Cairo University
Faculty of Computers and Information

Information Technology Department
IT423 - Information and Computers Networks Security
Fall 2022

Lab Assignment (3)

Implement RSA Algorithm encryption and decryption using C++ or JAVA

Input: -

1- Plain text of any size □ each character is changed to ASCII (***X***) and encrypted

Output: -
1- Cipher text in ASCII (decimal) for each character (***X***)
2- Selected p,q for each character (***X***)
3- Generated keys e,d for each character (***X***)

Read the hints which are mentioned through this document carefully.

<u>Grading Criteria: - (Total mark 15)</u>

1. **Encryption□ 1 mark**
2. **Square and multiply algorithm □ 2 marks**
3. **Key generation □ 7.5 marks**
   3.1. Select two prime numbers p, q ($p \neq q$) using fermat algorithm□ 3 marks
   3.2. Compute n □ 0.5 mark
   3.3. Compute ø (n) □ 0.5 mark
   3.4. Select e □ 0.5 mark
   3.5. Compute d using extended euclid□ 3 marks
4. **Display generated keys e & d □0.5 mark**
5. **Display p, q □0.5 mark**
6. **Decryption□ 3.5 mark**
   6.1.Transforming into the CRT domain □1 mark
   6.2. Computation in the CRT domain □1 mark
   6.3. Inverse transformation of the result □1.5 mark

<u>Remember:-</u>

<u>Hint1:-</u>**To do any exponentiation (x^H mod n) use Square and multiply algorithm as follows:-**

Square-and-Multiply for Modular Exponentiation
Input:
base element x
exponent $H = \sum_{i=0}^{t} h_i 2^i$ with $h_i \in 0,1$ and $h_t = 1$
and modulus n
Output: $x^H \mod n$
Initialization: $r = x$
Algorithm:

```
1     FOR $i = t-1$ DOWNTO 0
1.1       $r = r^2 \mod n$
          IF $h_i = 1$
1.2           $r = r \cdot x \mod n$
2     RETURN ($r$)
```

<u>**Example:**</u> **-** Compute x ^{26} using square and multiply algorithm

Step		
#0	$x = x^{\mathbf{1}_2}$	inital setting, bit processed: $h_4 = 1$
$\#1a$	$(x^1)^2 = x^2 = x^{\mathbf{10}_2}$	SQ, bit processed: h_3
$\#1b$	$x^2 \cdot x = x^3 = x^{10_2} x^{1_2} = x^{\mathbf{11}_2}$	MUL, since $h_3 = 1$
$\#2a$	$(x^3)^2 = x^6 = (x^{11_2})^2 = x^{\mathbf{110_2}}$	SQ, bit processed: h_2
$\#2b$		no MUL, since $h_2 = 0$
$\#3a$	$(x^6)^2 = x^{12} = (x^{110_2})^2 = x^{\mathbf{1100_2}}$	SQ, bit processed: h_1
$\#3b$	$x^{12} \cdot x = x^{13} = x^{1100_2} x^{1_2} = x^{\mathbf{1101}_2}$	MUL, since $h_1 = 1$
$\#4a$	$(x^{13})^2 = x^{26} = (x^{1101_2})^2 = x^{\mathbf{11010_2}}$	SQ, bit processed: h_0
$\#4b$		no MUL, since $h_0 = 0$

RSA Key Generation

Output: public key: $k_{pub} = (n, e)$ and private key: $k_{pr} = (d)$

1. Choose two large primes p and q.
2. Compute $n = p \cdot q$.
3. Compute $\Phi(n) = (p-1)(q-1)$.
4. Select the public exponent $e \in \{1, 2, \ldots, \Phi(n)-1\}$ such that

$$\gcd(e, \Phi(n)) = 1.$$

5. Compute the private key d such that

$$d \cdot e \equiv \bmod \Phi(n)$$

Hint2: -

- **For step 1 in key generation p & q must satisfy the following:-**
    1- p, q are random numbers $X <$ p, q $< 2^{15} - 1$ (where X is the ASCII (decimal) of one character in plaintext)
    2- p, q are prime numbers (Test primality of them using Fermat Primality Test mentioned below)
- **For step 4 and 5 in key generation use Extended Euclidean Algorithm OR Binary Extended Euclidean Algorithm**

Fermat Primality Test

Input: prime candidate $\tilde{p}$ and security parameter s

Output: statement "$\tilde{p}$ is composite" or "$\tilde{p}$ is likely prime"

Algorithm:

```
1      FOR $i = 1$ TO $s$
1.1        choose random $a \in \{2, 3, \ldots, \tilde{p}-2\}$
1.2        IF $a^{\tilde{p}-1} \not\equiv 1$ mod p
1.3            RETURN ("$\tilde{p}$ is composite")
2      RETURN ("$\tilde{p}$ is likely prime")
```

Hint3: **- Step 1: s = 100**

Step 1.2: is computed by using the square-and-multiply algorithm

RSA Encryption:

$$y = e_{k_{pub}}(x) \equiv x^e \bmod n$$

where $x, y \in \mathbb{Z}_n$.

Hint4: - **Compute Exponentiation (x^e mod n) using Square and Multiply method**

RSA Decryption with the Chinese Remainder Theorem (CRT)

We cannot choose a short private key without compromising the security for RSA as an attacked can use brute force attack.

In CRT the goal is to perform the exponentiation x^d mod n efficiently. First we note that the party who possesses the private key also knows the primes p and q. The basic idea of the CRT is that rather than doing arithmetic with one "long" modulus n, we do two individual exponentiations modulo the two "short" primes p and q. This is a type of transformation arithmetic. Like any transform, there are three steps:

1. Transforming into the CRT domain
2. Computation in the CRT domain
3. Inverse transformation of the result

➢ Transformation of the Input into the CRT Domain

We simply reduce the base element x modulo the two factors p and q of the modulus n, and obtain what is called the modular representation of x.

$$x_p \equiv x \bmod p$$
$$x_q \equiv x \bmod q$$

➢ Exponentiation in the CRT Domain

With the reduced versions of x we perform the following two exponentiations:

$$y_p = x_p^{d_p} \bmod p$$
$$y_q = x_q^{d_q} \bmod q$$

where the two new exponents are given by:

$$d_p \equiv d \bmod (p-1)$$
$$d_q \equiv d \bmod (q-1)$$

- **Inverse Transformation into the Problem Domain**

The remaining step is now to assemble the final result y from its modular representation (y_p, y_q). This follows from the CRT and can be done as:

$$y \equiv [q c_p] y_p + [p c_q] y_q \bmod n$$

where the coefficients c_p and c_q are computed as:

$$c_p \equiv q^{-1} \bmod p, \qquad c_q \equiv p^{-1} \bmod q$$

<u>Example:-</u>

$$\begin{aligned} p &= 11 & e &= 7 \\ q &= 13 & d &\equiv e^{-1} \equiv 103 \bmod 120 \\ n &= p \cdot q = 143 \end{aligned}$$

We now compute an RSA decryption for the ciphertext $y = 15$ using the CRT, i.e., the value $y^d = 15^{103} \bmod 143$.

In the first step, we compute the modular representation of y:-

$$y_p \equiv 15 \equiv 4 \bmod 11$$
$$y_p \equiv 15 \equiv 2 \bmod 13$$

In the second step, we perform the exponentiation in the transform domain with the short exponents. These are:-

$$d_p \equiv 103 \equiv 3 \bmod 10$$
$$d_q \equiv 103 \equiv 7 \bmod 12$$

Here are the exponentiations:-

$$x_p \equiv y_p^{d_p} = 4^3 = 64 \equiv 9 \bmod 11$$
$$x_q \equiv y_q^{d_q} = 2^7 = 128 \equiv 11 \bmod 13$$

In the last step, we have to compute x from its modular representation (x_p, x_q). For this, we need the coefficients:-

$$c_p = 13^{-1} \equiv 2^{-1} \equiv 6 \bmod 11 \qquad c_q = 11^{-1} \equiv 6 \bmod 13$$

<u>Hint5:-</u>

1- **Compute exponentiation (in example: $y_p^{d_p}$ and $y_q^{d_q}$) using square and multiply method.**
2- **Compute c_p and c_q using EEA <u>OR</u> Binary EEA.**

The plaintext x follows now as:-

$$x \equiv [q c_p] x_p + [p c_q] x_q \bmod n$$
$$x \equiv [13 \cdot 6] 9 + [11 \cdot 6] 11 \bmod 143$$
$$x \equiv 702 + 726 = 1428 \equiv 141 \bmod 143$$