

RSA-Public Key Encryption

Rukmin Rajenkumar Soni

Wilfrid Laurier University

CP-614-A- Applied Cryptography

Instructor: Professor Shaun Gao

Date: July 31 2022

Introduction: -

Any firm that wants to succeed must have a network. Nonetheless, all communication over any network, particularly the Internet, needs to be secure to prevent data breach involving sensitive and private information. Networks are utilised not just by corporations but also by individuals at home. As a result, network security is critical. This includes cryptography as a key component for allowing secure network connections (Priya,2017). Cryptography investigates strategies for concealing the true message from undesired receivers by scrambling the data using an algorithm. Modern encryption methods are more secure than historical ones and are classified into three types: secret key, public key, and hash functions. The National Bureau of Standards (NBS) method was substantially replaced by a cryptographic algorithm published in 1978 by Ron Rivest, Adi Shamir, and Leonard Adleman (Hermawan et al. 2021). The most crucial feature of RSA is the implementation of both digital signatures and a public-key cryptosystem. The published works of Diffie and Hellman from a few years earlier, who conceptualized the idea of such an algorithm but never really constructed it, served as inspiration for RSA(Wikipedia,2022).

RSA introduced two crucial concepts at the time when the age of electronic email was anticipated to arrive soon:

- i. Public-key cryptography: This concept eliminates the requirement for a "courier" to transfer keys to recipients over a different secure channel prior to delivering the message that was originally intended. Only the holder of the right decryption key may decipher an encrypted communication since with RSA, the encryption keys are public but the decryption keys are private. The encryption and decryption keys are unique to each person. The keys must be created in a way that prevents the public encryption key from simply being used to determine the decryption key.
- ii. Digital signatures: It may be necessary for the recipient to confirm that a transmitted message genuinely came from the sender (signature) and not merely from there

(authentication). The signature may then be confirmed by anybody using the appropriate public encryption key and the sender's decryption key. As a result, signatures cannot be faked. No signer may also afterwards claim not to have signed the communication.

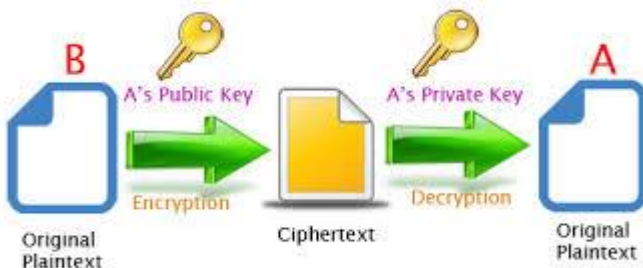
This is helpful for financial transfers among other electronic messages and transactions in addition to email. The notion of the **trapdoor** one-way function serves as the foundation of asymmetric-key cryptography. The term "**trapdoor function**" refers to a function that is simple to calculate in one way but is thought to be complex to compute in the other direction (finding its inverse) without special information (Gao, 2022).

RSA Working: -

The most difficult component of RSA, key generation, is used to create the public and private keys, while the other method, RSA function evaluation, examines encrypting and decrypting. Together, these two algorithms make up the RSA system. With the use of the two keys, Fig. 1 illustrates how to encrypt plaintext and decrypt ciphertext. In the subtopic of this topic, Key Generation, the mathematics underlying Key Generation is also explained.

Figure 1

The working of RSA encryption and decryption.



i. RSA Key Generation: -

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 $e \in \{2, \dots, \phi(n) - 1\}$ such that

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

5. Compute the private key d such that

$$d \cdot e \equiv 1 \pmod{\phi(n)}$$

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

ii. RSA Encryption and Decryption Algorithm: -

RSA operations are done over the Integer ring Z_n (i.e., arithmetic modulo n).

Encryption is simply exponentiations in the ring.

RSA Encryption Given the public key $(n, e) = k_{pub}$ and the plaintext x , the encryption function is:

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

where $x, y \in Z_n$.

RSA Decryption Given the private key $d = k_{pr}$ and the ciphertext y , the decryption function is:

$$x = dk_{pr}(y) = y^d \bmod n$$

where $x, y \in Z_n$

The security of the scheme relies on the fact that it is hard to derive the “private exponent” d given the public-key (n, e) (Gao, 2022).

Strengths and weaknesses of RSA: -

The fact that one of the two keys may be used to encrypt a message and the other key to decrypt it makes RSA one of the most used encryption algorithms. This guarantees the secrecy, integrity, authenticity, and non-reputability of data and electronic communications. As a result, care must be taken to ensure that two large random prime numbers and a public exponent e are used to determine the modulus, n , which will become useful for the public key, and that the private key is made up of those. This is important because a weak key generation will make RSA very vulnerable to attacks. Because it is challenging to computationally convert big numbers into primes, the RSA technique is successful. It is simple to multiply the two prime numbers, but it is challenging to do the opposite, or factor, when p and q 's values increase. Since it's difficult to factor big primes, RSA's strength resides on the size of its keys. However, using keys with higher modulus has drawbacks, especially when dealing with mobile devices because key generation consumes more processing power and battery life (Rouse, 2014).

Overview of the Implementation: -

To make the issue simpler to comprehend, we used the BIGNUM API and the C programming language in this project (OpenSSL Foundation, I., 2022). In the project, we used p and q as huge prime numbers, where $p = \text{"F7E75FDC469067FFDC4E847C51F452DF"}$ and $q = \text{"E85CED54AF57E53E092113E62F436F4F"}$. The public key is assumed to be $k_{pub} = (e, n)$, where e is assumed to be "0D88C3" (Du, 2021). $p \cdot q$ is used to compute n . Then, the value of (n)

is determined. To accomplish this, multiply by $(p-1)(q-1)$ (Du, 2021). For encryption and decryption techniques, we need n . Following these actions, we carry out the d computation. Deriving the Private key(d) in this way. The user is next asked to enter an ASCII string, which is then translated into a hexadecimal number and then into a Bignum for use in calculations before being stored in the variable m . We use all the findings to put the encryption technique into practise. The encryption formula is: $- m^e \bmod n$. We obtain the Cipher Text(enc) from that. Then, Cipher Text is used to decrypt the data. The decryption formula is: $- enc^d \bmod n$. We get the encrypted message (dec) as a Hexadecimal value, which we then convert back to Plaintext using a function to make it readable. The identical arithmetic that was described in the report has been employed (Du, 2021). A sample of execution is shown through the following diagrams and also each step is explained of the process.

Figure 2

Calculated private key d using p, q and n .

```
[08/04/22]seed@VM:~/.../crypto_rsa$ gcc task1.c -o task1.x -lcrypto
[08/04/22]seed@VM:~/.../crypto_rsa$ ./task1.x
p = F7E75FDC469067FFDC4E847C51F452DF
q = E85CED54AF57E53E092113E62F436F4F
n = E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1
d = 3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
Enter a string: █
```

Note. The private key(d) is calculated using the formula $d.e = \bmod \phi(n)$

Figure 2 shows the calculation of private key (d) and the values taken to calculate the same. That is, the values of p, q and n . These values are considered for the calculation of $\phi(n)$ which is done by calculating $(p-1)(q-1)$. After calculating these values, private key is calculated. After the calculation, the system is asking input for the message to encrypt and decrypt.

Figure 3

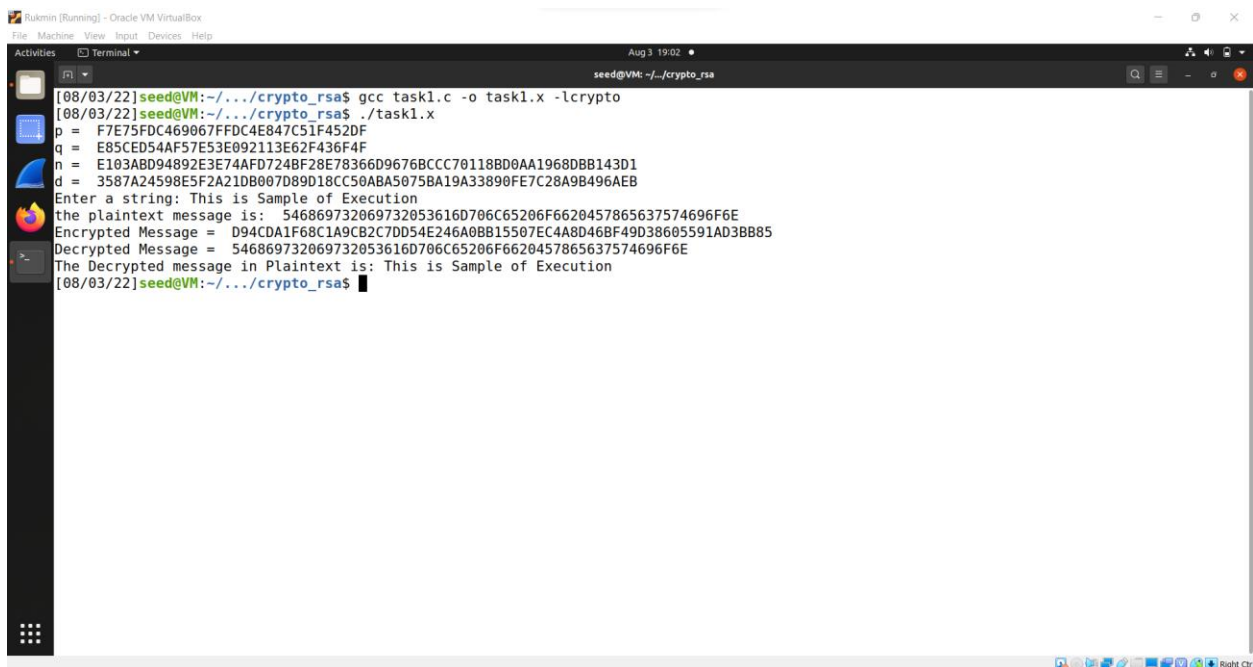
Taking message(m) from user, that needs to be encrypted

```
[08/04/22] seed@VM:~/.../crypto_rsa$ gcc task1.c -o task1.x -lcrypto
[08/04/22] seed@VM:~/.../crypto_rsa$ ./task1.x
p = F7E75FDC469067FFDC4E847C51F452DF
q = E85CED54AF57E53E092113E62F436F4F
n = E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1
d = 3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
Enter a string: This is Sample of Execution
```

Note. Here the string “This is Sample of Execution” is taken as input message from user. The Encryption and Decryption is done using the same message.

Figure 4

Execution of the RSA algorithm



```

[08/03/22] seed@VM:~/.../crypto_rsa$ gcc task1.c -o task1.x -lcrypto
[08/03/22] seed@VM:~/.../crypto_rsa$ ./task1.x
p = F7E75FDC469067FFDC4E847C51F452DF
q = E85CED54AF57E53E092113E62F436F4F
n = E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1
d = 3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
Enter a string: This is Sample of Execution
the plaintext message is: 546869732069732053616D706C65206F6620457865637574696F6E
Encrypted Message = D94CDA1F68C1A9CB2C7DD54E246A0BB15507EC4A8D46BF49D38605591AD3BB85
Decrypted Message = 546869732069732053616D706C65206F6620457865637574696F6E
The Decrypted Message in Plaintext is: This is Sample of Execution
[08/03/22] seed@VM:~/.../crypto_rsa$

```

Note. The string “This is Sample of Execution” is Encrypted and Decrypted

The sample implementation of our solution is shown in Figure 4. The image shows how the plaintext "This is Sample Execution" is converted into Hexadecimal before being utilised for encryption and decryption. So, we get the identical string as the plaintext in the decrypted message, that is: "This is Sample of Execution".

Conclusion: -

Although among the strongest, can the RSA algorithm resist anything? Nothing, without a doubt, can endure the test of time. No encryption method is even completely impervious to an assault by a practical cryptanalyst (Gilbert,2017). Simple but time-consuming techniques like brute-force may be able to decrypt a message, but probably not a whole encryption scheme. We must also use a probabilistic approach, which means there is always a possibility that "one key out of a million" will be found (Tahir,2015). We haven't figured out way to demonstrate that an encryption system is impenetrable yet. If we are unable to verify it, we shall at least try to crack the code. This is essentially how RSA and the NBS standard were certified.

References

- Du, W. (2021). *Computer & Internet Security: A hands-on approach, (2nd Edition)*. Wenliang Du. ISBN: 9781733003933
- Gao, S. (2022). RSA. CP614A-Applied-Cryptography-RSA. Retrieved May 2022, from <https://mylearningspace.wlu.ca/d2l/le/content/439272/viewContent/2971161/View>
- Hermawan, N. T., Winarko, E., & Ashari, A. (2021). Eight prime numbers of modified RSA algorithm method for more Secure Single Board Computer Implementation. *International Journal on Advanced Science, Engineering and Information Technology*, 11(6), 2375. <https://doi.org/10.18517/ijaseit.11.6.13700>
- OpenSSL Foundation, I. (2022). *OpenSSL*. Bignum Api. Retrieved August 3, 2022, from <https://www.openssl.org/docs/man1.0.2/man3/bn.html>
- Priya, N., & Kannan, M. (2017). Comparative study of RSA and Probabilistic Encryption/decryption algorithms. *International Journal Of Engineering And Computer Science*. <https://doi.org/10.18535/ijecs/v6i1.04>
- Rouse, M. (2014, November). *Information security definitions*. SearchSecurity. Retrieved 2017, from <http://searchsecurity.techtarget.com/definition/RSA>
- Tahir, A. S. (2015). Design and implementation of RSA algorithm using FPGA. *INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY*, 14(12), 6361–6367. <https://doi.org/10.24297/ijct.v14i12.1737>
- Wikipedia, W. (2022, July 1). *RSA numbers*. Wikipedia. Retrieved August 2, 2022, from https://en.wikipedia.org/wiki/RSA_numbers