CP460A Project Report

Group #9

Member 1: Zehao Liu 193074000

Member 2: Jialong Zhang 190227130

Member 3: Zongyang Li 180272360

Wilfrid Laurier University

Applied Cryptography – CP460A

Dr. Gao Shuan

Thursday, Dec 8, 2022

Table of Contents

CP460A Project Report

1 Introduction

The Document describes the internal algorithm of RSA, an asymmetric encryption and provide a simulation of RSA algorithm based on the Ubuntu environment, by calling BIGNUM provided by “openssl” (Github, 2022).

2 Background

Asymmetric key is secret method of key published by Martin Hellman and Whitfield Diffie in 1977. It provide a higher security compare to AES, since it separate key to private key and public key. Public key is shared in the internet and each terminal can generate their own private key. In that case, sender use receiver public key to encrypt data and receiver use their unique private key to decrypt data. The current criteria on length go key for RSA is at least 1024 bit.

3 Objectives

When transmitting information on the Internet, there is always a risk of being corrupted, and encryption can effectively reduce the risk. When transmitting information, if it has to go through some unreliable channels, the information can be leaked or even changed. The general encryption method requires sharing the encryption code beforehand. However, if sometimes there is no opportunity to share the code beforehand, RSA is a good choice in this case. Due to some unique mathematical properties of the RSA algorithm, once a message is encrypted by a public key, it can only be decrypted by another key, known as the private key. Public key encryption schemes differ from symmetric key encryption in that the same private key is used for

both the encryption and decryption processes. These differences make public key encryption like RSA useful when communicating without prior opportunity to securely assign keys.

4 How Does RSA Encryption Work?

4.1 Trap Door Functions

"RSA encryption works on the premise that the algorithm is easy to compute in one direction, but almost impossible in the reverse."(Josh, 2021) For example, given the product of two prime numbers as 382,297, it is very difficult to calculate these two prime numbers without using a computer. But on the other hand, calculating 431 * 887 is easy. This equation is easy to work out from one side, but seems impossible from the other. The numbers used in a real RSA are much larger and difficult to crack even for a computer.

4.2 Key Generation

The key for the RSA algorithm is generated in the following way:

1. Choose two large prime numbers p and q. To make factorization more difficult, p and q should be chosen randomly, of similar size, but different lengths.

2. Calculate n = p * q. n is used as the modulus of the public and private keys. Its length, usually expressed in bits, is the key length.

3. Calculate phi(n). Since n=p*q, phi(n) = (p-1) * (q-1).

4. Choose an integer e such that $1 < e < \text{phi}(n)$ and gcd(e, phi(n)) = 1; that is, e and phi(n) are coprime.

5. Determine d to be $d \equiv e\text{-}1 \pmod{\text{phi}(n)}$; that is, d is the modular multiplicative inverse of e, mod phi(n). This means: solve the equation $d*e \equiv 1 \pmod{\text{phi}(n)}$. (Wikipedia, 2022)

4.3 Encryption and Decryption

c represent cipher text; m represent message; e is coprime of φ(n) and 1<e< φ(n)

Encryption function can be written as: c ≡ m^e (mod n)

Decryption function can be written as: c^d ≡ (m^e)^d (mod n) ≡ m (mod n)

5 Program

5.1 Function List

Function name	Return data type	Data field	Description
hexString2string	char*	BIGNUM length < NBITS Message length < NBITS/2 NBITS defined as 512	ascii string to hex string
string2hexString	char*		hex string to ascii string
encryption	BIGNUM*		convert ascii message string to bn_cipher
decryption	char*		convert bn_cipher to ascii string
BN_are_coprime	int		return the gcd of two BIGNUM
printBN	void		print BIGNUM to hex string

5.2 Restrictions

Since this simulation is all about simulating RSA, it should also be able to ensure the the purpose of security. Using relatively small prime numbers during the key generation will give a lower security level and can be easily solved. Therefore we restrict our program by giving a minimum size of the prime numbers, of 512 bits.

The restricted size of the message is 256 bits, which 512/2, due to the reason that larger calculations would take up time and wouldn't be efficient in this case.

However, some property only applies to RSA and it depends on what scenarios that the RSA is being used. There are industry standards that ensures RSA to be secured. Considering the limitation of time, in our case we would decide our own standards.

5.3 Function Introduction

To approach big number object, the simulation program using BIGNUM interface. Therefore, all the following implementation should create interface to interact with BIGNUM interface.

It clear that the random prime number p and q can be generated by BN_generate_prime_ex interface. A constant ONE generated by BN_ONE and BN_mul are used for φ(n) calculation. To ensure the security, the BN_rand_range interface is used for generating e. Since true Random Number Generator is not practical, it used Cryptographically Secure Pseudorandom Number Generator instead. After random select e, BN_are_coprime apply to it to check whether gcd(e, φ(n)) ==1. By BN_mod_inverse function, private key d can be generated by e and φ(n).RSA p, q, n, φ(n), e and d are be initialized.

To interact with BIGNUM, the program make conversion between ascii string and bin string in BIGNUM. Since BIGNUM interface did not provide interface between bin string and ascii string, the programs provide a intermediate help function convert ascii with hex and convert hex with bin by BIGNUM interfaces. (details to see BN_hex2bn and BN_bn2hex)

After message being converted to BIGNUM, BN type cipher text is calculated through binary message to the power of e mod n (By interface BN_mod_exp). In a similar way, binary cipher text can be transfer back to bin message by bin_cipher to the power of d, modular inverse

of e on n mod n. With the help of binary to char transformation, the decrypted messages be calculated by the decrypted bn_message.

6 Results

All The following steps are the results of the project.

6.1 Get the Private Key

Random generated big prime numbers p and q. Get e. Use (e, n) as the public key. Calculate the private key d. The hexadecimal values of p, q, and e are listed in the following.

```
The random p generated is:  FFC87FE7864471DF2121B90164EFBFAB9B4C74F59F15E460AA0C86F0C5A9737DC
69617F9EC58CCC1F26F2451EB88FAC1367AE8E8C5E027271A23C260297DE7B7
The random q generated is:  D2CCA1063B7167B307B56FDB9E10D190E03A26B0EB823467C427DFA4734DBE096
674A21B31923C541E39A7C2CEF9F96AC4FFAF861ADFBA375FE977B22506087B
The e be selected is:  53F6EAF34033475336F1F854BB96CF51AE0B2CFF874B13AE7C48191BEBF3C0E539AE9B
F5265A22EB83DCF431D66A39CE680A0050F3EDA3C31D8BA2A8C86E9F3562BAC0E2521A81733030CEA35763ECCDC2F
788FAC2667D065D8EF9F46CB9C61DDE2DE16A0E2567DD3DAC1B764F06ECD511AAAD29837D6BF584BCFB6DB6F3C317
```

The private key d is:

```
By module inverse private key d is:  9EBE961D44362B3A082FEB91472DF3646A218CC24414EC0EACDDAB7C
3BBD5E8E7A11B3F1968886081133EB91630326B13C51978C5F74E8A1E7D580777FE36E274B135A20D9ED246145CAE
3498BAB012EE9FA8608B568F478E960852118AC972071B9E63F94325ACC56EC3C98CA503F091C3092D906170AC0B3
D246474DFD67A3
```

6.2 Encrypting a Message

```
Enter your message: hello world!
```

Let (e, n) be the public key. Encrypt the message “hello world!" (the quotations are not included). Get the hex string of cipher Text:

```
The cipher Text is:  9DFF2FBF7DD993B2C9740069B535CEF939D28C9CC165CF1B02C909077FA4C3371D587D49
B713E9A3A6E5192B192CADFD718C03C875D084AC40E75F7907C6FFA2904B87CAF43008844BEC504F2F89AA40B6177
D5B7280EC7FC1DBD0BED94B7D9A0E45206BED966E4F5C89F9E2DE1126D6D52C74E84E974673B0034C9D5D665875
```

6.3 Decrypting a Message

Decrypt the cipher text C above and convert it to text.

```
The message is: 'hello world!'
```

7 Conclusion

The paragraph illustrate the detail of RSA, the back mathematic support, why use large number and the reason why it can ensure the security of keys. The program be implemented is a simulation of how the RSA works with large number and simulate how RSA encrypt and decrypt the messages.

Reference

Github. (2022). https://github.com/openssl/openssl/blob/master/crypto/bn/bn_gcd.c

JOSH, L. (2021). What is RSA encryption and how does it work? https://www.comparitech.com/blog/information-security/rsa-encryption/#What_is_RSA_encryption

Wikipedia. (2022). RSA (cryptosystem): https://en.wikipedia.org/wiki/RSA_(cryptosystem)