Computer Network Lab

Exp 11: RSA Program

Dr. Ram P Rustagi
Sem V (2018-H2)
Dept of CSE, KSIT
rprustagi@ksit.edu.in

Ex11 Resources

- Example: Java programs
  -

Exp11 Description

- Program 11 (Java)
  - Write a program for simple RSA algorithm to encrypt and decrypt the data
  -

Fermat's Theorem

- For positive integer a, and prime p
  - $a^{p-1} \cong 1 \pmod{p}$
- Proof
  - Consider set [1,2,…. P-1]
  - Multiply each element by `a mod p`

    `X=[a, 2a, …, (p-1)a]`

    None of the elements are same
  - Multiply elements of both sets

    `a.2a….(p-1)a` **mod** `p` **=** `1.2….(p-1)` **mod** `p`

    => $a^{p-1}(p-1)! \bmod p = (p-1)! \bmod p$

    => $a^{p-1} = 1 \bmod p$

Euler's Theorem

- Euler Totient function Φ(n)
  - Defined as number of positive integers less than n and relatively prime to n
  - Examples
    - `Φ(37) = 36`
    - `Φ(35) = 24`
      - `1,2,3,4,6,8,9,11,12,13,16,17,18,`
      - `19,22,23,24,26,27,29,31,32,33,34`
  - For prime number p
    - `Φ(p) = p-1`

Euler's Theorem

- For every a, n relatively prime to each other
  - $a^{\Phi(n)} = 1 \bmod n$
- Proof:
  - If n is prime, it holds by Fermat's theorem
  - Consider set of integers corresponding to $\Phi(n)$
    - $R = [x_1, x_2, \ldots, x_{\Phi(n)}]$
    - For each x_i, $\gcd(x_i, n) = 1$
  - Multiply each element by $a \bmod n$
  - Each element is still unique i.e.
    - $ax_i \bmod n \neq ax_j \bmod n$

Euler's Theorem

$$\prod_{i=1}^{\phi(n)} (ax_i \bmod n) = \prod_{i=1}^{\phi(n)} x_i$$

$$\prod_{i=1}^{\phi(n)} ax_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n}$$

$$a^{\phi(n)} \times \left[\prod_{i=1}^{\phi(n)} x_i\right] \equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n}$$

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

Euler's Theorem

- From Fermat's theorem

  $a^n \cong a \pmod{n}$
- From Euler's theorem

  $a^{\Phi(n)+1} \cong a \bmod n$
- Though in original theorem, `a` should be relatively prime to `n`, but in the corollary, it need not be

RSA Encryption

- by Rivest, Shamir & Adleman of MIT in 1977
- best known & widely used public-key alg
- uses exponentiation of integers modulo a prime
- encrypt: $C = M^e \bmod n$
- *decrypt:* $M = C^d \bmod n = (M^e)^d \bmod n = M$
- both sender and receiver know values of n and e
- only receiver knows value of d
- public-key encryption algorithm with public key *PU* = $\{e, n\}$ & private key *PR*=$\{d, n\}$.
-

RSA Algorithm

- Requirement for public key encryption
  - It is possible to find values of e, d, n such that
    - $M^{ed} \bmod n = M$ for all $M < n$.
  - It is relatively easy to calculate M^e and C^d for all values of $M < n$.
  - It is infeasible to determine d given e and n.
- First two requirements are easily met
- Third requirement can be met for large e and n

RSA Algorithm

- First requirement
  - Consider Euler's totient function
    - For prime p, q, we have $\Phi(pq) = (p-1)(q-1)$
  - For RSA, we need $M^{ed} \bmod n = M$
    - relation between e, d can be stated as
    - $ed \bmod \Phi(n) = 1$
    - $\Rightarrow d \bmod \Phi(n) = e^{-1}$
    - Both, d and e should be relatively prime to $\Phi(n)$

RSA Algorithm

Key Generation	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$de \bmod \phi(n) = 1$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

RSA Algorithm

- Example:
- Select two primes $p=17$, $q=11$
- Calculate $pq = 17*11 = 187$
- Calculate $\Phi(187) = 16 \times 10 = 160$
- Select e relatively prime to 160 i.e. $\Phi(187)$,
  – e.g. $e = 7$
- Determine d such that $de_{\text{mod}}160 = 1$, thus $d=23$
- Thus, PU = $\{7,187\}$, and PR = $(23,187)$

Program

- Write a java program which takes following i/ps
  - `argv[0]` : Prime number `p`
  - `argv[1]`: Prime number `q`
  - `argv[2]`: Data to be encrypted.

- Computation
  - Calculate `n = p*q`
  - Calculate $\Phi(n) = \Phi(pq) = (p-1)(q-1)$
  - Calculate `e` relatively prime to $\Phi(n)$
  - Calculate `d` such that $de_{\mathrm{mod}}\ \Phi(n) = 1$
  - Encrypt data `M` i.e. $C = M^{e}{}_{\mathrm{mod}}\ n$
  - *Decrypt to get back* $M = C^{d}{}_{\mathrm{mod}}\ n$