of three components.
  1. Key generation
  2. Encryption algorithm
  3. Decryption algorithm

Key generation

1. Select a large random prime p and a generator g.
2. Generate a random integer x such that $1 \leq x \leq p-2$
3. Compute $y = g^x \bmod p$
   a. Public Key is (p, g, y)
   b. Private Key is x

Encryption

Given a message m such that $0 \leq m < p$, then user Bobby can encrypt m as below:

1. Pick an integer k between 1 and $p-2$
2. Compute the first component of the ciphertext $c_1 = g^k \bmod p$
3. Compute the second component of the ciphertext $c_2 = y^k * m \pmod p$
4. The ciphertext is the pair (c_1, c_2)

Decryption

- User Alice can decrypt (c_1, c_2) as original message $m = c_1^{p-1-x} * c_2 \pmod p$

Let's demonstrate the ElGamal algorithm with a practice question.

x. 3.7.1 : Use initial values of p = 7, g = 5 and x =2 and demonstrate ElGamal algorithm. Use other values as you prefer for your demonstration.

Soln. :

- Assume that the given values are for user Alex.

$$p = 7$$

$$g = 5$$

$$x = 2$$

- Calculate Alex's keys.

$$y = g^x \bmod p$$

$$y = 5^2 \bmod 7$$

$$y = 25 \bmod 7$$

$$y = 4$$

- Alex's Public Key = (p, g, y) = (7, 5, 4)
- Alex's Private Key = $x = 2$

Encrypt Message

- Similarly, assume that the user Bobby's Private Key is $k = 4$ and the message she wants to send is $m = 6$.
- She encrypts the message as

$$c_1 = g^k \bmod p$$

$$c_1 = 5^4 \bmod 7$$

$$c_1 = 625 \bmod 7$$

$$c_1 = 2$$

$$c_2 = y^k * m \pmod p$$

$$c_2 = 4^4 * 6 \pmod 7$$

$$c_2 = 1536 \bmod 7$$

$$c_2 = 3$$

- Hence, Bobby sends ciphertext $(c_1, c_2) = (2, 3)$ to Alex.

Decrypt Message

- Now, Alex wants to decrypt Bobby's message that she sent as ciphertext (2, 3)

$$m = c_1^{p-1-x} * c_2 \pmod p$$

$$m = 2^4 * 3 \pmod 7$$

$$m = 48 \bmod 7$$

$$m = 6$$

- Hence, Alex could get the encrypted message from Bobby and could use his private key to decrypt the m read it.

3.8 Concept Building – Information Accuracy

Q. What is authentication? SPPU – May 19 (May 19, 2 Marks)

- In chapter 2, you learnt about information secrecy – confidentiality. The focus of this chapter is information accuracy which is about ensuring the message integrity or message authentication.

Definition : *Message authentication is a process to ensure that the received message is exactly the same as it was sent.*

- What does this mean? This means that –
- The message has not been altered in anyway
  - No addition
  - No modification
  - No deletion
- The message is actually sent by the sender (proof of sender's identity).
- The order or the sequence of the messages is not changed.
- The messages are sent and received within an expected time frame.
- Consider a day to day scenario. You go to a shop to buy bread and in-exchange you give the shopkeeper a Rupees 100 currency note. The shopkeeper happily takes the note from you, examines it briefly, keeps it in the drawer and return you some change. You take the change, examine the returned currency briefly, slide it down in your wallet and move.
- What just happened? Why did the shopkeeper take the note you gave without any hesitation? Why did you take the change without any hesitation? What did the shopkeeper examine when you gave the note and why? What did you examine when you received the change and why? Are you telling me, *come on*, this is child simple? Are you trying to explain me the following?
  - There were two parties to the transaction – you and shopkeeper.
  - Shopkeeper wanted to accurately determine that the note you gave was genuine and not fake.
  - Shopkeeper looked at some of the known properties of the note (that were attached to the note itself) and verified it accurately.
  - You wanted to accurately determine that the change currency that you got was genuine and not fake.
  - You looked at some of the known properties of the note that you got and verified the note accurately.
- So, you understand that in this world, where the faith is based on the principle of "trust but verify", you need a mechanism to accurately determine the accuracy of the information that you get. Have you ever trusted a fake news or video clip and later laughed at yourself or felt bad about trusting the information without checking it genuinely?
- Don't you feel that there should be *someone* who could just tell you if the information you got was accurate or not? Did you really receive the email that you were sent, or someone modified it on its way? Did someone modify your file? If all these problems worry you, I request you not to worry, you have a way out. Please heartily welcome Cryptographic Hash functions!

3.9 Message Authentication Methods (Functions)

SPPU – May (May 19, 3 Mark

Q. Explain various methods of authentication.

At a high level, the message authentication can be performed using three mechanisms:

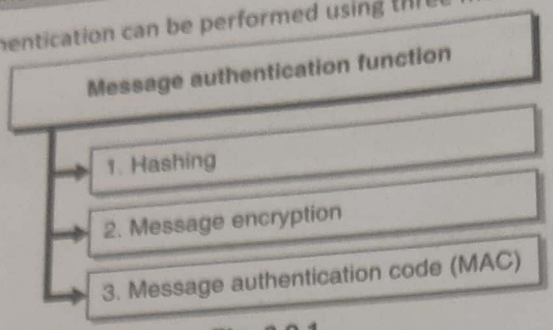


Fig. 3.9.1

1. Hashing

It deals with producing a unique hash value of the message that can be computed at the sender's and the receiver's end. If the hashes match at both the ends, the message is verified.

2. Message encryption

In this, the ciphertext of the entire message can be used to serve as its authenticator. The message is encrypted at the sender's and the receiver's end. If they both get the same ciphertext using the same key, it verifies the message. Note here that the focus is not to encrypt the message but to get the resulting ciphertext to serve as a way to verify the authenticity of the message.

3. Message Authentication Code (MAC)

MAC is very similar to hash. It uses a key to calculate the hash value of the message. The MAC is calculated at both the ends (sender's and receiver's) using the same key. If the MAC value matches at both the ends, the message is verified. You will learn about it in detail in the subsequent sections.

3.9.1 Cryptographic Hash Functions

SPPU – March 19(In Sem.)

Q. Explain in details the need and implementation of one way hash function (MD5). (March 19, 5 Marks)

As encryption provides message confidentiality, hashing provides message integrity and authentication.

Introduction

Definition : Hashing is the process of taking any length of input information and finding a unique fix length representation of that input information.

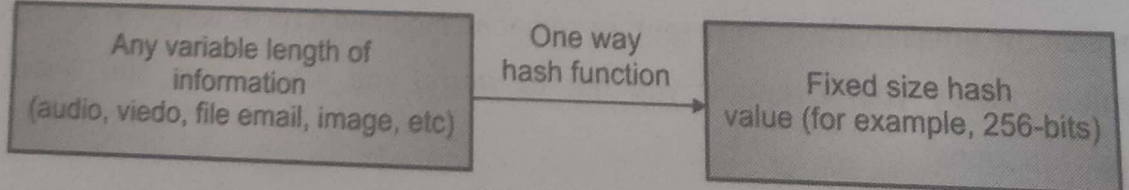


Fig. 3.9.2

- Hashing is the process of finding a unique message digest (or hash value) that corresponds to the input information. The length of input could be just one character or a huge video file. Hashing always produces a fixed size representation of the information.
- Here hashing does not make the information unreadable. It is not same as encryption. Hashing is just a way to attach some additional information to the original information that could later prove that the information is not modified. Remember our discussion from the concept building section? The shopkeeper can see Rupees 100 written clearly but verifies the currency note using other properties "attached" to it that proves its originality and authenticity.

A. How does this work?

- At the source of the information (it could be sender, website, company or anything else where the information is created) the hash value is calculated. This hash value along with the original information is sent to the receiver.

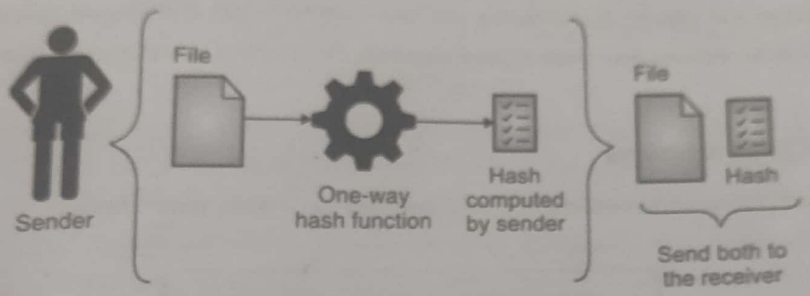


Fig. 3.9.3

- At the destination of the information (it could be a receiver, website, email program, or anything else where the information is received to process further), the hash value is calculated again and matched with the hash value that came with the information from source.
- If the two hash values (at source and at destination) match, the information is determined to be unmodified and is consumed. But, if the two hash values do not match, it proves that the information is altered and is often rejected.

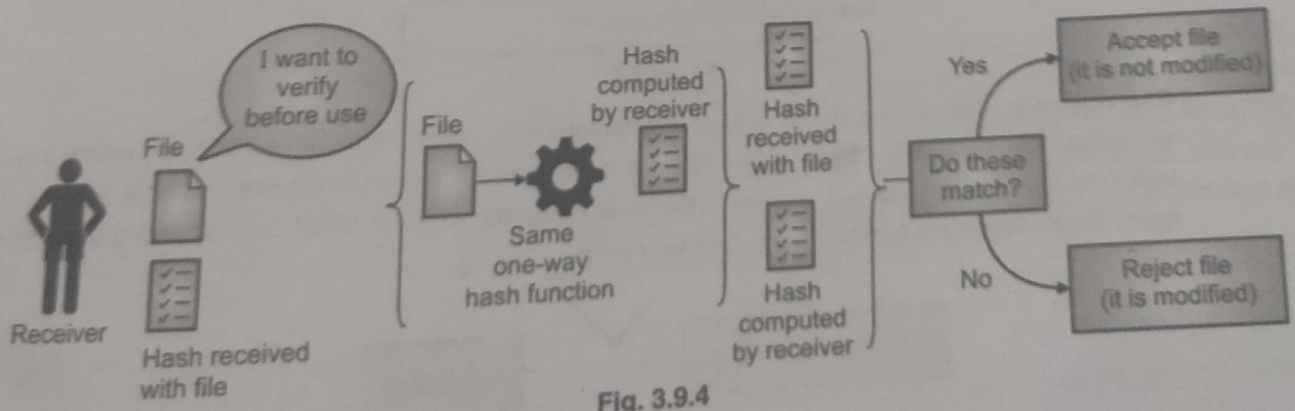


Fig. 3.9.4

- I want to again stress on the point that hashing is only for integrity checking of the information. The original information may or may not be encrypted. If the information is to be encrypted, following could be one of the sequences for integrity checking.

1. Create information
2. Calculate its hash value
3. Encrypt information
4. Send encrypted information and hash value
5. Receive encrypted information and hash value
6. Decrypt information
7. Calculate its hash value at the receiving end
8. Match source and destination hash
9. If matched, process information
10. If not matched, reject information

– Now, you might be thinking, yeah, this sounds good but what if someone captures the message and the has alters the message and creates and attaches the new hash to match the altered message and sends it to th receiver. How would receiver ever know? Great question, I must say. I would answer that later on. Let's park it a the moment.

B. Characteristics of hash functions

Unlike encryption, hash functions have some unique properties. Let's learn about them.

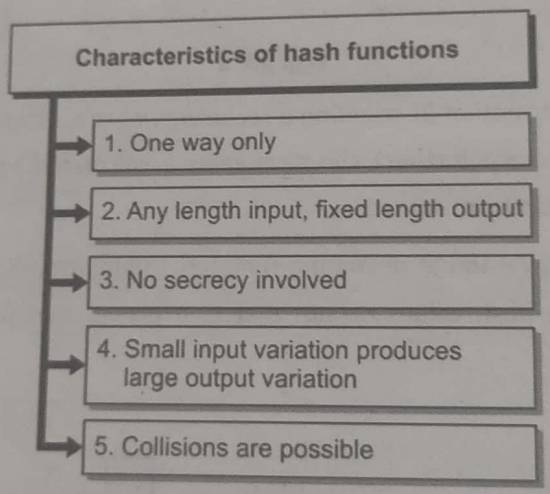


Fig. 3.9.5

1. **One way only :** This means that it is easy to calculate hash value from information and nearly impos to convert the hash value back to the original information. Understand it like this. It is easy to co milk into cheese, but can you convert cheese back to the original milk? Sounds weird and impo right? That's how hash functions are. One way only.

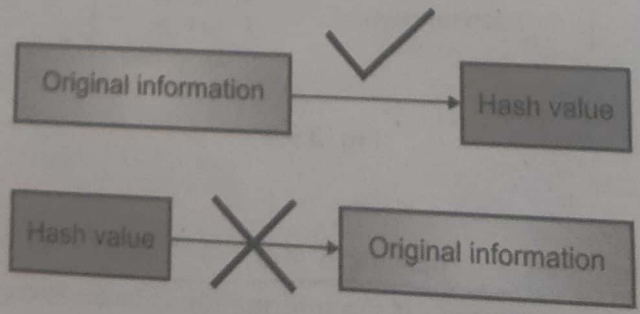


Fig. 3.9.6

2. **Any length input, fixed length output :** Hash functions can work on any length of input. It could work on a single character or a huge video file. It would always produce the same length output. Unlike encryption where the output size is more or less same as the input size, output from hashing process is only a unique representation of the original information and does not contain the information itself. Hence, the output length from hashing does not need to change with the length of input.

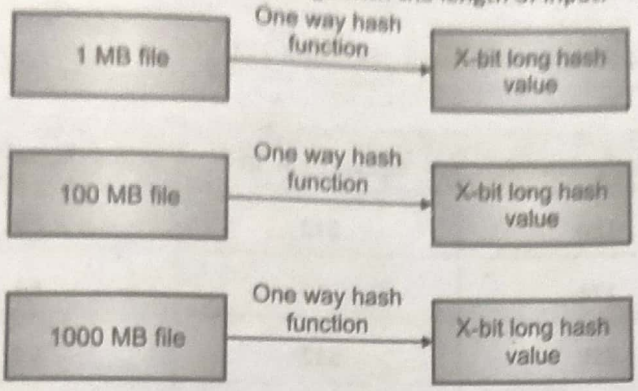


Fig. 3.9.7

3. **No secrecy involved :** Hashing process does not require a key and neither it requires any secret. The hashing algorithms are implemented such that they are only a one-way process producing a unique representation of the input information. Algorithms are publicly known as well.

4. **Small variation in input produces large variation in output :** It is nearly impossible to establish any relation between input and output in the hashing process. A small variation in the input totally changes the hash output. This is also called avalanche effect. Let's see an example. You can also try the following example at your end by calculating SHA-1 hash output using an online SHA-1 calculator such as http://www.sha1-online.com/.

   Original information - I love cybersecurity

   SHA-1 Hash value - 653da04906493b11d0735020db1f94360cac64f0

   Original information - U love cybersecurity

   SHA-1 Hash value - 8b9a7e5b4e5c8306c6ba678454e485356cb0aa24

   It does not matter which online calculator you use. You would get the same SHA-1 output for the given input.

5. **Collisions are possible :** While it is impossible to find original information from hash output, a collision condition is possible. Collision is a situation where two different inputs produce the same hash output. While this condition is extremely rare, some historical hashing algorithms such as MD2, MD4, MD5, SHA-1 have been shown to produce collision. These algorithms are no more used in the industry today. A hashing algorithm is considered strong if it provides high collision resistance. Newer hashing algorithms such as SHA-256 and SHA3-256 provide strong collision resistance.

C. Family of hashing algorithms

- National Institute of Standards and Technology (NIST) has specified the family of hashing algorithms that may or may not be fit for current use in the industry.

Hash Family Name	List of hashing algorithms	Approved for use?
SHA-1	**SHA-1**	**No**
SHA-2	SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256	Yes
SHA-3	**SHA3-224, SHA3-256, SHA3-384, SHA3-512**	**Yes**

Hashing Algorithm	Output size in bits	Block size in bits	Rounds	First published
SHA-1	**160**	**512**	**80**	**1995**
SHA-224	224	512	64	2004
SHA-256	**256**	**512**	**64**	**2001**
SHA-384	384	1024	80	2001
SHA-512	**512**	**1024**	**80**	**2001**
SHA-512/224	224	1024	80	2012
SHA-512/256	**256**	**1024**	**80**	**2012**
SHA3-224	224	1152	24	2015
SHA3-256	**256**	**1088**	**24**	**2015**
SHA3-384	384	832	24	2015
SHA3-512	**512**	**576**	**24**	**2015**

1. SHA-1

✍ ***Definition :*** *Secure Hash Algorithm 1 is a cryptographic hash function that produces 160-bit long ha value from a given input.*

- Collisions have been found in the algorithm and hence it is avoided in the industry wherever possible.
- The algorithm has two stages:

1. **Pre-processing :** Pre-processing involves padding a message, parsing the padded message into equal blocks and then setting initial values to be used for hash computation.
2. **Hash computation :** The hash computation generates a message schedule (sequence of processing) the padded message and uses that schedule along with various functions, constants and mathem operations to generate a series of intermediate hash values per round. The final hash is value is com after the last round and is 160-bit long.

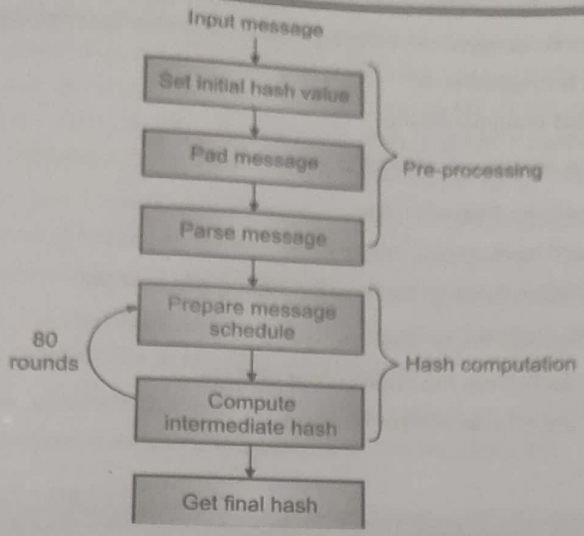


Fig. 3.9.8

2. SHA-3

Definition : *Secure Hash Algorithm 3 is the newest addition to the hash function family. It is comprised of various hash functions that compute secure hash values.*

- It is considered to be most secure against collision and is highly recommended to be used in the industry. SHA-3 is structurally quite different from earlier versions – SHA-1 and SHA-2. SHA-3 is a subset of the broader cryptographic primitive family KECCAK.
- KECCAK is the family of all sponge functions with a KECCAK-*p* permutation as the underlying function and multi-rate padding as the padding rule. The idea behind the sponge function is to "absorb" the information and "squeeze out" the hash value. The KECCAK-*p* permutations are specified with two parameters

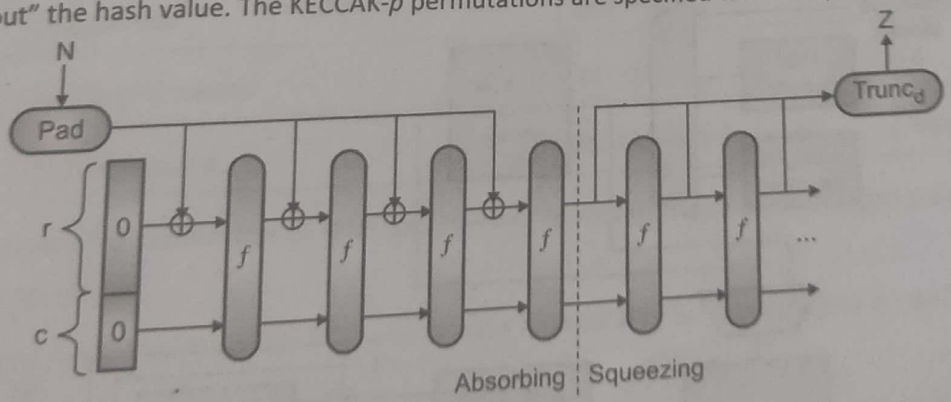


Fig. 3.9.9

1. The fixed length of the strings that are permuted.
2. The number of rounds.

- The sponge construction is a framework for specifying functions on binary data. It has three components.

1. Function *f* that works on fixed-length strings.
2. Parameter *r* that determines the rate of operation.
3. A padding rule denoted by *pad*.

- The sponge function can be denoted as *SPONGE[f, pad, r]*. A sponge function takes two inputs:
  1. A bit string that is denoted by *N*.
  2. The bit length that is denoted by *d*.
- So, the overall function is denoted as *SPONGE[f, pad, r](N, d)*.
- In the sponge function, Absorb Phase involves.
  1. XORing of message blocks into a subset of the state.
  2. Using the permutation function for transferring the whole state.
- In the sponge function, Squeeze Phase involves.
  1. Reading output blocks from the same subset of state.
  2. Replacing with state transformation function.

3. MD5

Definition : MD5 is a hashing algorithm.

MD5 was designed by Ronald Rivest in 1991.

A. Major attributes of MD5

- Block size – 512 bits
- Output size (hash size) – 128 bits
- Rounds – 4
- It is broken (collisions have been shown) and obsolete and is no more considered fit for cryptographic use.
- It is replaced by newer algorithms such as SHA-1 and SHA-2.

B. MD5 Algorithm Details

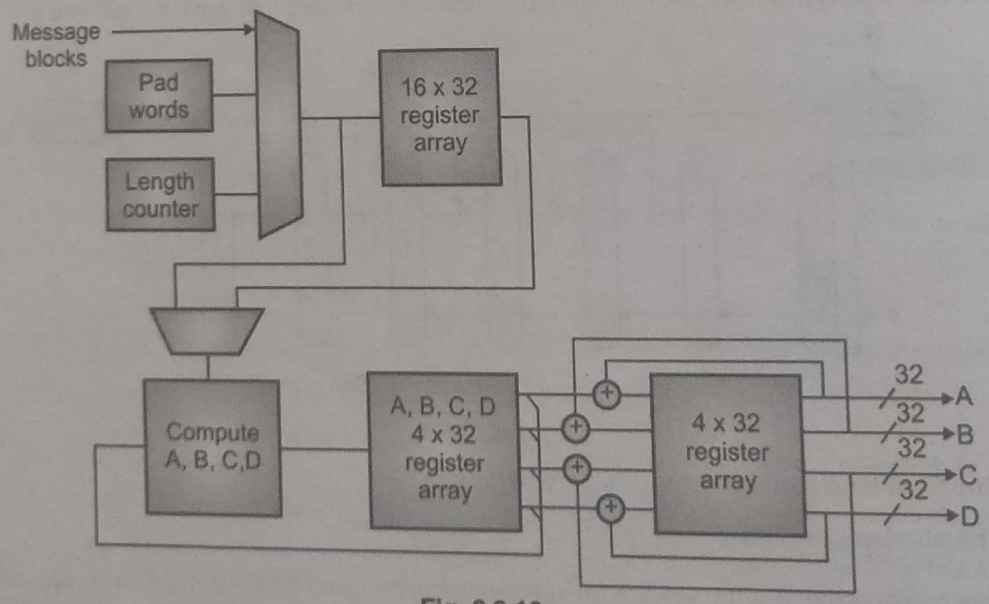


Fig. 3.9.10

MD5 algorithm specifies the following 5 steps for computing the message digest (or the hash value).

1. **Append Padding Bits :** The message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512.

That is, the message is extended so that it is just 64 bits shorter of being a multiple of 512 bits long. Padding is always performed, even if the length of the message is already congruent to 448, modulo 512. Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, at least one bit and at most 512 bits are appended.

2. **Append Length** : A 64-bit representation of the length, of the message before the padding bits were added, is appended to the result of the previous step. After this step, the resulting message length (after padding with bits and with length specification) is an exact multiple of 512 bits.

3. **Initialize MD Buffer** : A four-word buffer (A,B,C,D) is used to compute the message digest. Here each of A, B, C, D is a 32-bit register. These registers are initialized.

4. **Process Message in 16-Word Blocks** : At this stage, the four auxiliary functions take as input three 32-bit words and produce as output one 32-bit word. The functions are

   a. F(X,Y,Z) = X AND Y OR NOT(X) AND Z

   b. G(X,Y,Z) = X AND Z OR Y AND NOT(Z)

   c. H(X,Y,Z) = X XOR Y XOR Z

   d. I(X,Y,Z) = Y XOR (X OR NOT(Z))

5. **Output** : From these respective four functions, a message digest is produced and stored in the respective registers – A, B, C, D. The final output (message digest or the hash value) is given by concatenating (bringing together) these values.

Comparison between SHA and MD5

Sr. No.	Comparison Attribute	SHA	MD5
1.	Output bits	**160 – 512**	**128**
2.	Number of rounds	24 – 80	4
3.	Collision Found	**No (for SHA-2 above)**	**Yes**

- As you understand, the output bits of MD5 and number of rounds in the algorithm are quite less compared to SHA family of algorithms.
- Several collisions have been found for MD5 which make it unsuitable for modern data integrity requirements.

3.10 MAC (Message Authentication Code)

- Hash values provide integrity. But what if someone alters the message and adds a new hash? How do you know that you received the original message that the sender had intended to send? Hash values are computed on the original message and usually sent along with the message.
- Creating hash values does not require any keys. Hence, hash values can only provide integrity but cannot provide any additional assurance that the message is authentic.

Definition : *A Message Authentication Code (MAC) is a piece of information that can be ... authenticate a message.*

- MAC confirms that the message came from the stated sender and has not been changed. The MAC value ... both a message's data integrity as well as its authenticity. MAC is sometimes also called as *keyed hash*.

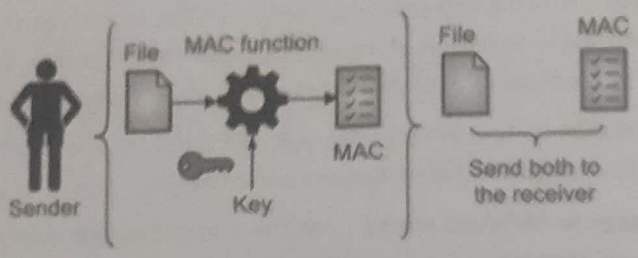


Fig. 3.10.1

- The sender intends to send a file (or a message). He computes the MAC value for the file using the pre-shared ... then forwards the message as well as the corresponding MAC value to the receiver.
- At the receiver's end, the MAC value is again computed using the same pre-shared key. This computed MAC ... then compared with the received MAC. If these two match, the file (or the message) is not altered ... authenticated.

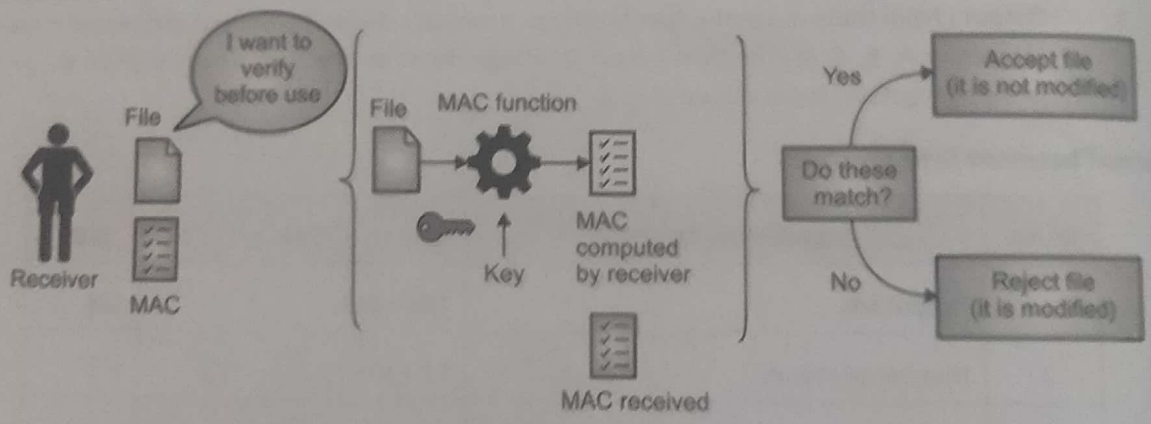


Fig. 3.10.2

- There are three types of MAC :

1. HMAC (Hash MAC)
2. CBC-MAC
3. CMAC (Cipher-based MAC)

- Let's learn about each of them.

1. HMAC

Definition : *In HMAC, a hashing function is used as a MAC function to calculate the MAC value.*

- The hashing function could be general hash functions such as MD5, SHA-1, or SHA-2.

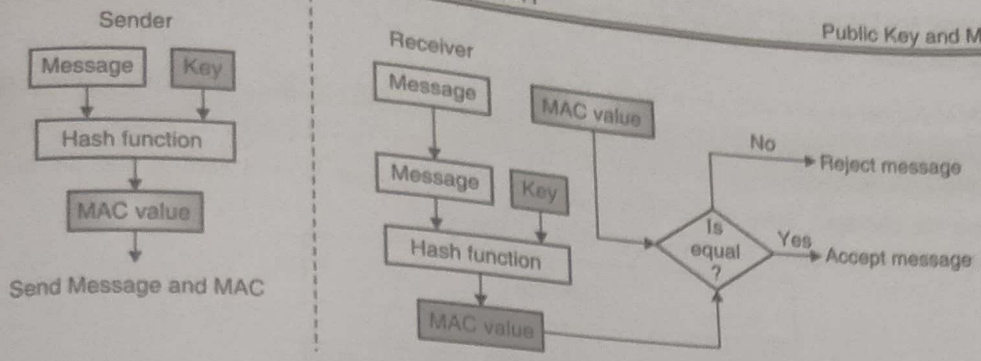


Fig. 3.10.3

- On the sender's side, the message and a pre-shared key is passed together into a hash function to compute a MAC. The computed MAC along with the original message is sent to the receiver. The key is not sent.
- Note here that the key is not used to encrypt the message. The key just provides a random value that when passed with the message to a hashing function generates a MAC value. The key has no other role except to provide a random value.
- On the receiver's side, the message is again passed through the same hashing function using the same pre-shared key. A MAC value is computed at the receiver's side and is matched with the MAC value that came from the sender. If the two MAC values match, the message is accepted else the message is rejected.

2. CBC-MAC

Definition : *In CBC-MAC, a symmetric block cipher encryption function in the CBC mode is used as the MAC function to calculate the MAC value.*

- The message is encrypted with a symmetric block cipher in the CBC mode, and the output of the final block of ciphertext is used as the MAC. The sender does not send the encrypted version of the message, but instead sends the plaintext version and the computed MAC.

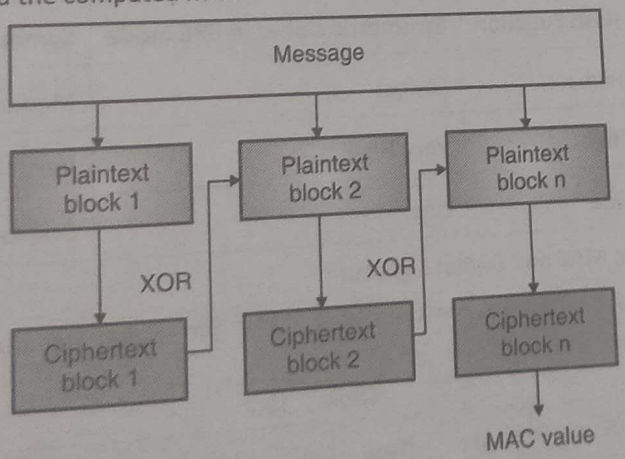


Fig. 3.10.4

- The receiver receives the plaintext message and encrypts it with the same symmetric block cipher in CBC mode and calculates a MAC value at her end. If the two MAC values (one received and one computed) match, the message is accepted else it is rejected. This method does not use a hashing algorithm as in HMAC.

3. CMAC

- CMAC (Cipher-based MAC) is very similar to CBC-MAC.

Definition : In CMAC, a symmetric block cipher encryption function is used as the MAC function calculate the MAC value.

- Here like CBC-MAC, a symmetric block cipher encryption function is used here as well but not in CBC m That's the major difference between CBC-MAC and CMAC.

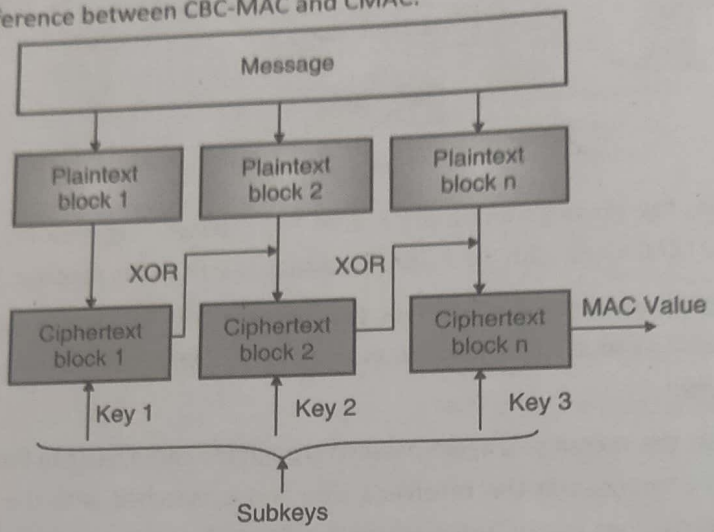


Fig. 3.10.5

- CMAC is typically calculated using AES-128 algorithm and provides strongest form of message authentication a integrity. It is also called as One-Key MAC or OMAC.

a. Comparison between HMAC, CBC-MAC and CMAC

Table 3.10.1 summarizes the key differences between HMAC, CBC-MAC and CMAC.

Table 3.10.1

Comparison Attribute	HMAC	CBC-MAC	CMAC
MAC generation function	**Hash Function**	**Symmetric cipher in CBC mode**	**Symmetric cipher**
Speed of MAC generation	Highest	Lowest	Medium
Strength of MAC	**High**	**Very High**	**Very High**
Number of Keys used	One	One	One key divided into multiple sub-keys

b. Comparison between Hash, MAC and Digital Signature

Table 3.10.2 summarizes the difference between Hash, MAC and Digital Signatures. Your understanding of thes differences is crucial.

Table 3.10.2

Comparison Attribute	Hash	MAC	Digital Signature
Integrity	**Yes**	**Yes**	**Yes**
Authentication	No	Yes	Yes
Non-repudiation	**No**	**No**	**Yes**
Keys Used	None	Symmetric Keys	Asymmetric Keys

c. **Security of Hash Functions and MAC**

– Hash functions and MAC need to be secure by themselves to be useful. You cannot rely on these message authentication mechanisms if it is easy to "break" them. Let's understand some of the desired security properties of these mechanisms and possible attacks on them.

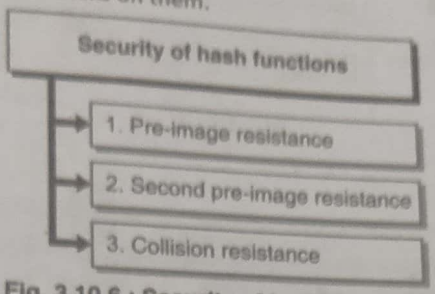


Fig. 3.10.6 : Security of hash functions

– The three desired security properties of hash functions are as following.

1. **Pre-image resistance :** This property ensures the *one-way only* security criteria of hash functions. It states that for a given hash value *h*, it should be impossible to find any message *m* such that *h = hash (m)*. So, in a nutshell, you cannot find the original message from its hash value.
2. **Second pre-image resistance :** This property states that given a message m_1, it is hard to find another message m_2, such that hash(m_1) = hash(m_2). So, looking at a message it should be hard to find another message which could produce the same hash. This property is also called weak collision resistance.
3. **Collision resistance :** This property states that it should be hard to find any two messages pairs m_1 and m_2 such that hash(m_1) = hash(m_2). This property is also called strong collision resistance.

d. **Attacks on hash functions and MAC**

There are two possible ways to attack hash functions and MAC.

1. **Brute-force attack :** In this, the attacker repeatedly tries to find various combinations of messages so as to produce a collision. The brute-force attack is more difficult to carry out on MAC than hash functions.
2. **Cryptanalysis :** Similar to finding weakness in the encryption algorithms, the attackers tries to find a weakness in the hash functions and exploit that weakness to carry out further attacks.

3.11 Digital Signature

Let's answer a question that I asked you before – what if someone captures the information and the hash and alters both. How would you know? The answer is digital signature. Let's learn about it.

Definition :A digital signature is a hash value that has been encrypted with the sender's private key. The act of signing means encrypting the message's hash value with a private key (since no one else knows the sender's private key).

3.11.1 How does this work?

– So, the sender computes and encrypts the hash value with her private key. At the receiving end, you decrypt the hash value with the sender's public key. Now, because no one else knows the private key of the sender, altering hash value and re-signing with the private key of the sender is not possible.

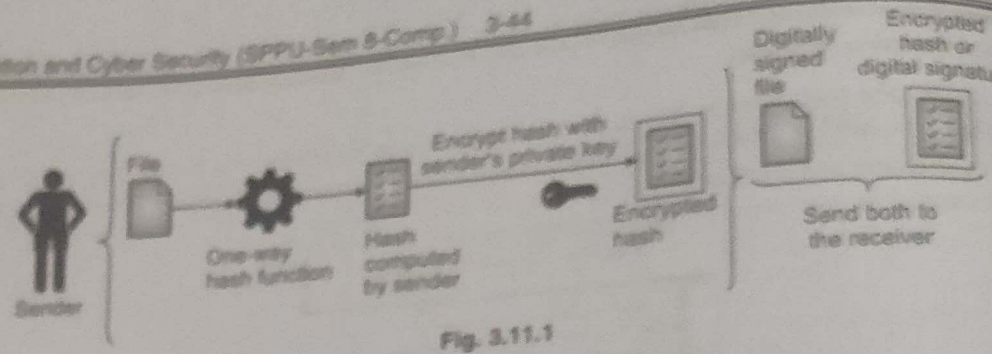


Fig. 3.11.1

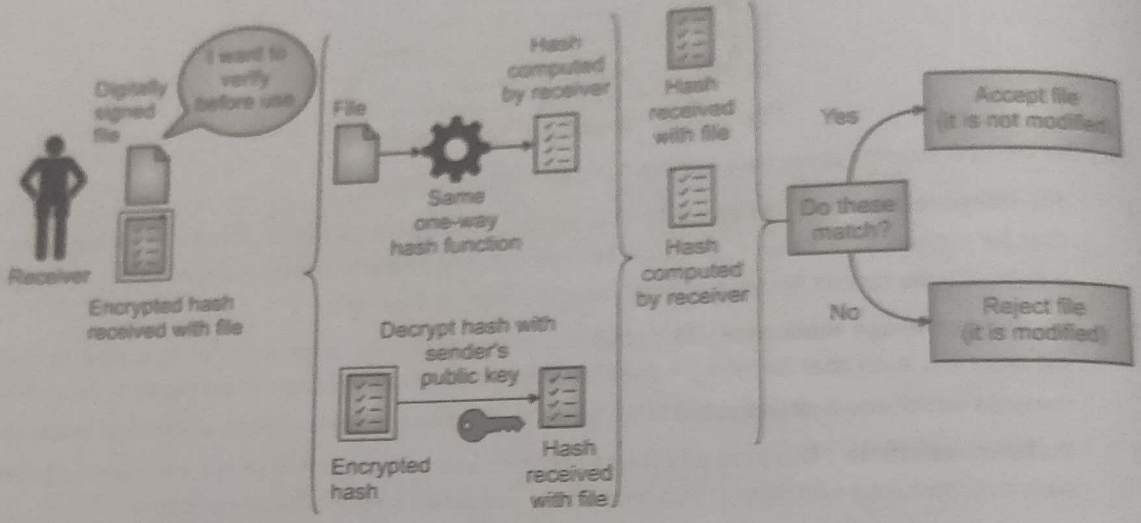


Fig. 3.11.2

Processing applied on a message	Security Property achieved
Encryption	Confidentiality
Hashing	Integrity
Digitally Signing	Integrity, authentication, non-repudiation
Encryption and digitally signing	Confidentiality, Integrity, authentication, non-repudiation

Application and use of digital signature

1. Sending and receiving secure emails
2. Signing documents. For example, you can sign income tax returns using digital signature
3. Sending and receiving important files. For example, insurance policy documents, Aadhar card e-letter, etc.

3.11.2 Properties of Digital Signature

Digital signature provides three security properties

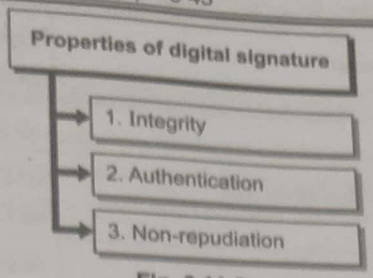


Fig. 3.11.3

1. **Integrity :** via hash value calculation
2. **Authentication :** via the ability to prove sender's identity by decrypting hash with the sender's known public key
3. **Non-repudiation :** Sender cannot deny sending the message because she used her private key to encrypt the hash

3.11.3 X.509 Certificate

- X.509 is a standard that defines the requirements for public key certificates.

> ✍ ***Definition :*** *A certificate is a signed data structure that binds a public key to a person, computer, or organization.*

- The certificate uniquely identifies the owner of a public key. Certificates are issued by Certification Authorities (CAs) such as Verisign, Global Sign etc. In a secure communication, certificates are used to identify the systems and establish trust amongst the communicating parties.
- Since its inception in 1998, three versions of the X.509 public key certificate standard have evolved. The most commonly used version of X.509 is version 3. X.509 certificates are used for secure communication such as establishing a https connection.

Contents of a X.509 version 3 certificate		
Fields	Purpose	Example
Version	Identifies the version of the certificate	V3
Serial Number	Unique number for the certificate	79ad16a14aa0a5ad4c7358f407132e65
Signature algorithm	Algorithm used to create digital signature for certificate	sha1RSA
Signature hash algorithm	Algorithm used to calculate hash of certificate	sha1
Issuer	Name of certificate issuer	CN = Microsoft Root Certificate Authority DC = microsoft DC = com
Valid from	Date from which certificate is valid	Thursday, May 10, 2001 4:49:22 AM
Valid to	Date until which certificate is valid	Monday, May 10, 2021 4:58:13 AM

Subject	Name of certificate owner	CN = Microsoft Root Certificate Auth... DC = microsoft DC = com
Public key	Public key	a5 94 ef 15 14 89 fd 4b 73 ... (4096 bits)
Issuer unique ID	ID of issuing Certificate Authority (CA)	03475573948593hgfyuerwe327e7e525... ae3
Subject unique ID	ID of subject	0eac826040562797e52513fc2ae10a539... 4a4
Extensions	Optional Information	Thumbprint, Friendly Name, Key Usage, ...

Note : If you use Microsoft® Windows® OS, you can go to (Windows Key + R) Run and type certmgr.msc. It would open certificate manager where you can browse various certificates stored on your system. Double-click on any cert... and examine the various fields it has.

3.11.4 Digital Signature Schemes

There are various schemes to create and verify digital signatures. These schemes are developed and implemented using various encryption and hashing algorithms. Let's learn about a few of them.

1. RSA Digital Signature Scheme

Definition :RSA signatures are based on public key cryptography.

RSA uses public key cryptography for creating and verifying digital signatures.

Key Generation

RSA digital signatures work on public and private key pairs. They can be generated by the regular key generating method by a Certificate Authority (CA) or on the user's system by herself. Recall from your reading RSA algorithm that the keys are as following:

- The public key = *(n, e)*
- The private key = *(n, d)*

Message Signing

To sign a message, M

- Calculate the hash value of the message M at sender's end
- *h = hash(M)*
- Encrypt *h* using RSA private key
- Signature $S = (h)^d \bmod n$

Signature Verification

- Decrypt Signature *S* using public key
- $h' = (S)^e \bmod n$
- Calculate the hash value of the message M at receiver's end
- *h = hash(M)*
- If $h = h'$, the signature is valid else the signature is invalid

2. Schnorr Digital Signature Scheme

Definition :Schnorr signatures are based on discrete logarithm problems.

Schnorr signatures were developed by Claus P Schnorr and subsequently protected by U.S. Patent until late 2008. As a result of the patent, Schnorr signatures have not been standardized or widely used in crypto libraries today. It is believed to be a more elegant signature solution with a simple mathematical proof.

Key Generation

- Choose a private signing key, x
- The public verification key is $y = g^x$ where g is a generator point.

Message Signing

To sign a message, M

- Choose a random value k
- Let $r = g^k$
- Let e = Hash (r|| M)
- Let s = k – xe

The signature is the pair (s, e).

Signature Verification

- Let $r_v = g^s y^e$
- Let e_v = Hash(r_v|| M)

If $e_v = e$, then the signature is verified.

3. ElGamal Digital Signature Scheme

Definition : The ElGamal digital signature scheme is based on the difficulty of computing discrete logarithms.

- It was conceived by Taher ElGamal in 1984. It is not used widely in the industry today.
- Suppose a message m need to be signed. Following are the set of steps in the scheme.

Key Generation

- Choose a large prime p and a primitive root α.
- Choose a secret integer z and calculates $\beta \equiv \alpha z \pmod p$.
- The values of p, α and β are made public and z is kept private.

Message Signing

- Select a secret random integer k such that $GCD(k, p-1) = 1$.
- Compute $r \equiv \alpha^k \pmod p$.
- Compute $s \equiv k^{-1}(m - zr) \pmod{p-1}$.
- The signed message is the triplet (m, r, s).

Signature Verification

- Compute $v_1 \equiv \beta^r r^s \pmod p$ and $v_2 \equiv \alpha^m \pmod p$.

- The signature is declared valid if $v_1 \equiv v_2 \pmod{p}$.

3.11.5 Digital Signature Standard (DSS)

Definition : National Institute of Standards and Technology (NIST) defined the Digital Signature Standard to provide approved techniques for generating and validating digital signatures for authenticating messages (or any binary data in general).

- It approved three techniques as part of the standard.

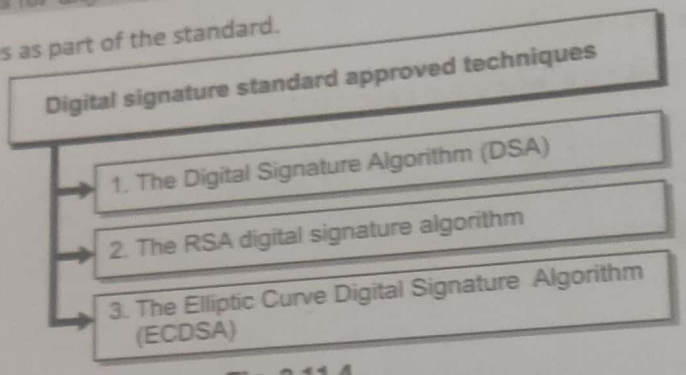


Fig. 3.11.4

- Out of the three currently listed techniques, DSA was the primary and the original proposal in the standard. Let's learn about it in brief.

3.11.6 Digital Signature Algorithm (DSA)

Definition :The Digital Signature Algorithm (DSA) is based on the difficulty of computing discrete logarithms.

It is based on ElGamal and Schnorr digital signature schemes.

Key Generation

- p is a prime number
- q is a prime divisor of $(p - 1)$
- $g = h^{(p-1)/q} \bmod p$ where h is any integer with $1 < h < (p - 1)$
- x is user's private key
- y is user's public key

Message Signing

- M is message to be signed
- H(M) = SHA1(M)
- $r = (g^k \bmod p) \bmod q$
- $s = [k^{-1} (H(M) + xr)] \bmod q$
- Signature = (r, s)

Signature Verification

- Assume M', r', s' are as received at the receiver's end
- $w = (s')^{-1} \bmod q$
- $u_1 = [H(M')w] \bmod q$

- $u_2 = (r')w \bmod q$
- $v = [(g^{u1} y^{u2}) \bmod p] \bmod q$
- v should match r'

3.12 Kerberos

Q. What is Kerberos? Explain operation in detail. SPPU - March 19(In Sem.) (March 19, 5 Marks)

Definition : *Kerberos is a network-based authentication protocol.*

It was developed around mid-1980s at MIT. It works on the client/server model and uses symmetric key cryptography. Today, Kerberos is extensively used for authentication in Microsoft Windows, Unix, Linux and Apple OS.

A. Problems addressed by Kerberos

1. **Sharing passwords over the network :** With Kerberos, you need not share passwords over the network for authentication.
2. **Establishing trust between two parties :** Kerberos acts as a third party and helps to establish trust between two non-trusting parties.
3. **Difficult to spoof authentication :** Kerberos employs several mechanisms to secure the authentication process and makes it difficult to spoof authentication.
4. **Scalable :** Kerberos supports a large number of servers and clients and hence is suitable for distributed network architectures.

B. Components of Kerberos

The Kerberos environment has several components that are required for functioning.

1. **Key Distribution Center (KDC) :** KDC is the most important component in the Kerberos environment. It holds all the information required for the functionality of Kerberos. Basically, it consists of the following sub-components.
   a. **Authentication Service (AS) :** The Authentication Service (AS) issues Ticket-Granting Tickets (TGTs) that is used to connect to the Ticket Granting Service (TGS). It also verifies principals that require authentication.
   b. **Ticket Granting Service (TGS) :** The TGS issues the tickets to the clients using which the clients can connect to the desired server.
   c. **Principals :** In the Kerberos terminology, Principals can be users, clients, servers, applications or network services that require authentication. The KDC has the information about each principal account and its secret key. For example, a user could be a principal requiring access to a print server that could be another principal.
   d. **KDC Database :** All the principal related information is stored in the KDC database.
   e. **Realm :** In Kerberos terminology, a realm is the set of principals who can authenticate to each other. It is a logical grouping of principals. One KDC can have one realm or several realms.
2. **Client :** Typically, a client is the principal that requires to authenticate to another principal (server).
3. **Server :** Typically, a server is the principal that holds resources that the client is interested in and provides the resources that can be consumed after successful authentication.

C. How does it work?

Kerberos heavily uses the concept of tickets that works very much like your train ticket. A ticket is just a temp proof. Let's understand the working in detail.

Scenario 1 – User authenticating to a computer

1. The user enters the username and password on the computer.
2. The Kerberos software running on the computer sends the username to the Authentication Service (AS).
3. The Authentication Service checks if the username is present. If yes, it sends back a Ticket-Granting Ticket (T which is encrypted with the user's pre-shared secret key (password).
4. If the user entered the correct password, she can decrypt the TGT and then is granted access to the computer

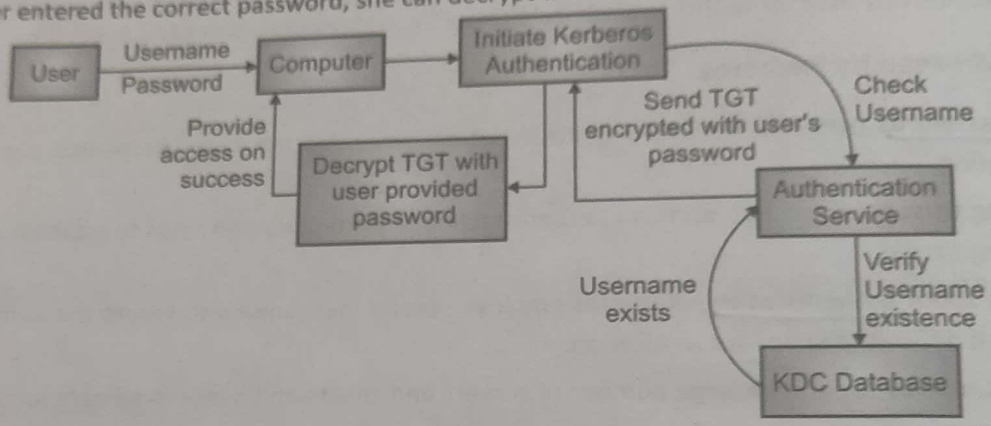


Fig. 3.12.1

Scenario 2 – Authenticated User now wants to print a document

From Scenario 1, the user has successfully authenticated to her computer. Now, suppose that she wants t document and hence needs to authenticate to the print server.

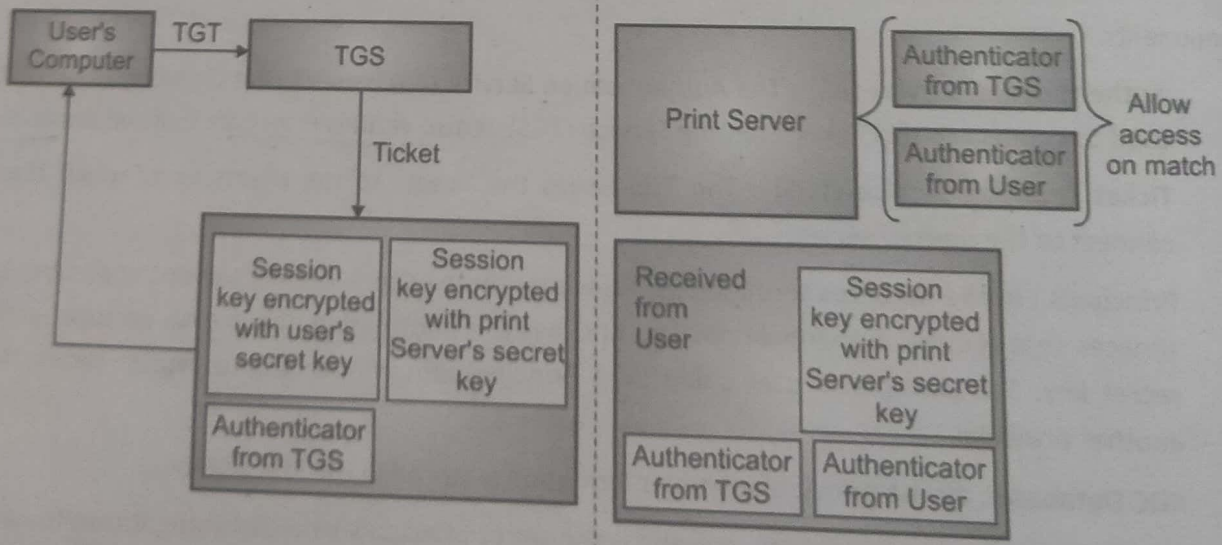


Fig. 3.12.2

The following steps are taken:

1. The computer sends the TGT it received earlier (when the user wanted to authenticate) to the TGS. The TGT is a proof for the TGS that the user has already been successfully authenticated (as in scenario 1).
2. TGS creates a new ticket and puts two copies of the same session key (temporary secret key) in it. It encrypts the first copy of the session key with the user's secret key and the second copy with print server's secret key. This ticket also contains authenticator information that holds the value of the user's computer's IP Address, sequence number and timestamp from where the TGT came. It then sends the ticket to the user's computer.
3. The user decrypts the ticket created by the TGS with her secret key and obtains the session key. It adds another set of authenticator information (computer's IP Address, sequence number and timestamp) to it and sends the ticket to the print server.
4. The print server receives the ticket and extracts the session key by decrypting it. It knows that the KDC created the ticket because only KDC had the knowledge about the print server's secret key. It also gets the two authenticators (one from TGS and one from user) that uniquely identify the user's computer. If the two authenticators match, the user is successfully authenticated, and the print server prints the user's document.

D. Limitations of Kerberos

1. KDC can be a single point of failure. If KDC fails, no authentication can take place.
2. Kerberos depends on the accuracy of time. So, all clients and servers should be time synchronized.
3. Session keys are stored on user's computer. So, if the computer is breached, the sessions key might be stolen.

3.13 Needham Schroeder Authentication Protocol

Needham Schroeder proposed two authentication protocols – one using symmetric keys and one using asymmetric keys. Let's learn about both of them.

3.13.1 The Needham–Schroeder Symmetric Key Based Authentication Protocol

- In symmetric key based authentication protocol, there are 3 entities
  - 2 users – let's call them Alice (A) and Bob (B)
  - 1 Server (S)

✍ ***Definition :*** *The goal of this protocol to generate and share a key that can be used for securing communication between the two users - A and B.*

- Here the Needham–Schroeder Symmetric Key Based Authentication Protocol forms a basis for Kerberos based authentication. It solves the key distribution problem.
- Assume the following primitives:
  - A and B are identities of Alice and Bob respectively.
  - K_{AS} is a symmetric key known only to A and S.
  - K_{BS} is a symmetric key known only to B and S.
  - N_A and N_B are nonce (random number used once) generated by A and B respectively.
  - K_{AB} is the symmetric key that needs to be generated and shared between A and B for secure communication.

A. Protocol Operation

Following is the sequence of activities that the protocol follows.

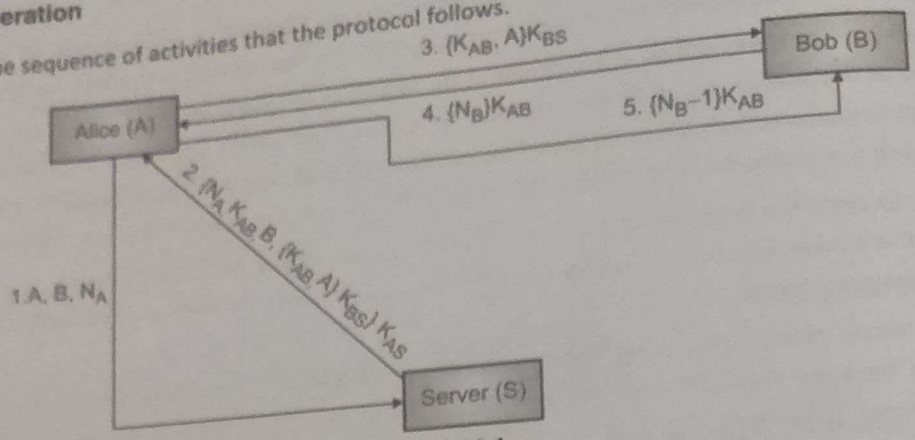


Fig. 3.13.1

1. Alice sends a message to the server identifying herself and telling the server that she wants to communicate with Bob.
2. The server generates K_{AB} and sends it to Alice encrypting the entire response with K_{AS}.
   a. A copy of K_{AB} is encrypted using K_{BS} for Alice to forward to Bob and also identify herself and
   b. A copy for Alice herself.
3. Alice forwards the key K_{AB} to Bob. Bob decrypts it with the key K_{BS}.
4. Bob sends Alice a nonce N_B encrypted using K_{AB} to show that he has the key.
5. Alice performs a simple operation on the nonce N_B, re-encrypts it and sends it back verifying that she is still alive and that she holds the key as well.

B. Attack on the protocol

This protocol is vulnerable to replay attack. The attacker can grab the older and compromised value for K_{AB}. Hence then replay the message $\{K_{AB}, A\}K_{BS}$ to Bob, who will accept it. Kerberos solves this problem by adding timestamp to avoid replaying older communication.

3.13.2 The Needham–Schroeder Asymmetric Key Based Authentication Protocol

- In asymmetric key based authentication protocol, there are 3 entities as well
  - 2 users – let's call them Alice (A) and Bob (B)
  - 1 Server (S)
- The goal of this protocol is to share the respective public keys between the two users - A and B.

Assume the following primitives :

- 3 key pairs → P stands for Public Key, Q stands for Private Key.
  - K_{PA} and K_{QA} → Public and Private Keys of A respectively.
  - K_{PB} and K_{QB} → Public and Private Keys of B respectively.
  - K_{PS} and K_{QS} → Public and Private Keys of S respectively.
- K_{PS} is known to both A and B and is trusted.

A. Protocol Operation

Following is the sequence of activities that the protocol follows.

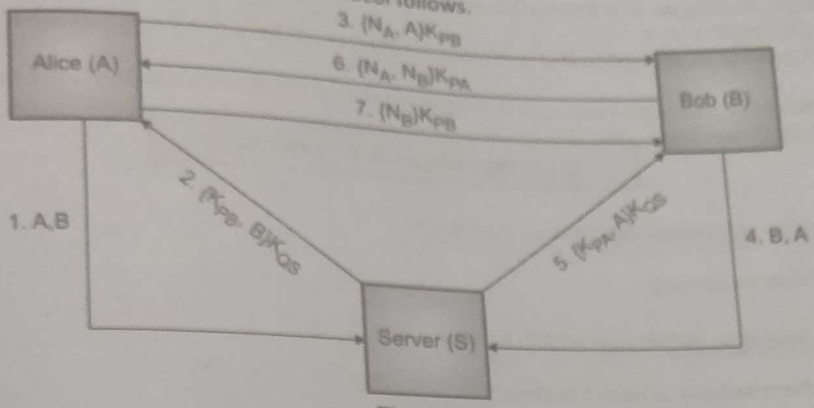


Fig. 3.13.2

1. A requests B's public keys from S
2. S responds with B's public key K_{PB} alongside B's identity, encrypted by the server's private key K_{QS}
3. A chooses a random nonce N_A and sends it to B by encrypting it using B's public Key K_{PB} received in step 2
4. B now knows that A wants to communicate. So, B requests A's public keys from S.
5. S responds with A's public key K_{PA} alongside A's identity, encrypted by the server's private key K_{QS}
6. B chooses a random nonce N_B and sends it to A along with N_A to prove his ability to decrypt with K_{QB}
7. A confirms N_B to B, to prove her ability to decrypt with K_{QA}

B. Attack on the protocol

- This protocol is vulnerable to Man-in-the-Middle attack. The attacker can make A and B believe that they are communicating.
- This could be fixed by updating the step 6 by passing along the identity - $\{N_A, N_B, B\}K_{PA}$. So, A would know who she is actually communicating with instead of being attacked by the middle-man.

Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

Public Key Cryptography

Q. 1 List the principles of public key cryptosystems. **[6 Marks]**

Q. 2 Take an example of your choice and work it through explaining RSA. **[8 Marks]**

Q. 3 Explain the various attacks on RSA. **[6 Marks]**

Q. 4 Write a short note on Diffie-Hellman Key Exchange Algorithm. **[4 Marks]**

Q. 5 Take an example of your choice and work it through explaining Diffie-Hellman Key Exchange Algorithm. **[8 Marks]**

Q. 6 With a suitable diagram, explain how Elliptic Curve Cryptography works.

Q. 7 List the usage, advantages and disadvantages of Elliptic Curve ArithmeticCryptography.

Q. 8 Explain key generation, encryption and decryption function in ElGamal Curve Cryptography.

Q. 9 Take an example of your choice and work it through explaining ElGamal Curve Cryptography.

Cryptographic Hash Functions

Q. 10 Give an overview of Message Authentication Methods.

Q. 11 Write a short note on hashing.

Q. 12 With suitable diagrams, explain how a hash function works.

Q. 13 Describe the characteristics of hash functions.

Q. 14 Explain SHA1.

Q. 15 Explain SHA3.

Q. 16 Explain MD5.

Q. 17 Compare SHA and MD5. Which one would you suggest using and why?

Q. 18 Explain Security of hash functions and possible attacks on them.

Message Authentication Code

Q. 19 Write a short note on MAC (Message Authentication Code).

Q. 20 With suitable diagrams, explain how MAC (Message Authentication Code) works.

Q. 21 Describe the various types of MAC (Message Authentication Code).

Q. 22 With a suitable diagram, explain HMAC.

Q. 23 With a suitable diagram, explain CBC-MAC.

Q. 24 With a suitable diagram, explain CMAC.

Q. 25 Compare HMAC, CBC-MAC and CMAC.

Q. 26 Compare Hash, MAC and Digital Signature.

Digital Signature

Q. 27 Write a short note on digital signature and also list its usage.

Q. 28 With suitable diagrams, explain how digital signature works.

Q. 29 List the properties of digital signature.

Q. 30 Write a short note on X.509.

Q. 31 Describe the format of a X.509 Certificate. [6 Marks]

Q. 32 List the steps for key generation, message signing and signature verification in RSA Digital Signature Scheme. [8 Marks]

Q. 33 List the steps for key generation, message signing and signature verification in Schnorr Digital Signature Scheme. [8 Marks]

Q. 34 List the steps for key generation, message signing and signature verification in ElGamal Digital Signature Scheme. [8 Marks]

Q. 35 List the steps for key generation, message signing and signature verification in Digital Signature Algorithm (DSA). [8 Marks]

Kerberos

Q. 36 Write a short note on Kerberos. [4 Marks]

Q. 37 List the problems addressed by Kerberos. [4 Marks]

Q. 38 Describe the core components of Kerberos. [6 Marks]

Q. 39 With suitable diagrams explain how Kerberos works. [8 Marks]

Q. 40 Explain the limitations of Kerberos. [4 Marks]

Authentication Protocol

Q. 41 Explain the Needham–Schroeder Symmetric Key Based Authentication Protocol. [8 Marks]

Q. 42 Explain the Needham–Schroeder Asymmetric Key Based Authentication Protocol. [8 Marks]

❑❑❑