/\*

\*

\*

\* Description:

\* RSA DecryptionKey using OpenSSL library and Python encode/decode

\* \*/

#include <stdio.h>

#include <string.h>

#include <openssl/bn.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

void printBN(char \*msg, BIGNUM \*tmp){

char \*number\_str = BN\_bn2hex(tmp); // Convert BIGNUM to hex

printf("%s%s\n", msg, number\_str); // Print hex

OPENSSL\_free(number\_str); // Free memory

}

int main(int argc, char \*argv[]){

BN\_CTX \*ctx = BN\_CTX\_new();

// Here initialize all needed BIGNUM variables

// 1- Encryption Key variable

BIGNUM \*EncryptionKey = BN\_new();

// 2- Decryption Key variable

BIGNUM \*DecryptionKey = BN\_new();

// 3- product of large prime numbers p and q

BIGNUM \*ProductPQ = BN\_new();

// 4- Totient of (n) Euler's totient function

BIGNUM \*Totient = BN\_new();

// 5- Encrypted Message variable

BIGNUM \*Message = BN\_new();

// 6- Decrypted Ciphertext variable

BIGNUM \*Ciphertext = BN\_new();

// Find DecryptionKey Key (d) using (e) and (Phin):

// 1- Assign value to (e) EncryptionKey Key from hex

BN\_hex2bn(&EncryptionKey, "010001");

// 2- Assign value to (Phin) EncryptionKey Key from hex

BN\_hex2bn(&Totient, "E103ABD94892E3E74AFD724BF28E78348D52298BD687C44DEB3A81065A7981A4");

// 3- Calculate the DecryptionKey Key (Private Key) d=e mod(Phi(n))

BN\_mod\_inverse(DecryptionKey, EncryptionKey, Totient, ctx);

char \*CC= malloc(100 \* sizeof(char));

printf("\nEnter your Encrypted Message:\n");

// Read the Encrypted Message from the user to variable CC

fgets(CC, 100, stdin);

// Assign the input value in variable (CC) to Encrypted Message variable

BN\_hex2bn(&Message, CC);

/\*

Decrypt ciphertext using D=C^d(mod(n)) ,

where: (D) is the Decrypted Ciphertext and (C) is the Ciphertext

\*/

// Assign value to (n) product of two large prime numbers from hex

BN\_hex2bn(&ProductPQ, "E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1");

// decrypt Ciphertext using the Private Key

BN\_mod\_exp(Ciphertext, Message, DecryptionKey, ProductPQ, ctx);

// Convert Hex string to ASCII letters

printf("\nOriginal Message:\n");

char str1[500]="print(\"";

char \*str2 = BN\_bn2hex(Ciphertext);

char str3[]="\".decode(\"hex\"))";

strcat(str1,str2);

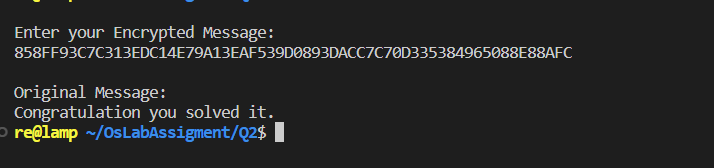
strcat(str1,str3);

char\* args[]={"python2", "-c",str1, NULL};

execvp("python2", args);

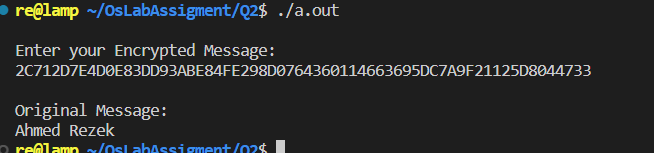
return EXIT\_SUCCESS;

}



A black screen with numbers

Description automatically generated



Discussion:

In this assignment, we utilize the RSA encryption algorithm to securely transmit plaintext messages. RSA is a form of public key cryptography, where the sender encrypts the message using a public key, and the receiver decrypts the message using a private key. The public key is freely available to anyone who wants to encrypt a message, while the private key is kept secret and only used by the intended recipient to decrypt the ciphertext. This ensures that the message remains confidential and can only be read by the intended recipient.