LAB ASSIGNMENT - 3
Deadline: 23/09/2024 (Monday) 11.59 PM

Problem-1

RSA Encryption Scheme Consists of the following three algorithms:-

- Key Generation Algorithm
- Encryption Algorithm
- Decryption Algorithm

The algorithms are as given below:-

Key Generation Phase:

- p, q: large prime no.s
- $N = p \times q$
- $\phi(n) = (p-1) \times (q-1)$
- $d \xleftarrow{R} Z^*_{\phi(n)}$
- $ed \equiv 1 \bmod \phi(n)$ *in other words,* $e \equiv d^{-1} \bmod \phi(n)$
- **Public Key:** (e, n)
- **Private Key:** d
- **NOTE:** p, q and $\phi(n)$ are deleted after key generation

Review of RSA Encryption System:-

Encryption Phase:

- **Inputs:** *Plaintext:* $m \xleftarrow{R} < Z_n, +, * >$, *Public Key:* (e, n)
- **Output:** *Ciphertext:* $m^e \bmod n \Rightarrow c$

Decryption Phase:

- **Inputs:** *Ciphertext:* C, *Private Key:* (d, n)
- **Output:** *Plaintext:* $c^d \bmod n = m^{ed} \bmod n = m \bmod n \Rightarrow m$
- *Note:* We have used a theorem: $m^x \equiv m^y \bmod n$; iff, $x \equiv y \bmod \phi(n)$

Security:

- **given** (e, n) it is Hard to **find** d
- **given** C; (e, n) it is Hard to **find** m
- Implied by "Integer Factorization Problem"
- Proofs: will be covered in Cryptography Course.

Implement the above three algorithms using Crypto++ library. You must implement the fourth step of the key generation algorithm (generating the private key) as a separate function.

Note that you are not allowed to use the RSA class of Crypto++ library. However, you may use library functions for the following purposes only:-

1. Large prime number generation
2. Random number generation
3. Coprimeness Checking
4. Multiplicative Inverse Calculation
5. Modular Exponentiation

Following Class/Methods are suggested for the above mentioned purposes:-

Purpose	**Recommended Class / Method**
Large prime number generation	Class : PrimeAndGenerator Method: Generate()
Random number generation	Class: Integer Method: Randomize
Coprimeness Checking	Method: RelativelyPrime()
Multiplicative Inverse Calculation	Class : Integer Method: InverseMod()
Modular Exponentiation	Method: a_exp_b_mod_c()

Ensure that the implemented programs are non-interactive. All necessary files and inputs should be provided as command-line arguments. For example:-

./KeyGen

Output: public_key_file, private_key_file

./encrypt public_key_file data_file

Output: ciphertext_file

./decrypt private_key_file ciphertext_file

Output: decrypted_file

Note that the data file should contain a string of length not more than 128 characters.

Problem-2

Implement the RSA digital signature scheme.
RSA DS scheme consists of three algorithms, viz., 1) Key Generation, 2) Sign and 3) Verify. The Key Generation algorithm is exactly the same as that in the RSA Encryption scheme. The Signing and Verification algorithms are explained below:-

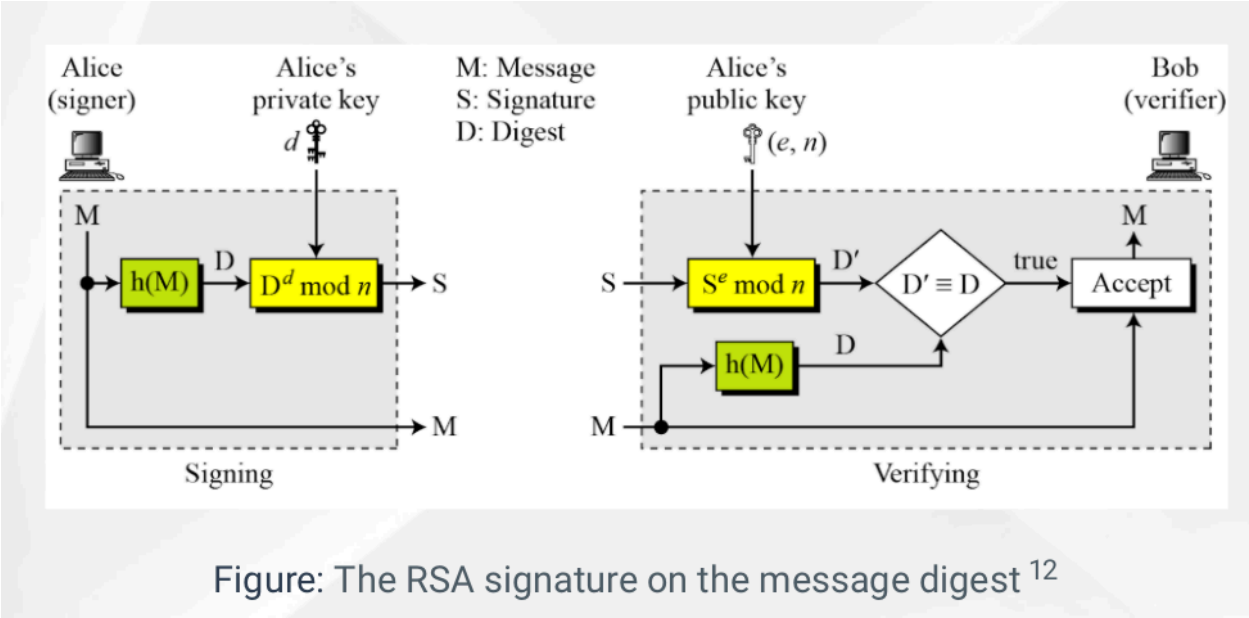

Figure: The RSA signature on the message digest [12]

Implement the Signing and Verification algos. As two non-interactive programs as follows:-

./sign private_key_file data_file

Output: signature_file

./verify public_key_file data_file signature_file

Output: Prints Success/Failure On Screen