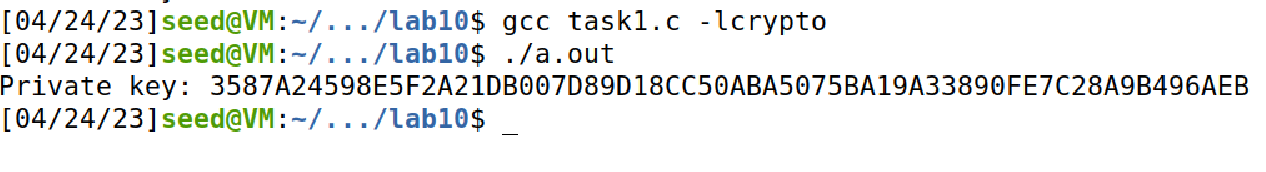
Lab 10:

Task 1:



Code:

#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, \*q, \*n, \*phi, \*e, \*d, \*m, \*c, \*res;

BIGNUM \*new\_m, \*p\_minus\_one, \*q\_minus\_one;

p = BN\_new(); q = BN\_new(); n = BN\_new(); e = BN\_new();

d = BN\_new(); m = BN\_new(); c = BN\_new();

res = BN\_new(); phi = BN\_new(); new\_m = BN\_new();

p\_minus\_one = BN\_new(); q\_minus\_one = BN\_new();

// Set the public key exponent e

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

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

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

BN\_sub(p\_minus\_one, p, BN\_value\_one()); // Compute p-1

BN\_sub(q\_minus\_one, q, BN\_value\_one()); // Compute q-1

BN\_mul(n, p, q, ctx); // Compute n=pq

BN\_mul(phi, p\_minus\_one, q\_minus\_one, ctx); // Compute (\*@$\phi(n)$@\*)

// Compute the private key exponent d, s.t. ed mod phi(n) = 1

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

printBN("Private key:", d);

// Clear the sensitive data from the memory

BN\_clear\_free(p); BN\_clear\_free(q); BN\_clear\_free(d);

BN\_clear\_free(phi); BN\_clear\_free(m); BN\_clear\_free(new\_m);

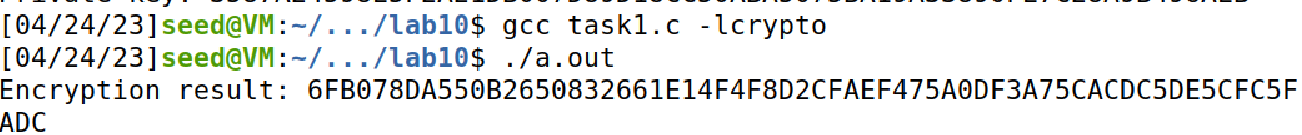
BN\_clear\_free(c); BN\_clear\_free(res);

BN\_clear\_free(p\_minus\_one); BN\_clear\_free(q\_minus\_one);

return 0;

}

Task 2:



Code:

#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, \*q, \*n, \*phi, \*e, \*d, \*m, \*c, \*res;

BIGNUM \*new\_m, \*p\_minus\_one, \*q\_minus\_one;

p = BN\_new(); q = BN\_new(); n = BN\_new(); e = BN\_new();

d = BN\_new(); m = BN\_new(); c = BN\_new();

res = BN\_new(); phi = BN\_new(); new\_m = BN\_new();

p\_minus\_one = BN\_new(); q\_minus\_one = BN\_new();

// Set the public key exponent e

BN\_dec2bn(&e, "65537");

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

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

// Encryption: calculate m^e mod n

BN\_hex2bn(&m,"4120746f7020736563726574210a");

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

printBN("Encryption result:", c);

// Clear the sensitive data from the memory

BN\_clear\_free(p); BN\_clear\_free(q); BN\_clear\_free(d);

BN\_clear\_free(phi); BN\_clear\_free(m); BN\_clear\_free(new\_m);

BN\_clear\_free(c); BN\_clear\_free(res);

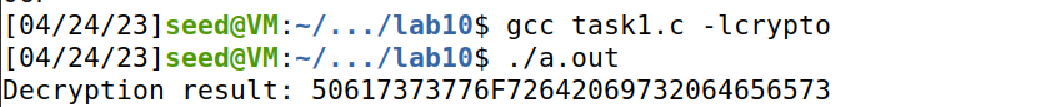
BN\_clear\_free(p\_minus\_one); BN\_clear\_free(q\_minus\_one);

return 0;

}

Task3:

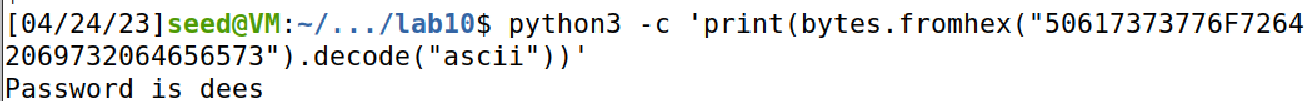
Screenshot1:



Converting hex value to string:

For conversion I used the below command:

python3 -c 'print(bytes.fromhex("50617373776F72642069732064656573").decode("ascii"))'



Code:

#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, \*q, \*n, \*phi, \*e, \*d, \*m, \*c, \*res;

BIGNUM \*new\_m, \*p\_minus\_one, \*q\_minus\_one;

p = BN\_new(); q = BN\_new(); n = BN\_new(); e = BN\_new();

d = BN\_new(); m = BN\_new(); c = BN\_new();

res = BN\_new(); phi = BN\_new(); new\_m = BN\_new();

p\_minus\_one = BN\_new(); q\_minus\_one = BN\_new();

// Set the public key exponent e

BN\_dec2bn(&e, "65537");

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

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

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

/\* // Check whether e and (\*@$\phi(n)$@\*) are relatively prime.

printBN("Encryption result:", c);\*/

// Decryption: calculate c^d mod n

BN\_mod\_exp(new\_m, c, d, n, ctx);

printBN("Decryption result:", new\_m);

// Clear the sensitive data from the memory

BN\_clear\_free(p); BN\_clear\_free(q); BN\_clear\_free(d);

BN\_clear\_free(phi); BN\_clear\_free(m); BN\_clear\_free(new\_m);

BN\_clear\_free(c); BN\_clear\_free(res);

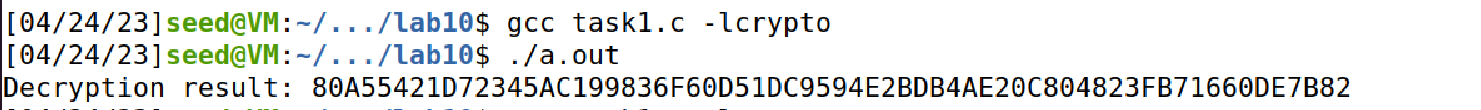
BN\_clear\_free(p\_minus\_one); BN\_clear\_free(q\_minus\_one);

return 0;

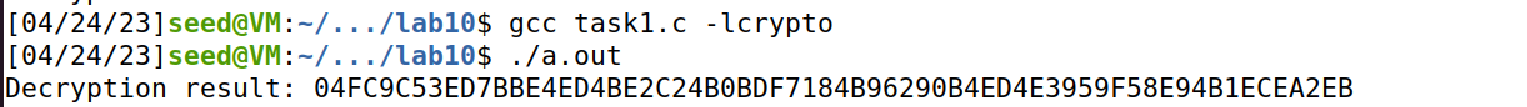
}

Task 4:

Hex value for I owe you $2000 is 49206f776520796f75202432303030



Hex value for I owe you $3000 is 49206f776520796f75202433303030



Code:

#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, \*q, \*n, \*phi, \*e, \*d, \*m, \*c, \*res;

BIGNUM \*new\_m, \*p\_minus\_one, \*q\_minus\_one;

p = BN\_new(); q = BN\_new(); n = BN\_new(); e = BN\_new();

d = BN\_new(); m = BN\_new(); c = BN\_new();

res = BN\_new(); phi = BN\_new(); new\_m = BN\_new();

p\_minus\_one = BN\_new(); q\_minus\_one = BN\_new();

// Set the public key exponent e

BN\_dec2bn(&e, "65537");

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

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

BN\_hex2bn(&c,"text hex value");

// Decryption: calculate c^d mod n

BN\_mod\_exp(new\_m, c, d, n, ctx);

printBN("Decryption result:", new\_m);

// Clear the sensitive data from the memory

BN\_clear\_free(p); BN\_clear\_free(q); BN\_clear\_free(d);

BN\_clear\_free(phi); BN\_clear\_free(m); BN\_clear\_free(new\_m);

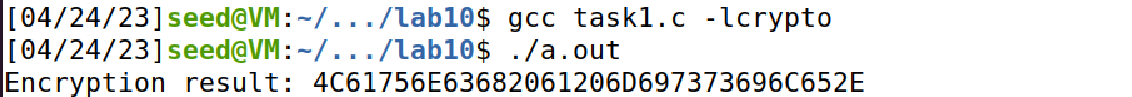
BN\_clear\_free(c); BN\_clear\_free(res);

BN\_clear\_free(p\_minus\_one); BN\_clear\_free(q\_minus\_one);

return 0;

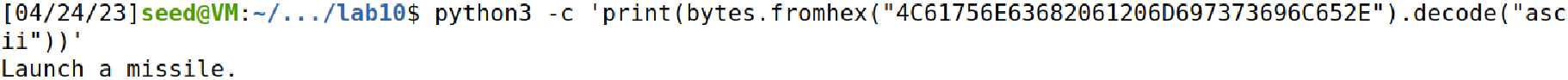
}

Task 5:



Verifying using the following command:

python3 -c 'print(bytes.fromhex("4C61756E63682061206D697373696C652E").decode("ascii"))'



Code:

#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, \*q, \*n, \*phi, \*e, \*d, \*m, \*c, \*res;

BIGNUM \*new\_m, \*p\_minus\_one, \*q\_minus\_one;

p = BN\_new(); q = BN\_new(); n = BN\_new(); e = BN\_new();

d = BN\_new(); m = BN\_new(); c = BN\_new();

res = BN\_new(); phi = BN\_new(); new\_m = BN\_new();

p\_minus\_one = BN\_new(); q\_minus\_one = BN\_new();

// Set the public key exponent e

BN\_dec2bn(&e, "65537");

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

// Encryption: calculate m^e mod n

BN\_hex2bn(&m,"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F");

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

printBN("Encryption result:", c);

// Clear the sensitive data from the memory

BN\_clear\_free(p); BN\_clear\_free(q); BN\_clear\_free(d);

BN\_clear\_free(phi); BN\_clear\_free(m); BN\_clear\_free(new\_m);

BN\_clear\_free(c); BN\_clear\_free(res);

BN\_clear\_free(p\_minus\_one); BN\_clear\_free(q\_minus\_one);

return 0;

}

Task 6:

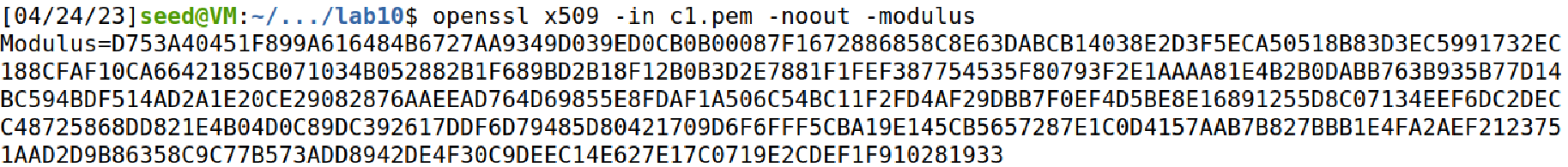
Step 1: Download a certificate from a real web server.

Text

Description automatically generated

Step 2: Extract the public key (e, n) from the issuer’s certificate

N value:



E value:

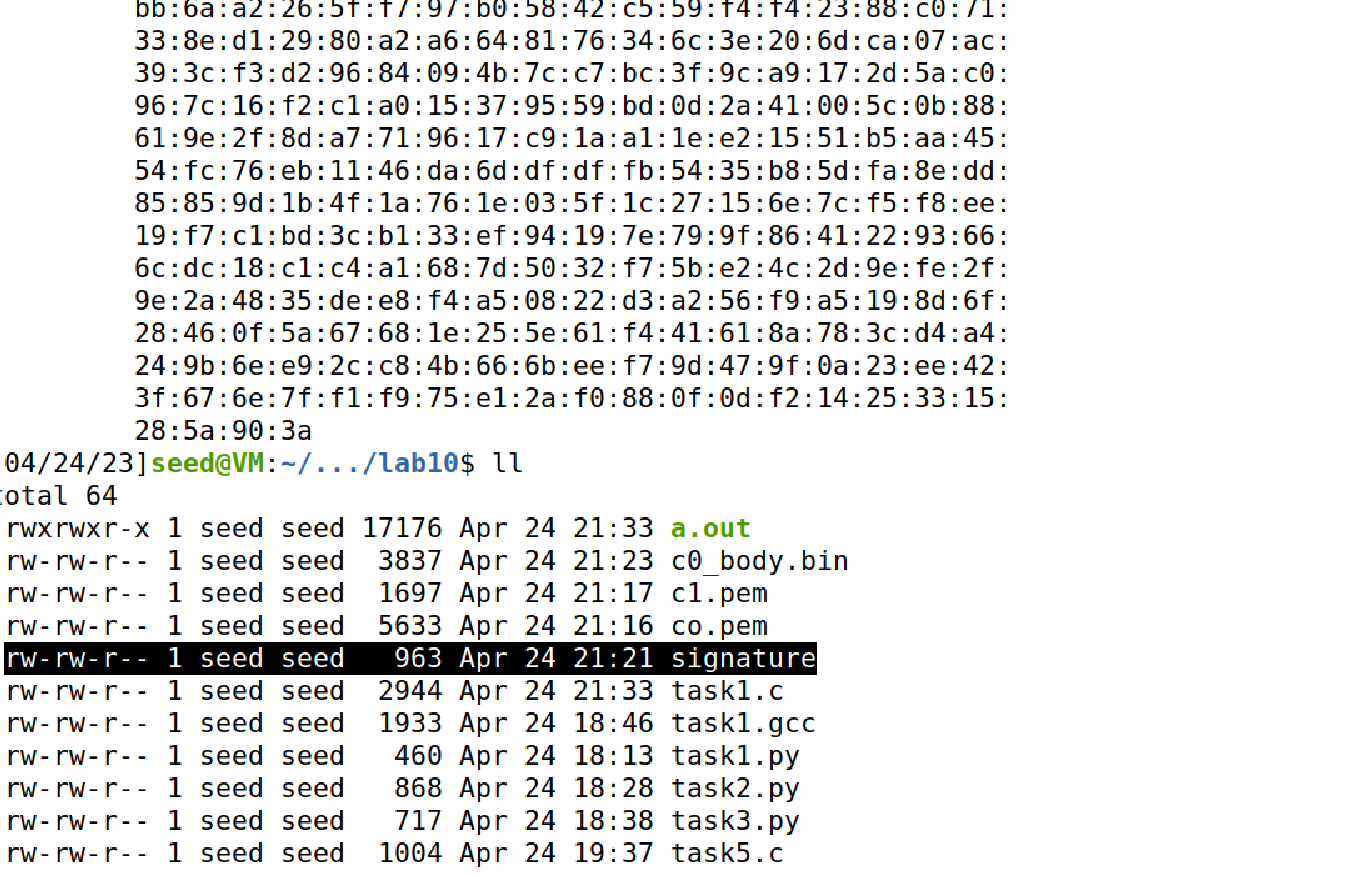
Text

Description automatically generated

Text

Description automatically generated

Step 3: Extract the signature from the server’s certificate.



Step 4: Extract the body of the server’s certificate.

Text

Description automatically generated

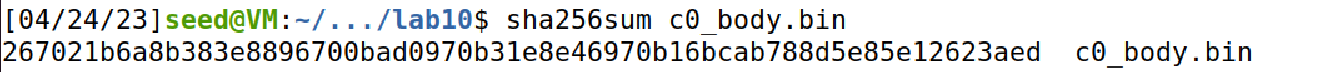
Step 5: Verify the signature.

A screenshot of a computer

Description automatically generated with medium confidence

The highlighted part is equal to the signature body

Signature body:



Code:

#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, \*q, \*n, \*phi, \*e, \*d, \*m, \*c, \*res;

BIGNUM \*new\_m, \*p\_minus\_one, \*q\_minus\_one;

p = BN\_new(); q = BN\_new(); n = BN\_new(); e = BN\_new();

d = BN\_new(); m = BN\_new(); c = BN\_new();

res = BN\_new(); phi = BN\_new(); new\_m = BN\_new();

p\_minus\_one = BN\_new(); q\_minus\_one = BN\_new();

// Set the public key exponent e

BN\_dec2bn(&e, "65537");

BN\_hex2bn(&n

// Encryption: calculate m^e mod n

BN\_hex2bn(&m,"");

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

printBN("Encryption result:", c);

// Clear the sensitive data from the memory

BN\_clear\_free(p); BN\_clear\_free(q); BN\_clear\_free(d);

BN\_clear\_free(phi); BN\_clear\_free(m); BN\_clear\_free(new\_m);

BN\_clear\_free(c); BN\_clear\_free(res);

BN\_clear\_free(p\_minus\_one); BN\_clear\_free(q\_minus\_one);

return 0;

}