# RSA and Public-Private Key

This lab is about to explore the RSA cryptosystem that is the first public-private key algorithm for computer security.

Image: Use the SEED image [SEEDUbuntu16.04](http://www.cis.syr.edu/~wedu/seed/lab_env.html) VM.

Reference: <http://www.cis.syr.edu/~wedu/seed/Labs_16.04/Crypto/Crypto_RSA/>

1. Generating the public key and private key using RSA algorithm. Create a file named BN\_simple\_find\_d.c following the below code. Compile the file with the parameter –lcrypto and output BN\_simple\_find. Run the program and show what it is displayed?

#include <stdio.h>

#include <openssl/bn.h>

#define NBITS 256

void printBN(char \*msg, BIGNUM \* a)

{

/\* Use BN\_bn2hex(a) for hex string

\* Use BN\_bn2dec(a) for decimal string \*/

char \* number\_str = BN\_bn2hex(a);

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

OPENSSL\_free(number\_str);

}

int main ()

{

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

BIGNUM \*a = BN\_new();

BIGNUM \*b = BN\_new();

BIGNUM \*n = BN\_new();

BIGNUM \*res = BN\_new();

BIGNUM \*p = BN\_new();

BIGNUM \*q = BN\_new();

BIGNUM \*e = BN\_new();

BIGNUM \*d = BN\_new();

BIGNUM \*res1 = BN\_new();

BIGNUM \*res2 = BN\_new();

BIGNUM \*res3 = BN\_new();

BIGNUM \*one = BN\_new();

// Initialize a, b, n

BN\_generate\_prime\_ex(a, NBITS, 1, NULL, NULL, NULL);

BN\_dec2bn(&b, "273489463796838501848592769467194369268");

BN\_rand(n, NBITS, 0, 0);

// res = a\*b

BN\_mul(res, a, b, ctx);

printBN("a \* b = ", res);

// res = aˆb mod n

BN\_mod\_exp(res, a, b, n, ctx);

printBN("aˆc mod n = ", res);

//Segment 1

BN\_hex2bn(&p, "F7E75FDC469067FFDC4E847C51F452DF");

BN\_hex2bn(&q, "E85CED54AF57E53E092113E62F436F4F");

BN\_hex2bn(&e, "0D88C3");

BN\_dec2bn(&one, "1");

BN\_sub(res1, p, one);

BN\_sub(res2, q, one);

BN\_mul(res3, res1, res2, ctx);

BN\_mod\_inverse(d, e, res3, ctx);

printBN("d= ", d);

return 0;

}

﻿a \* b =A36E7DD9D3D6138A42C1F3FF8CBDB0F3B38E5E67A20AFF1FD117B9BC40D6151E46416BB2FF0F64B44C7F9EAB2F257D4C

aˆc mod n = 750899B5BC6F2058C4173E2D48110380F66333E3B915D442ACCF5C9DE73CC9DF

d= 3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB

1. Read the slides posted on the BlackBoard. Consider how the d is generated. Please explain it in a few thrusts.

**Choose 2 distinct prime numbers, compute n = pq, compute totient of the product, choose any number between 1 and the totient that is coprime, compute d (modular multiplicative inverse of e).**

1. Create a new file named BN\_encrypt\_decrypt.c following the below code. This file is to encrypt the message using the public key and to decrypt the ciphertext using the private key. However, it still needs some parameters to run. According to our slides and the code provided here, please point out which ones are for public key and which ones are for private key.

#include <stdio.h>

#include <openssl/bn.h>

#define NBITS 256

void printBN(char \*msg, BIGNUM \* a)

{

                /\* Use BN\_bn2hex(a) for hex string

                \* Use BN\_bn2dec(a) for decimal string \*/

                char \* number\_str = BN\_bn2hex(a);

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

                OPENSSL\_free(number\_str);

}

int main ()

{

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

                BIGNUM \*m = BN\_new(); // **public key**

                BIGNUM \*d = BN\_new(); //**private key**

                BIGNUM \*n = BN\_new();//**private and public key**

                BIGNUM \*e = BN\_new();//**public key**

                BIGNUM \*enc = BN\_new(); // **for Public Key**

                BIGNUM \*dec = BN\_new(); // **for Private Key**

                //m,e,n,d is inserted here:

                BN\_hex2bn(&m, "");

                BN\_hex2bn(&e, "");

                BN\_hex2bn(&n, "");

                BN\_hex2bn(&d, "");

                BN\_mod\_exp(enc, m, e, n, ctx);

                printBN("encrypted message = ", enc);

                BN\_mod\_exp(dec, enc, d, n, ctx);

                printBN("decrypted message = ", dec);

                return 0;

}

1. Given the message is ***A top secret!***, use the python tool provided here: ***python -c ’print("A top secret!".encode("hex"))’*** to get the hex value of the message. Please fill the blanks in BN\_encrypt\_decrypt.c using the message’s hex value as well as the following hexadecimal values:

n = DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5   
e = 010001   
d = 74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D

Compile the file as BN\_encrypt\_decrypt and run it. Please show the output.

**﻿4120746f702073656372657421**

1. Use another python tool provided here: **python -c ’print("xxxxxx".decode("hex"))’** to decode the ASCII text. Please show the output.

**﻿python -c 'print("4120746f702073656372657421".decode("hex"))'**

﻿**A top secret!**

1. Change the message content in the program to a text that you want to encrypt and decrypt. Please (1) show your new message and (2) display the printout as well as (3) the decoded text using python.

***python -c ’print("Madina Martazanova".encode("hex"))’*** t

**﻿4d6164696e61204d617274617a616e6f7661**

**﻿[04/01/19]seed@VM:~$ python -c 'print("4d6164696e61204d617274617a616e6f7661".decode("hex"))'**

**Madina Martazanova**

**[04/01/19]seed@VM:~$**