Mini Project #3 **Due date: 15 January, 2017**
Fundamentals of Programming **Total points 20**

RSA Public-Key Cryptosystem

The RSA (Rivest-Shamir-Adleman) public-key cryptosystem is a kind of asymmetric cipher in which encryption and decryption employ different keys. Recall that Caesar cipher used a key k to encrypt and a key $-k$ to decrypt—referred to as symmetric key encryption. In this project, you will implement asymmetric RSA cryptosystem that is widely used for secure communication in browsers, bank ATM machines, credit card machines, mobile phones, smart cards, and the Windows operating system.

Operation

The encryption of RSA uses a public key pair (e,n) where $n = pq$ is the product of two different prime numbers p and q the public key (e,n) is published on the Internet; the private key (d,n) used for decryption is known only by Bob. If Alice wants to send Bob a message (e.g., her credit card number) she encodes her message as an integer M that is between 0 and $n-1$. Then she computes:

$$E(M) = M^e \bmod n$$

and sends the integer $E(M)$ to Bob. As an example, if $M = 2003$, $e = 7$, $d = 2563$, $n = 3713$, then Alice computes

$$\begin{aligned} E(M) &= 2003^7 \bmod 3713 \\ &= 129,350,063,142,700,422,208,187 \bmod 3713 \\ &= 746 \end{aligned}$$

When Bob receives the encrypted communication $E(M)$, he decrypts it by computing:

$$M = E(M)^d \bmod n$$

Continuing with the example above, Bob recovers the original message by computing:

$$M = 746^{2563} \bmod 3713 = 2003$$

Cryptanalysis of RSA

A cracker can decipher a message encrypted by (e,n) only if he/she knows the two factors p and q of $n = pq$. When that is the case, the cracker can use $e \times d \bmod \varphi(n) = 1$ to solve d first, and then use the procedure to decipher the message.

The factorization of n into pq is computing feasible only if both p and q are small enough. Theoretically you need to check all integers from 2 to n , to find a proper divisor of n. When we pick big primes p and q both of at least 50 digits (so n is 100 digits or more), then there will be 10^50 modulo operations if $(n\ mod\ j == 0)\ for\ 2 \leq j \leq n$ by the brute force way. Suppose your computer is 1000 MIPS or 109 operations a second, this will be 10^41 seconds or at least 10^36 days to find the prime divisor $p < q$ of n.This is computationally infeasible or impossible.

RSA Project Modules

For now we want to tackle RSA with small keys (e,n) and (d,n) so that $n = pq$ is of datatype "unsigned long long". This might provide little or no security. You can—in future—design, implement, and analyze an extended precision arithmetic data type that is capable of manipulating much larger integers, to get an uncrackable RSA cryptosystem.

Task. Encrypt and Decipher

Consider that a string (uppercase) with 28 possible characters ('A' to 0, 'B' to 1, $\cdots$, 'Z' to 25, ' ' to 26 (space), '.' to 27). To encrypt a string message M, it is first converted into message blocks $M_1, M_2, \cdots, M_n$ (each block can consist of 1 to some number k of characters). Then each message block M_i is mapped to an integer P_i. Let $k = 2$, so for two characters c_1 and c_2 of block M_i we get n_1 and n_2 with $0 \le n_1, n_2 \le 27$. Now map M_i to $P_i = n_1 \times 100 + n_2$. For example if $M_i = "ID"$, then we have 'I' = 8, 'D' = 3, so $P_i = 803$.

From P_i we calculate C_i (P is encrypted as $C = P^e (mod n)$) and then we send out the integer sequence $C_1 C_2 \cdots C_n$. On the other end, the encrypted message C_i is converted into decrypted messege blocks D_i, which are then converted to the original message string M.

When done implementing this module, test it with keys $(e,n) = (17, 3233)$ and $(d,n) = (2753, 3233)$ for this string "Rome is not built in one day." If you get the message ciphered and deciphered correctly, then work on the formating given in the "Sample Run" given at the end.

Bonus task. Key Generation

Here is an example of RSA encryption and decryption.

1. Choose two distinct prime numbers, such as $p = 61$ and $q = 53$.
2. Compute $n = pq$ giving $n = 61 \times 53 = 3233$.
3. Compute the totient of the product as

$$\varphi(n) = (p-1)(q-1)$$
$$\varphi(3233) = (61-1)(53-1) = 3120$$

4. Choose any number $1 < e < 3120$ that is coprime to 3120. Choosing a prime number for e leaves us only to check that e is not a divisor of 3120. Let $e = 17$.
5. Compute d, the modular multiplicative inverse of $e(\text{mod } \varphi(n))$ yielding, $d = 2753$. Worked example for the modular multiplicative inverse:

$$e \times d \text{ mod } \varphi(n) = 117 \times 2753 \text{ mod } 3120 = 1$$

   Use Extended Euclidean algorithm that finds solutions for Bezout's identity $ax + by = \gcd(a,b)$. The function findd() to do that is provided.

6. The public key is $(n = 3233, e = 17)$. For a padded plaintext message M, the encryption function is,

$$C(M) = M^{17} \text{ mod } 3233$$

7. The private key is $(n = 3233, d = 2753)$. For an encrypted ciphertext C, the decryption function is

$$M(C) = C^{2753} \text{ mod } 3233$$

8. For instance, in order to encrypt $M = 65$, we calculate

$$C = 65^{17} \bmod 3233 = 2790$$

9. To decrypt $C = 2790$, we calculate

$$M = 2790^{2753} \bmod 3233 = 65$$

Sample run.

Your final solution should have a printed console screen with somewhat similar messages.

```
+-----------------------------+
| RSA Public-key Cryptosystem |
+-----------------------------+

+-------------------+
| I. Key Generation |
+-------------------+

 Public  key. (e,n) = (  17,3233)
 Private key. (d,n) = (2753,3233)

+-------------------------+
| II. Cipher and Decipher |
+-------------------------+

  Input. | 'R' | 'S' | 'A' | ' ' | 'C' | 'R' | 'Y' | 'P' | 'T' | 'O' |
         |  17 |  18 |   0 |  26 |   2 |  17 |  24 |  15 |  19 |  14 |

      M. |  RS  |  A   |  CR  |  YP  |  TO  |
   P(M). | 1718 |   26 |  217 | 2415 | 1914 |
   C(P). |  120 | 1519 | 2132 | 3157 | 2891 |
   D(C). | 1718 |   26 |  217 | 2415 | 1914 |
      M. |  RS  |  A   |  CR  |  YP  |  TO  |

 Output. |  17 |  18 |   0 |  26 |   2 |  17 |  24 |  15 |  19 |  14 |
         | 'R' | 'S' | 'A' | ' ' | 'C' | 'R' | 'Y' | 'P' | 'T' | 'O' |
```

Hand-in

This is a TEAM project. Form one with atmost 2 (two) people (team with single member is also allowed). Designate one member as the "team leader", who will be responsible for the submission of the project.

Hand-in ONLY the files listed below at the appropriate location on the blackboard system at LMS. You should hand in a single compressed/archived file named Project_3_<your reg. No. XXX without angle brackets>.zip that contains the following files.

1. COMMENTED source files (rsa.c) with author information at the beginning.

2. CLEAR snapshot of your console (SNAPSHOT.png) to show that your program is working.

3. PLAIN text file (OUTPUT.txt) that includes a) author information at the beginning, b) a brief explanation of the project, c) difficulties faced, and d) any comments, or suggestions.

Programs will be graded on the following criteria:

1. **Program Specifications / Correctness.** does it compile? Are there obvious errors? Are there subtle errors? (25%)
2. **Documentation.** Is your program consistently indented in a manner that reflects the structure of your code? Is it easy to read? Are there blank lines which break up the major sections of your code? (15%)
3. **Efficieny.** Is your program efficiently organized, or is there a lot of duplicated code? Does it look well-written, or barely finished? (5%)
4. **Project Specifications.** Does your program fulfill the basic requirements? Is it done? And what else does it do? (15%)
5. **In time.** (10%)
6. **Plagiarism.** (30%)

Honor code

The student should agree to the terms below; any infringements will result in minimum marks.

- will not cheat on project
- will not share solutions to the project; and
- will notify the instructor immediately if he or she becomes aware of any other group cheating

References

RSA cryptosystem