RSA and SHA-256 Algorithm

June 23, 2023

Kritika Bansal (2021CSB1184) ,
Manik Gupta (2021MCB1187)

Instructor:
Dr. S.R.S. Iyengar

Teaching Assistant:
Ms. Poonam Adhikari

Summary: Our project entails the implementation and analysis of some cryptographic algorithms. In this project, we have implemented two algorithms: RSA algorithm and the SHA-256 algorithm. We have shown asymmetric encryption using RSA algorithm and hashing using SHA-256 algorithm. RSA is an asymmetric encryption algorithm that works on two keys: Public and Private. It encrypts using Public key and decrypts using Private key. SHA-256 algorithm is a cryptographic hashing algorithm which converts plain text into its hash value. These cryptographic algorithms play a vital role in authentication, data privacy, password protection, cryptocurrencies and many more.

---

1. Introduction

In cryptography, the prefix "crypto" means hidden and the suffix "graphy" means writing. Thus, cryptography consists of techniques that make use of mathematical concepts to convert information to forms that are hard to decode. Thus, only people for whom information is intended can understand it. Two such algorithms are RSA and SHA-256.

In this project, we have analyzed and implemented RSA and SHA-256 algorithms.

RSA is an asymmetric public-key signature algorithm that uses the public key to encrypt data and a private key to decrypt it. In this project we have used various basic algorithms like Miller Rabin's Test to generate probabilistic prime number, extended euclid's algorithm to generate private key and then binary exponential algorithm to encrypt and decrypt our message.

SHA-256 is a hashing algorithm that, irrespective of the size of plain text, converts it into a hash value of 256 bits. We have implemented SHA-256 algorithm which uses basic bitwise operators such as xor, right shift operator to generate and modify buffer values used in every round of operation. We have generated time-based OTP using SHA-256 algorithm.

2. RSA and SHA-256 Algorithm

This section talks about RSA and SHA-256 Algorithms.

2.1. RSA Algorithm

RSA is an asymmetric cryptographic algorithm that makes use of a public and private key for encryption and decryption respectively. Public Key is used for encryption purpose while private key is used for the decryption purpose. In RSA algorithm the sender and receiver need not share the same key. While encrypting text messages/numbers we can take prime number of length upto 500-1000 bits also.

How it works:

We need to take very large prime numbers in order to secure our message as the multiplication of two numbers is easy but factorizing the product of large numbers is difficult.

1. Public Key:

- Select two prime numbers P and Q
- Define n = P * Q
- Take an integer e s.t. $1 < e < \phi(n)$ where $\phi(n)$ = (P-1) * (Q-1)
- Public key: n and e

2. Private Key:

- d = (k * $\phi(n)$ + 1) / e where k is some integer
- Private key: d

Next, we convert the message into integer form using ASCII code.

3. Encryption:

- Encrypted value, c = m $\wedge$ e (mod n) where m is message in integer form

4. Decryption:

- Decrypted value, m = c $\wedge$ d (mod n)

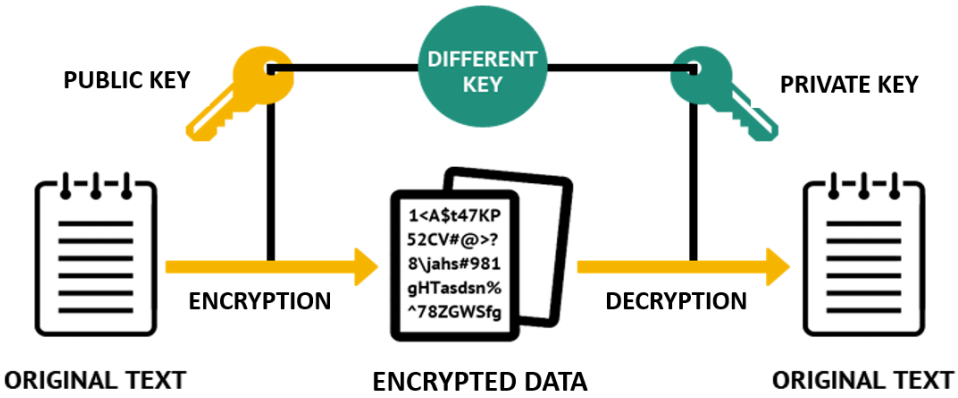

Figure 1: RSA Encryption process

2.2. SHA-256

SHA-256 is a cryptographic hashing algorithm which converts any text into a hash value of 256 bits.

It majorly follows the following three steps:

1. Appending bits:

In the bit form, we have to modify our message such that its total length is a multiple of 512. For that, we will append 1 followed by 0 bits so that the total length is 64 less than the multiple of 512. Next, we will append length of message in 64 bit form.

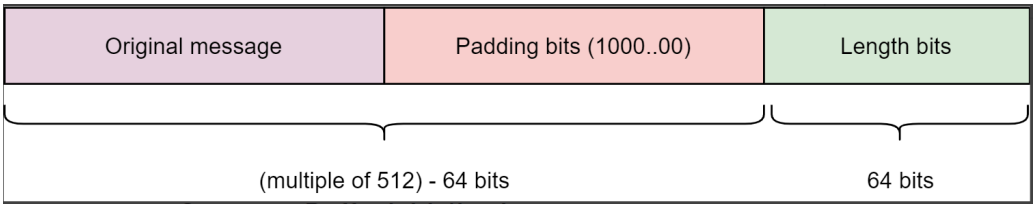

Figure 2: Original message after preprocessing

2. Buffer initialization:

Following are the default eight buffer values which are actually the first 32 bits of the fractional part of the square root of first 8 primes.

A = 0x6a09e667
B = 0xbb67ae85
C = 0x3c6ef372
D = 0xa54ff53a
E = 0x510e527f
F = 0x9b05688c
G = 0x1f83d9ab
H = 0x5be0cd19

Next, we have to initialize an array containing 64 constants denoted by K.

For example,

K = ['0x428a2f98', '0x71374491', '0xb5c0fbcf', '0xe9b5dba5', '0x3956c25b', '0x59f111f1', '0x923f82a4', '0xab1c5ed5', '0xd807aa98', '0x12835b01', '0x243185be', '0x550c7dc3', '0x72be5d74', '0x80deb1fe', '0x9bdc06a7', '0xc19bf174', '0xe49b69c1', '0xefbe4786', '0x0fc19dc6', '0x240ca1cc', '0x2de92c6f', '0x4a7484aa', '0x5cb0a9dc', '0x76f988da', '0x983e5152', '0xa831c66d', '0xb00327c8', '0xbf597fc7', '0xc6e00bf3', '0xd5a79147', '0x06ca6351', '0x14292967', '0x27b70a85', '0x2e1b2138', '0x4d2c6dfc', '0x53380d13', '0x650a7354', '0x766a0abb', '0x81c2c92e', '0x92722c85', '0xa2bfe8a1', '0xa81a664b', '0xc24b8b70', '0xc76c51a3', '0xd192e819', '0xd6990624', '0xf40e3585', '0x106aa070', '0x19a4c116', '0x1e376c08', '0x2748774c', '0x34b0bcb5', '0x391c0cb3', '0x4ed8aa4a', '0x5b9cca4f', '0x682e6ff3', '0x748f82ee', '0x78a5636f', '0x84c87814', '0x8cc70208', '0x90befffa', '0xa4506ceb', '0xbef9a3f7', '0xc67178f2']

3. Compression Function:

As our message is a multiple of 512, it will be divided into 512 bits. Each chunk will then undergo 64 round of operation. The output from each round serves as the input for the next round.

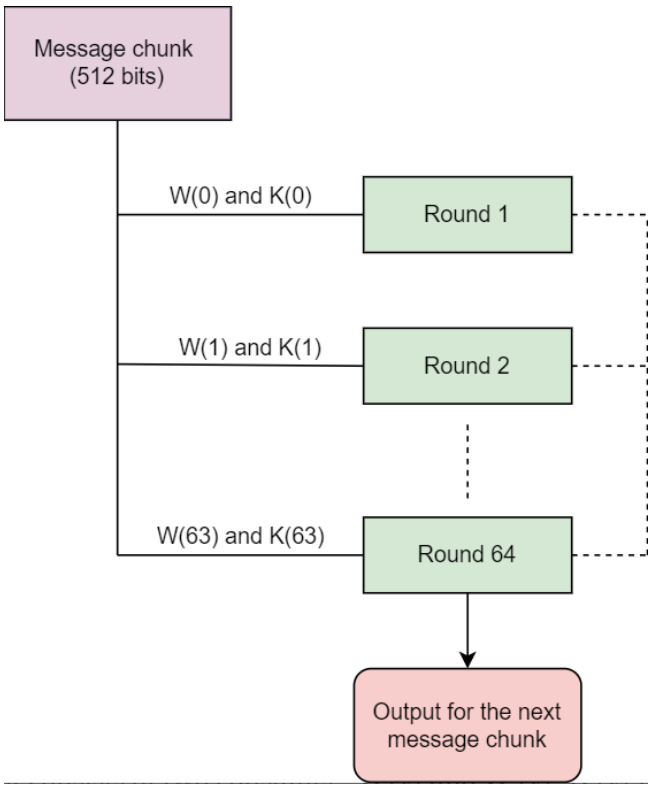

Figure 3: Rounds of operation performed a 512 bit message chunk

Next, W[i]s are initialized by breaking chunk of 512 bits into 16 subblocks of 32 bits each. These 16 W[i]s work for the first 16 rounds and for the subsequent 48 rounds, following formulas are used for the calculation of W[i]s.

1. s0 = (w[i-15] rightrotate 7) xor (w[i-15] rightrotate 18) xor (w[i-15] rightshift 3)
2. s1 = (w[i- 2] rightrotate 17) xor (w[i- 2] rightrotate 19) xor (w[i- 2] rightshift 10)
3. w[i] = w[i-16] + s0 + w[i-7] + s1

Each round of operations consists of following process:

1. ch = (e and f) xor ((not e) and g)
2. ma = (a and b) xor (a and c) xor (b and c)
3. S0 = (a rightrotate 2) xor (a rightrotate 13) xor (a rightrotate 22)
4. S1 = (e rightrotate 6) xor (e rightrotate 11) xor (e rightrotate 25)
5. temp1 = h + S1 + ch + k[i] + w[i]
6. temp2 := S0 + ma
7. h = g
8. g = f
9. f = e
10. e = d + temp1
11. d = c
12. c = b
13. b = a
14. a = temp1 + temp2

After 64 rounds are over for each chunk, the modified original buffer values are added to the modidfied ones.

At last, all the eight buffer values are converted to their hex form and appended in order to form the 256-bit hash value.

3. Analysis

In this section, we will see the time complexities and space complexities associated with RSA and SHA-256 Algorithms.

3.1. Time Complexity

Time Analysis of RSA Algorithm :

1. Prime number Generation:

We need to generate very large prime numbers for RSA Algorithm. For this purpose, we are using Miller-Rabin Test. The time complexity for generating probabilistic prime numbers is O(k log3 n).

2. Key Generation:

Public Key: To generate a public key, we can select any number between 2 and ϕ(n). As the public key is generated randomly between these numbers, its time complexity is O(1).

Private Key: We are using Extended Euclid's algorithm to generate private key and its time complexity is O(min(e,ϕ(n))) where e is the value of public key and ϕ(n) is the totient function.

3. Encryption: We are using binary exponentiation function to generate our cipher message and the time complexity for the function is O(log(e)) (where e is the public key) as our message is raised to power of e and then taken modulo with our number n.

4. Decryption: For decrypting our cipher text, the time complexity is O(log(d)) where d is our private key.

Generally time taken for RSA algorithm is quite large so only message of short length can be encrypted efficiently and thus it prevents its usage to encrypt large images or large text file.

Time Analysis of SHA-256 Algorithm:

The time complexity of the SHA-256 (Secure Hash Algorithm 256-bit) algorithm is proportional to the size of the input message being hashed. If n is the length of input message, time complexity of SHA-256 algorithm increases as n increases.

3.2. Space Complexity

Space Analysis of RSA Algorithm :

Text: The time complexity is O(n) where n is the length of message.

Space Analysis of SHA-256 Algorithm:

The space is used to store the input message in bit form. As the ASCII value of every character of our message takes 8-bits to store it in binary form, we can say the space complexity is O(8*n) where n is the number of characters in our message. Hash value takes the constant space of 256 bits.

4. Demonstration of encryption and decryption using RSA and SHA-256 Algorithm

This section covers demonstration of encryption and decryption of text using RSA and SHA-256 Algorithm using various examples.

4.1. Demonstration of RSA Algorithm

Numbers:

Length of Prime Number: 100
Public Key: 123694662439176519964570275514 3
Private key: 469661231456858563926650015837669983120524941786377372818727
Number: 479527592840273439729

Cipher: 134328614693255157920590065393071179523922034142823182247929 0
Decrypted Number: 479527592840273439729

Text:

Length of Prime Number: 10
Public Key: 1303
Private key: 651847

Message: hello world
Cipher Matrix: [25679, 216874, 561062, 561062, 327213, 310766, 479164, 327213, 546213, 561062, 663675]
Decrypted Message: hello world

4.2. Demonstration of SHA-256 Algorithm

1. Input: hello

Hash value: cee558b4e4fd56a59d73051e16794ffee2d58eaf414d4941323f2d8754dd97e0

2. Input: abc

Hash Value: 735141618f052de21ee3a00c915f66af5a5d0ae52a35e009a334bdd1338a1484

3. Input: abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopqpqpmomom

Hash Value: b3b7e3eb7ac95d52e1fe14a8cca562796977949e430f2b23379e3273da2c7c8f

4. Input: abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopqp

Hash Value: 7881f04ca5dc81c817974cbfb6792a95957a94933b55b99825daf0acb7b513a

5. Input: ibsnqwpzhillptcinmtvamymvixjxaumjddwxsxxjhjhnftynajhsluuctgjytazlcdewsexbjcpumdcfbbbmzwxcmjmnx fqurvaarapdswyatlyvqsxdefmehicwwdnkshzgysaxxenmtpirbhphxyaesgwigdxzqpekouenexqkqgpnzzwyjppc

Hash Value: a3f03d97509b53dc6b48dc8514d4afcfc2e947f55bf351e62c136c2aa6ef588e

5. Applications of RSA Algorithm

RSA Algorithms have various applications in practical life which we come across from time to time. Its main implementation comes in authentication and secure data transfer.

5.1. Key Exchange

RSA algorithm can be used for secure key exchange, which means that two parties can exchange a secret key without actually sending the key over the network. The information sent by the sender will be encrypted using the recipient's public key and on receiving the cipher message, it will be decrypted using the recipient's private key. This ensures that data is not leaked in the way.

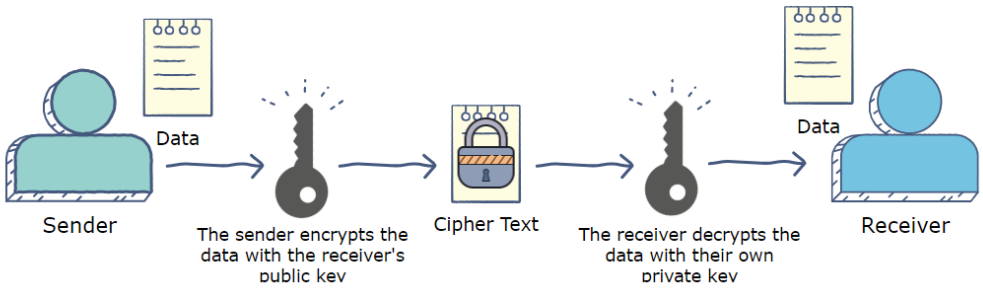

Figure 4: Key Exchange

5.2. Secure File Transfer

RSA can be used to encrypt and sign files during their transfer to ensure confidentiality and integrity.

5.3. Digital Signatures

Digital Signatures can be sent using the RSA algorithm as the sender can sign a message using their private key, and on receiving the message, the receiver can verify the digital signature using the sender's public key.

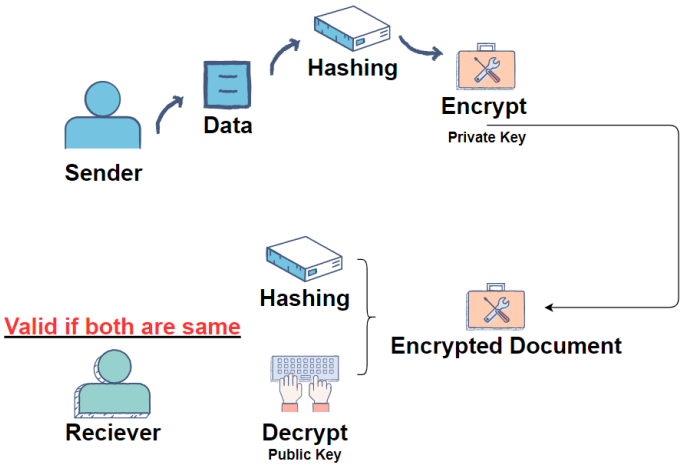

Figure 5: Key Exchange

5.4. Encryption and Data Protection

RSA can be used for encryption and decryption of sensitive data. RSA encryption is relatively slow compared to symmetric encryption algorithms, so it is typically used for encrypting small data chunks.

5.5. Secure Login and Authentication

Many authentication systems like RSA tokens and smart cards which are based upon RSA are used for secure login and user authentication. Basically, digital signature is generated using RSA and then the same is verified by it. This helps us in ensuring secure access to systems and protecting user credentials.

6. Applications of SHA-256 Algorithm

SHA-256 algorithmhave various applications in practical life which we come across from time to time. Its main implementation comes in password authentication and cryptocurrencies.

6.1. Password Storage

Login pages need user to enter their password to successfully enter into their account. Now the password set by user is saved as a hash value using SHA-256 algorithm in the database. Now when user enters the password, SHA-256 converts password to corresponding hash value. Now the stored value is compared with the produced value and this is used to authenticate the user.

6.2. Blockchain Technology

In blockchain implementations, many cryptocurrencies make wide use of SHA-256 as it is performed repeatedly to solve cryptographic puzzles and thus it is used in mining which in turn helps to secure and validate transactions.

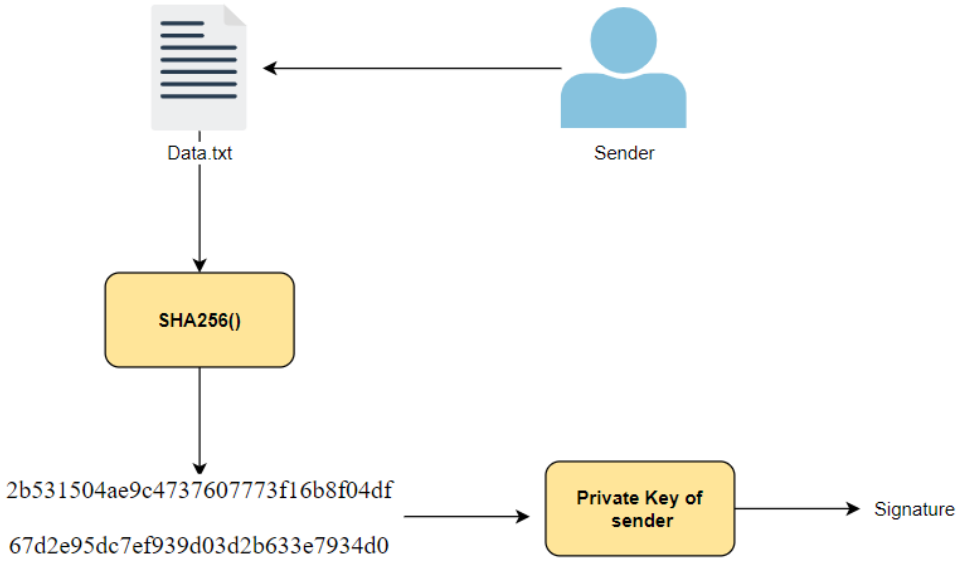

Figure 6: Use in Blockchain

6.3. Time-Based OTP Generation

SHA-256 plays a crucial role in producing time-based OTP generation. The system's current time and date is taken as an input string and hashed using the SHA-256 algorithm. Then a random prime number is chosen so as to convert the hash value into a 6-digit OTP by taking modulus with it.

7. Conclusions

During the implementation of RSA and SHA-256 algorithms in this project, we got to learn about a lot of new things. We learned how data can be exchanged between two individuals without getting leaked the way using the RSA algorithm. We learned how OTP is generated using the SHA-256 algorithm. Further, we can also implement different applications of these algorithms in the future.

8. Bibliography and citations

Here is a list of all the sources consulted while developing the work:

https://www.geeksforgeeks.org/rsa-algorithm-cryptography/
https://www.educative.io/answers/what-are-the-different-steps-in-sha-256
https://medium.com/@domspaulo/python-implementation-of-sha-256-from-scratch-924f660c5d57
https://en.wikipedia.org/wiki/RSA_(cryptosystem)

9. Acknowledgement

I would like to express my special thanks of gratitude to Prof. SRS Iyengar for giving me this chance to work on a topic of my interest which allowed me to do a lot of research and I learned a whole lot of new things. I would also like to thank Ms. Poonam Adhikari for guiding me through every step of the project and helping me out in case of any problems. The completion of my project would not have been possible without their exceptional guidance and support.