SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NAAC-UGC with 'A' Grade

Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

Course Code and Name : 19IT602– CRYPTOGRAPHY AND CYBER SECURITY

III YEAR / VI SEMESTER

Unit 3: ASYMMETRIC KEY CRYPTOGRAPHY

Topic : Euler Totient Function

Fermat's Theorem

- Useful in public key and primality testing
- If P is Prime Number & a is a positive integer not divisible by p then
    - $a^{p-1} = 1 \pmod p$
    - where p is prime and gcd(a,p)=1
- Also known as Fermat's Little Theorem
- Also have: $a^p = a \pmod p$

Euler Totient Function ø(n)

- Euler Totient Function ø(n) = pq
  - ø(n) = pq =ø(p)*ø(q) = (p-1) *(q-1)
- Where p and q are two prime numbers; p≠q

- **Example**
  - ø(21) = (3–1)x(7–1) = 2x6 = 12

Euler Totient Function ø(n):some Values

n	$\phi(n)$
1	1
2	1
3	2
4	2
5	4
6	2
7	6
8	4
9	6
10	4

n	$\phi(n)$
11	10
12	4
13	12
14	6
15	8
16	8
17	16
18	6
19	18
20	8

n	$\phi(n)$
21	12
22	10
23	22
24	8
25	20
26	12
27	18
28	12
29	28
30	8

Euler's Theorem

- Every a and n that are relatively prime:
  - $a^{ø(n)} \equiv 1 (\text{mod } n)$
- **Example**

  a=3;n=10; ø(10)=4;

  hence 34 = 81 = 1 mod 10

  a=2;n=11; ø(11)=10;

  hence 210 = 1024 = 1 mod 11

Testing for Primality

- often need to find large prime numbers
- traditionally sieve using trial division
  - ie. divide by all numbers (primes) in turn less than the square root of the number
  - only works for small numbers
- alternatively can use statistical primality tests based on properties of primes
  - for which all primes numbers satisfy property
  - but some composite numbers, called pseudo-primes, also satisfy the property
- can use a slower deterministic Primality test

Miller Rabin Algorithm

a test based on prime properties that result from Fermat's Theorem algorithm is:

TEST (n) is:

1. Find integers k, q, k > 0, q odd, so that $(n–1)=2^k q$
2. Select a random integer a, $1<a<n–1$
3. if a^q mod n = 1 then return ("inconclusive");
4. for j = 0 to k – 1 do
   5. if (a^{2jq} mod n = n-1)
      then return("inconclusive")
6. return ("composite")

Thank You