**RSA Public-Key Encryption and Signature Lab**

**Task 1: Deriving the Private Key**

#include<stdio.h>

#include<openssl/bn.h>

#define NBITS 512

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

{

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 \*p = BN\_new() ;

BIGNUM \*q = BN\_new() ;

BIGNUM \*e = BN\_new() ;

BIGNUM \*d = BN\_new() ;

BIGNUM \*n = BN\_new() ;

BIGNUM \*totient = BN\_new() ;

BIGNUM \*v = BN\_new() ;

BIGNUM \*b = BN\_new() ;

BIGNUM \*m = BN\_new() ;

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

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

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

BN\_hex2bn(&m, "0001");

BN\_sub(v, p,m);

BN\_sub(b, q,m);

BN\_mul(n, p, q, ctx);

printBN("p \* q = ", n);

BN\_mul(totient, v, b, ctx);

printBN("totient = ", totient);

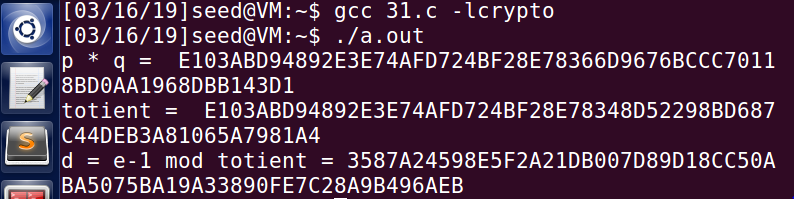
BN\_mod\_inverse(d, e, totient, ctx);

printBN("d = e-1 mod totient =", d);

return 0;

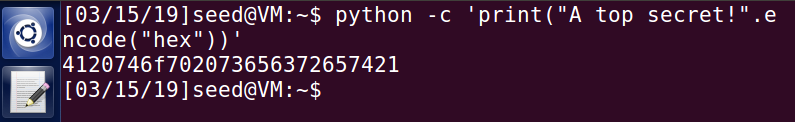
}

**OUTPUT:**



**Task 2: Encrypting a Message**

**Using the python command for encryption, I got the following output:**



**Using a C code for the same task, I was able to fetch the following output:**

#include<stdio.h>

#include<openssl/bn.h>

#define NBITS 512

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

{

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 \*n = BN\_new() ;

BIGNUM \*e = BN\_new() ;

BIGNUM \*M = BN\_new() ;

BIGNUM \*C = BN\_new() ;

BN\_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");

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

BN\_hex2bn(&M, "4120746f702073656372657421");

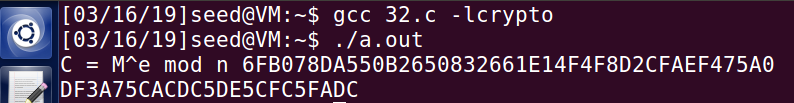
BN\_mod\_exp(C,M,e,n,ctx);

printBN("C = M^e mod n", C);

return 0;

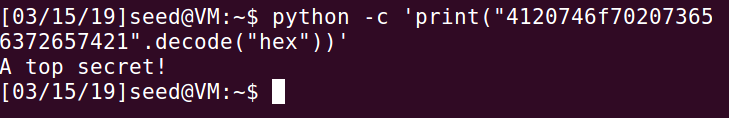
}

**OUTPUT:**



**Task 3: Decrypting a Message**

**Using the python code for decryption, the following output was observed.**



**Using a C code for the same task with the hex value “4120746f702073656372657421”, I was able to fetch the following output:**

#include<stdio.h>

#include<openssl/bn.h>

#define NBITS 512

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

{

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 \*p = BN\_new() ;

BIGNUM \*q = BN\_new() ;

BIGNUM \*e = BN\_new() ;

BIGNUM \*d = BN\_new() ;

BIGNUM \*n = BN\_new();

BIGNUM \*M = BN\_new() ;

BIGNUM \*C = BN\_new() ;

BN\_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

BN\_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");

BN\_hex2bn(&C, "8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F");

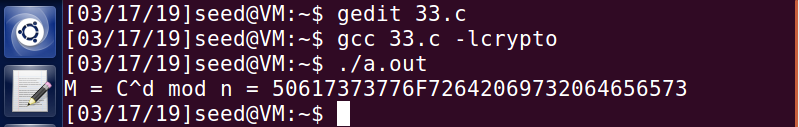
BN\_mod\_exp(M,C,d,n,ctx);

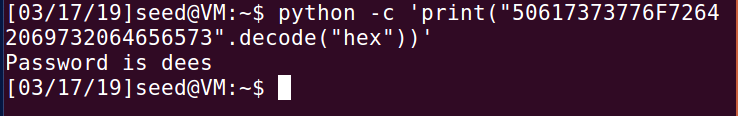
printBN("M = C^d mod n =", M);

return 0;

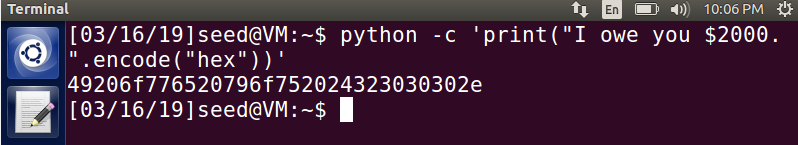
}

**OUTPUT**





**3.4 Task 4: Signing a Message**



#include<stdio.h>

#include<openssl/bn.h>

#define NBITS 512

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

{

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

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

OPENSSL\_free(number\_str);

}

//49206f776520796f75202433353633

int main ()

{

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

BIGNUM \*s = BN\_new() ;

BIGNUM \*M = BN\_new() ;

BIGNUM \*d = BN\_new() ;

BIGNUM \*e = BN\_new() ;

BIGNUM \*n = BN\_new() ;

BN\_hex2bn(&M, "49206f776520796f752024323030302e");

BN\_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

BN\_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");

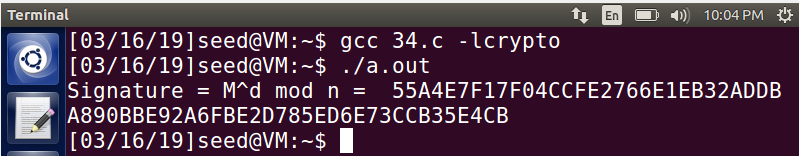
BN\_mod\_exp(s,M,d,n,ctx);

printBN("Signature = M^d mod n = ", s);

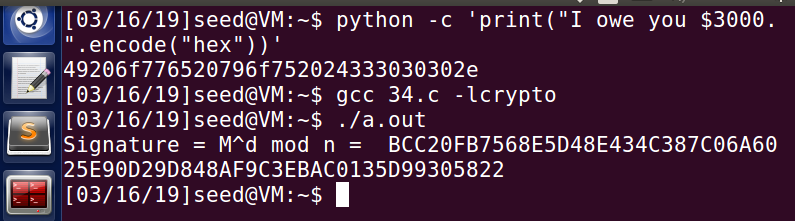
return 0;

}

**OUTPUT:**

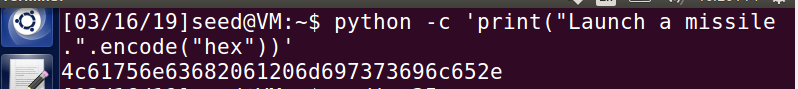


Changing the message from $2000 to $3000 results in the following observation:



The above snapshot clearly depicts that changing the message slightly results in a completely different signature without adding any additional bits to the signature.

**3.5 Task 5: Verifying a Signature**



#include<stdio.h>

#include<openssl/bn.h>

#define NBITS 512

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

{

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

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

OPENSSL\_free(number\_str);

}

//4c61756e63682061206d697373696c652e

int main ()

{

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

BIGNUM \*s = BN\_new() ;

BIGNUM \*M = BN\_new() ;

BIGNUM \*d = BN\_new() ;

BIGNUM \*e = BN\_new() ;

BIGNUM \*n = BN\_new() ;

BN\_hex2bn(&s, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F");

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

BN\_hex2bn(&n, "AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");

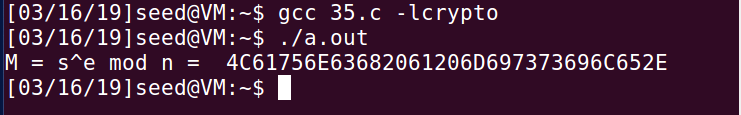
BN\_mod\_exp(M,s,e,n,ctx);

printBN("M = s^e mod n = ", M);

return 0;

}

**OUTPUT:**



Changing the last byte of the signature from 2F to 3F adds 20 more bits and results in a totally different message as shown in the snapshot below:

