

RSA for software license generation program

Milena Flores

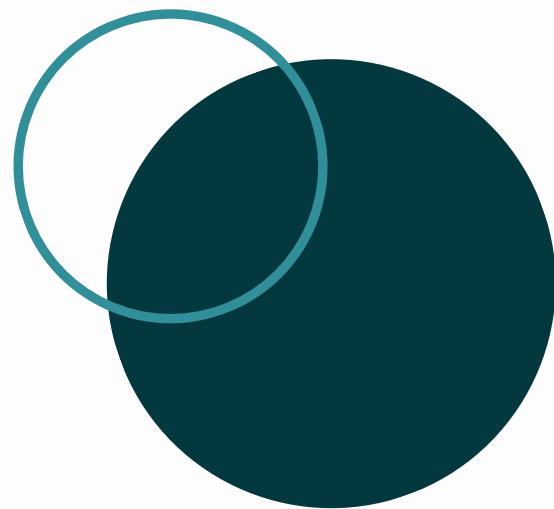
TOPICS

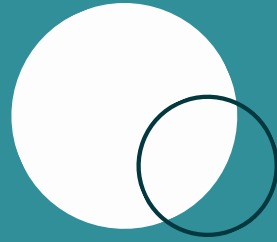
Objective

What is RSA

Explanation of the code

Conclusions

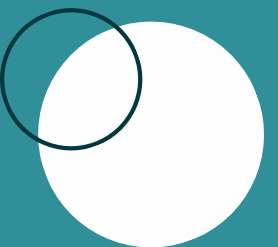




”

The objective of this project is use the RSA encryption method for the purpose of generate, verify and decrypt software licenses.

“



WHAT IS RSA

RSA IS A ENCRYPTION METHOD

In general terms this algorithm asks for two prime numbers commonly called p and q , then multiply them and get $n=p*q$. This n will be a public and private number which will help us to encrypt the message. The way to decrypt the message is according to a number of mathematical procedures that consist of obtaining new numbers that are going to be private.



RSA ALGORITHM

Let us see step by step the algorithm and how it works:

1. Chosse to numbers p and q.

2. Calculated

$$n = p * q \quad (1)$$

3. Using the fundamental property of Euler in RSA, calculate

$$\phi(n) = (p - 1) * (q - 1) \quad (2)$$

4. Chosse an e such that

$$1 < e < \phi(n) \quad (3)$$

and is coprime to phi

$$gdc(e, \phi(n)) = 1 \quad (4)$$

so, they do not share any divisor except 1.

5. Calculated a d such that it is the inverse of e modulo phi

$$d = inv(e, \phi(n)) \quad (5)$$

i.e., we want to search for a number such that

$$(d * e) mod(\phi(n)) = 1 \Leftrightarrow (d * e) = 1(\phi(n)) \quad (6)$$

6. Now, we have the private and public key that will be:

Public key : (e,n) and a private key : (d,n)

7. To encrypt a message we will use the public key like this:

$$c = m^e mod(n) \quad (7)$$

Where,

m : Is the original message represented by a number.

e,n : Are public keys.

c : Is the encrypted message.

8. To decrypt the message we will use the private keys like this:

$$m = c^d mod(n) \quad (8)$$

Where,

m : Is the original message represented by a number.

e,n : Are public keys.

c : Is the encrypted message.

This algorithm works thanks to Euler's and Fermat's theorems applied to encryption.



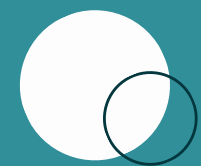
EXPLANATION OF THE CODE

```
===== RSA for software licenses =====  
Using primes:  
P = 61, Q = 53, N = 3233  
Public Exponent e = 17  
Private Exponent d = 2753  
  
Select an option:  
1) Show public key (N, E)  
2) Generate a license code  
3) Verify a license code  
4) Decrypt a message  
0) Exit
```

CONSOLE OF THE PROGRAM

- Take constants.
- Show options:
 - Each option have a function

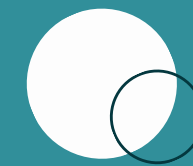
Functions



RSA METHOD

As funtions defined to RSA method we have:

- modPow(long base, long exp, long mod): helps to calculate the encrypted and desencrypted number.
- calculateAsciiSum(name): add all the ASCII values of name.



IMPLEMENTATION

As funtions defined to implement with the program we have:

- showPublicKey()
- generateLicense()
- verifyLicense()
- decryptMessage()

How it works

DEFINE THE CONSTANTS FOR RSA

```
const long P    = 61;
const long Q    = 53;
const long N    = P * Q;
const long PHI  = (P - 1) * (Q - 1);
const long E    = 17;
const long D    = 2753;
```

DEFINE FUNCTION MODPOW

```
long modPow(long base, long exp, long mod) {
    long resultado = 1;
    base = base % mod;
    while (exp > 0) {
        if (exp % 2 == 1) {
            resultado = (resultado * base) % mod;
        }
        exp = exp / 2;
        base = (base * base) % mod;
    }
    return resultado;
}
```

DEFINE FUNCTION CALCULATEASCIISUM

```
long calculateAsciiSum(const string &name) {
    long sum = 0;
    for (char c : name) {
        sum += static_cast<int>(c);
    }
    return sum;
}
```


DEFINE FUNCTION SHOWPUBLICKEY

```
void showPublicKey() {
    cout << "\n--- Public key ---\n";
    cout << "Modulus (n) = " << N << "\n";
    cout << "Public Exponent (e) = " << E << "\n";
    cout << "You can share these values with anyone  
who needs to verify licenses.\n";
    cout << "\n-----\n";
}
```

DEFINE FUNCTION GENERATELICENSE

```
void generateLicense() {
    cout << "\nEnter username (no spaces): ";
    string username;
    cin >> username;

    cout << "Enter expiration year: ";
    int expirationYear;
    cin >> expirationYear;

    long asciiSum = calculateAsciiSum(username);
    long payload = asciiSum + expirationYear;
    long licenseCode = modPow(payload, D, N);

    cout << "\n--- Generate license ---\n";
    cout << "Username: " << username << "\n";
    cout << "Expiration Year: " << expirationYear << "\n";
    cout << "ASCII Sum: " << asciiSum << "\n";
    cout << "Payload (asciiSum + year) = " << payload << "\n";
    cout << "License Code (signature) = " << licenseCode << "\n\n";
    cout << "Save this license code and provide it to the user.\n";
    cout << "\n-----\n";
}
```

DEFINE FUNCTION VERIFYLICENSE

```
void verifyLicense() {
    cout << "\nEnter username (no spaces): ";
    string username;
    cin >> username;

    cout << "Enter expected expiration year (e.g., 2025): ";
    int expectedYear;
    cin >> expectedYear;

    cout << "Enter license code (integer): ";
    long licenseCode;
    cin >> licenseCode;

    long asciiSum = calculateAsciiSum(username);
    long verifiedPayload = modPow(licenseCode, E, N);

    cout << "\n--- Verification result ---\n";
    cout << "Username: " << username << "\n";
    cout << "ASCII Sum: " << asciiSum << "\n";
    cout << "Expected Expiration Year: " << expectedYear << "\n";
    cout << "Verified Payload = " << verifiedPayload << "\n";
```

```
long expectedPayload = asciiSum + expectedYear;
if (verifiedPayload != expectedPayload) {
    cout << "=> ERROR: Signature does not match the expected payload.\n";
    cout << "    The license is NOT valid.\n";
    return;
}

int licensedYear = static_cast<int>(verifiedPayload - asciiSum);

time_t t = time(nullptr);
tm *localTime = localtime(&t);
int currentYear = localTime->tm_year + 1900;

cout << "=> Signature is valid. Licensed Year = " << licensedYear << "\n";
cout << "    Current Year = " << currentYear << "\n";

if (currentYear <= licensedYear) {
    cout << "=> The license is STILL VALID.\n";
} else {
    cout << "=> The license HAS EXPIRED.\n";
}
cout << "-----\n";
```

DEFINE FUNCTION DECRYPTMESSAGE

```
void decryptMessage() {  
    cout << "\nEnter the encrypted number: ";  
    long ciphertext;  
    cin >> ciphertext;  
  
    long decrypted = modPow(ciphertext, D, N);  
  
    cout << "\n--- Decryption result ---\n";  
    cout << "Encrypted number: " << ciphertext << "\n";  
    cout << "Decrypted number = " << decrypted << "\n\n";  
    cout << "-----\n";  
}
```

MAIN

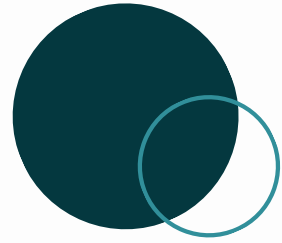
```
int main() {
    cout << "==== RSA for software licenses ==== \n";
    cout << "Using primes: \n";
    cout << "P = " << P << ", Q = " << Q << ", N = " << N << " \n";
    cout << "Public Exponent e = " << E << " \n";
    cout << "Private Exponent d = " << D << " \n \n";

    cout << "Select an option: \n";
    cout << "  1) Show public key (N, E) \n";
    cout << "  2) Generate a license code \n";
    cout << "  3) Verify a license code \n";
    cout << "  4) Decrypt a message \n";
    cout << "  0) Exit \n";

    int option;
    cin >> option;
    cin.ignore(1000, '\n'); // clear input buffer
```

```
switch (option) {
    case 0:
        cout << "Exiting... \n";
        break;
    case 1:
        showPublicKey();
        break;
    case 2:
        generateLicense();
        break;
    case 3:
        verifyLicense();
        break;
    case 4:
        decryptMessage();
        break;
    default:
        cout << "Invalid option. Program terminated. \n";
        break;
}

return 0;
```



RESULTS OF THE CODE

```
===== RSA for software licenses =====  
Using primes:  
P = 61, Q = 53, N = 3233  
Public Exponent e = 17  
Private Exponent d = 2753  
  
Select an option:  
1) Show public key (N, E)  
2) Generate a license code  
3) Verify a license code  
4) Decrypt a message  
0) Exit
```

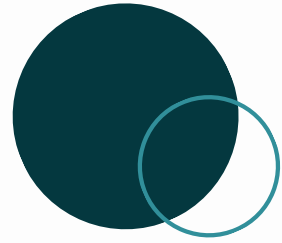
Console

```
--- Public key ---  
Modulus (n) = 3233  
Public Exponent (e) = 17  
You can share these values with anyone who needs to verify licenses.  
-----
```

Option 1

```
Enter username (no spaces): milena  
Enter expiration year: 2030  
  
--- Generate license ---  
Username: milena  
Expiration Year: 2030  
ASCII Sum: 630  
Payload (asciiSum + year) = 2660  
License Code (signature) = 1075  
  
Save this license code and provide it to the user.  
-----
```

Option 2



RESULTS OF THE CODE

```
Enter username (no spaces): milena
Enter expected expiration year (e.g., 2025): 2030
Enter license code (integer): 1075

--- Verification result ---
Username: milena
ASCII Sum: 630
Expected Expiration Year: 2030
Verified Payload = 2660
=> Signature is valid. Licensed Year = 2030
    Current Year = 2025
=> The license is STILL VALID.
-----
```

```
Enter username (no spaces): milena
Enter expected expiration year (e.g., 2025): 2030
Enter license code (integer): 1079

--- Verification result ---
Username: milena
ASCII Sum: 630
Expected Expiration Year: 2030
Verified Payload = 3106
=> ERROR: Signature does not match the expected payload.
    The license is NOT valid.
```

Option 3

```
Enter the encrypted number: 1075

--- Decryption result ---
Encrypted number: 1075
Decrypted number = 28

-----
```

Option 4

CONCLUSION

- THE RESULTS OBTAINED MEET THE PROPOSED OBJECTIVE AS IT USES THE RSA ENCRYPTION METHOD TO GENERATE, VERIFY AND DECRYPT SIMPLE LICENSE CODES.
- IT IS WORTH NOTING THAT EACH STEP AND/OR OPTION PROVIDES THE USER WITH A CLEAR UNDERSTANDING AND TRANSPARENCY ABOUT WHAT IS HAPPENING.
- DETECTED LIMITATION IS THAT SINCE SMALL PRIME NUMBERS P AND Q WERE USED, THE CODE IS NOT TRULY SECURE.



**THANKS FOR YOUR
ATTENTION**