

# Foundations Network Security

## RSA LAB

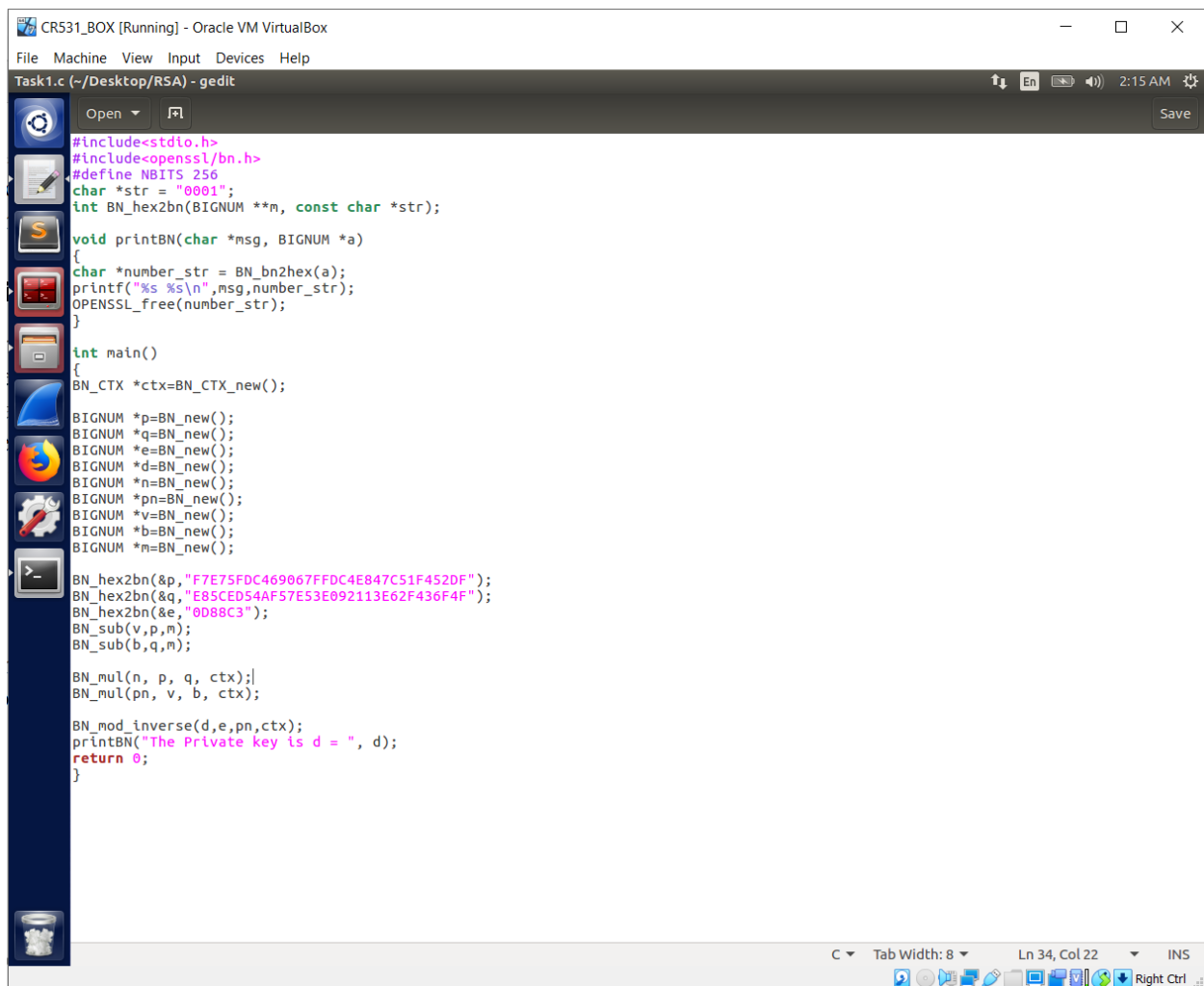
Chakradhar Reddy Donuri

E949F496

### Task 1: Deriving the Private Key

**Deliverable.** Your code should print out “The private key is d=...”, where the dots should be replaced with the actual value of d that you calculated

**Code:**



```
#include<stdio.h>
#include<openssl/bn.h>
#define NBITS 256
char *str = "0001";
int BN_hex2bn(BIGNUM **m, const char *str);

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 *pn=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_sub(v,p,m);
    BN_sub(b,q,m);

    BN_mul(n, p, q, ctx);
    BN_mul(pn, v, b, ctx);

    BN_mod_inverse(d,e,pn,ctx);
    printBN("The Private key is d = ", d);
    return 0;
}
```

## Output:

```
/bin/bash
[04/03/20]seed@VM:~/.../RSA$ gcc Task1.c -lcrypto
[04/03/20]seed@VM:~/.../RSA$ ./a.out
The Private key is d = 182363E2DA763AD4DC94DBE64CD6869FEDD1B10B1E8810416A9CD4
E9AF6B7FC5
[04/03/20]seed@VM:~/.../RSA$
```

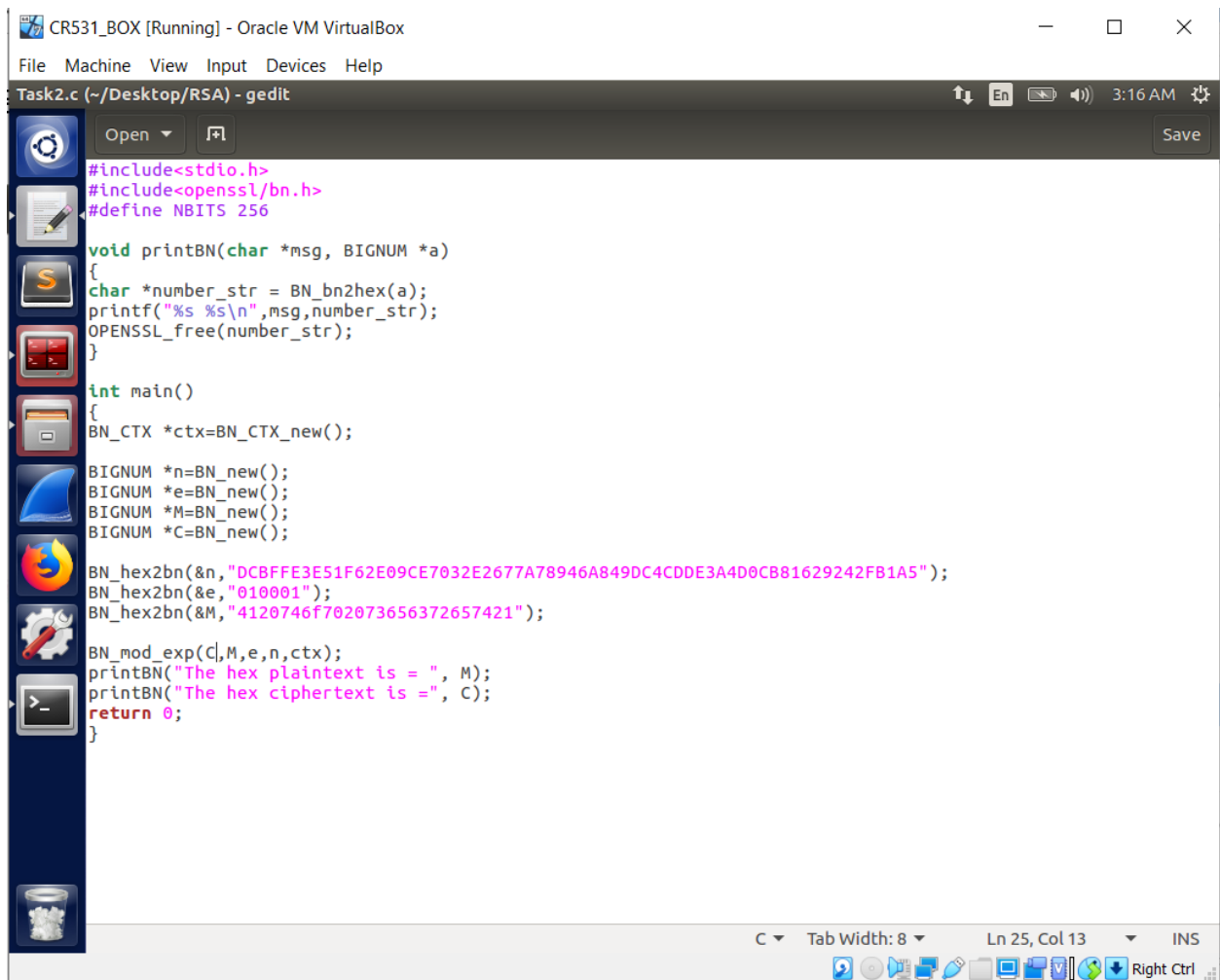
## Task 2: Encrypting a Message

**Deliverable.** Your code should print out "The hex plaintext is ... and the hex ciphertext is ...", where the dots should be replaced with the actual hex values.

```
$ python -c 'print("A top secret!".encode("hex"))'
```

```
CR531_BOX [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
[04/03/20]seed@VM:~/.../RSA$ python -c 'print("A top secret!".encode("hex"))'
4120746f702073656372657421
[04/03/20]seed@VM:~/.../RSA$
```

## Code:



The screenshot shows a gedit editor window titled "Task2.c (~/.Desktop/RSA) - gedit" running inside an Oracle VM VirtualBox. The code implements RSA encryption using the OpenSSL library. It includes headers for stdio and openssl/bn.h, and defines NBITS as 256. A printBN function is defined to print a BIGNUM in hexadecimal. The main function initializes a BN\_CTX, creates BIGNUM variables n, e, M, and C, and sets them with specific hexadecimal values. It then performs modular exponentiation (BN\_mod\_exp) and prints the resulting ciphertext C.

```
#include<stdio.h>
#include<openssl/bn.h>
#define NBITS 256

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();

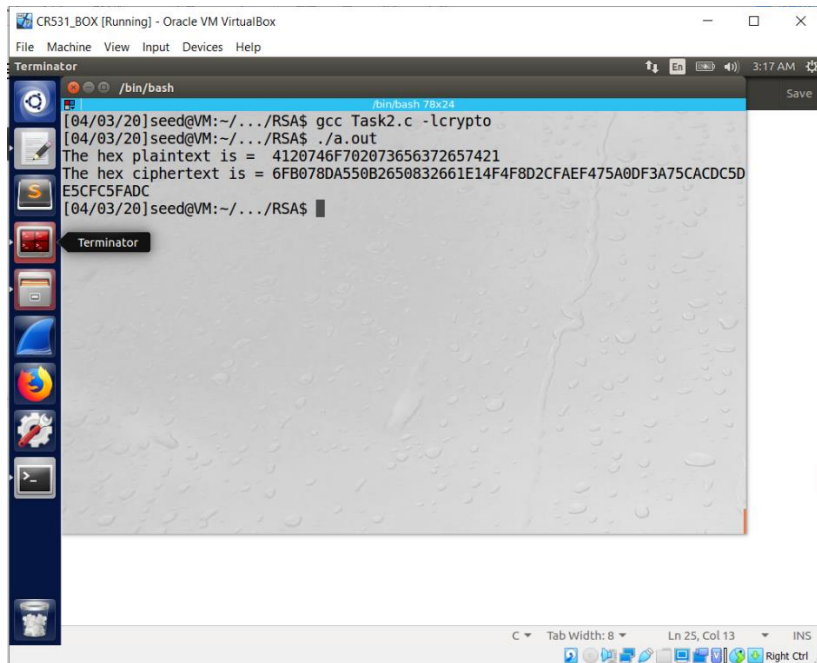
    BIGNUM *n=BN_new();
    BIGNUM *e=BN_new();
    BIGNUM *M=BN_new();
    BIGNUM *C=BN_new();

    BN_hex2bn(&n,"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&e,"010001");
    BN_hex2bn(&M,"4120746f702073656372657421");

    BN_mod_exp(C,M,e,n,ctx);
    printBN("The hex plaintext is = ", M);
    printBN("The hex ciphertext is =", C);
    return 0;
}
```

The status bar at the bottom indicates "Tab Width: 8", "Ln 25, Col 13", and "INS" mode. The system tray shows various icons including a clock, network, and volume.

## Output:

A screenshot of an Oracle VM VirtualBox window titled 'CR531\_BOX [Running]'. Inside the window is a Linux desktop environment with a blue sidebar containing icons for applications like a terminal, file manager, and web browser. A terminal window is open, showing the following text:

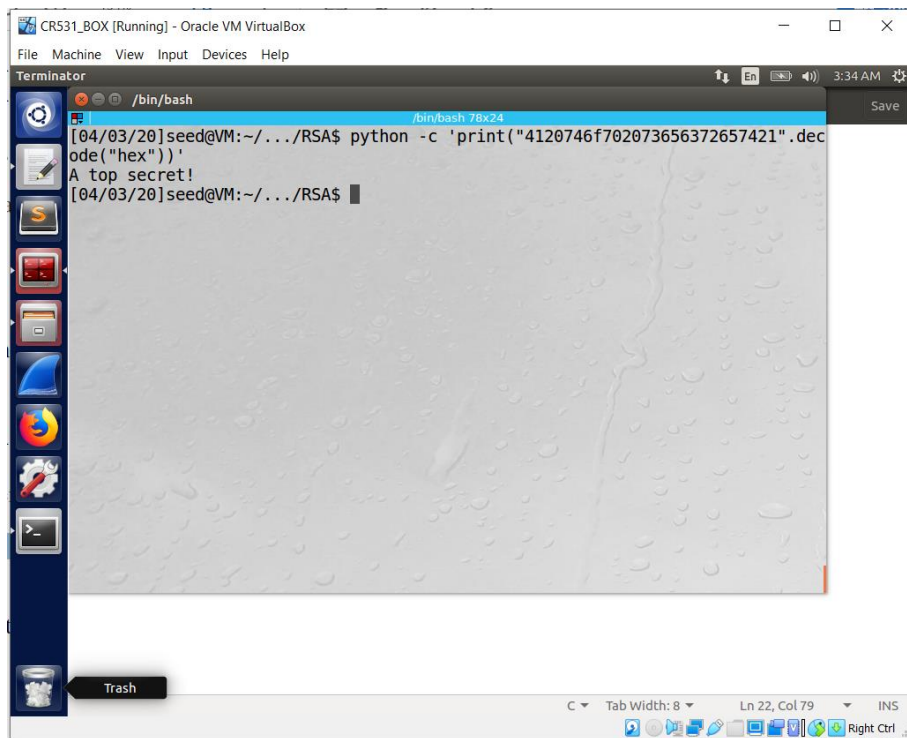
```
/bin/bash
[04/03/20]seed@VM:~/.../RSA$ gcc Task2.c -lcrypto
[04/03/20]seed@VM:~/.../RSA$ ./a.out
The hex plaintext is = 4120746f702073656372657421
The hex ciphertext is = 6fb078da550b2650832661e14f4f8d2cfaef475a0df3a75cacdc5d
E5CFC5FADC
[04/03/20]seed@VM:~/.../RSA$
```

The terminal window has a title bar that says 'Terminator' and a 'Save' button. The status bar at the bottom of the terminal shows 'C Tab Width: 8 Ln 25, Col 13 INS'.

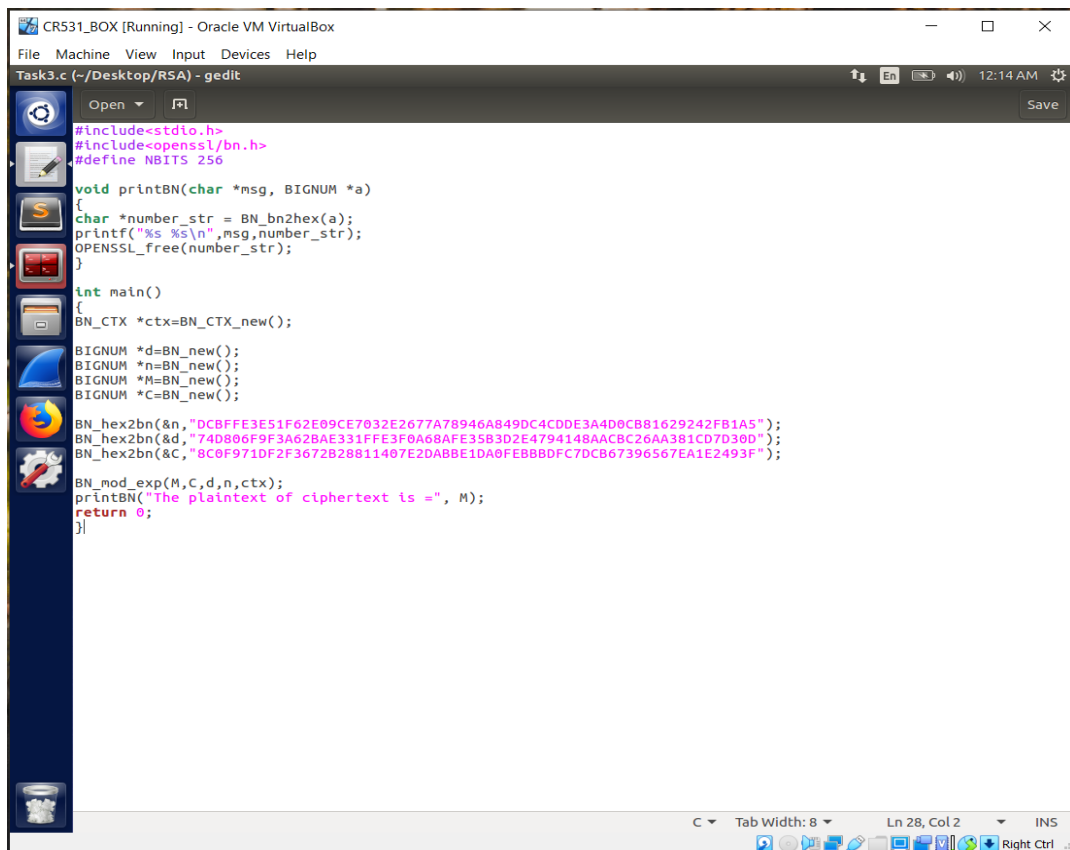
## Task 3: Decrypting a Message

**Deliverable.** Your code should print out "The plaintext of ciphertext ... is ..." where the dots should be replaced with the actual values.

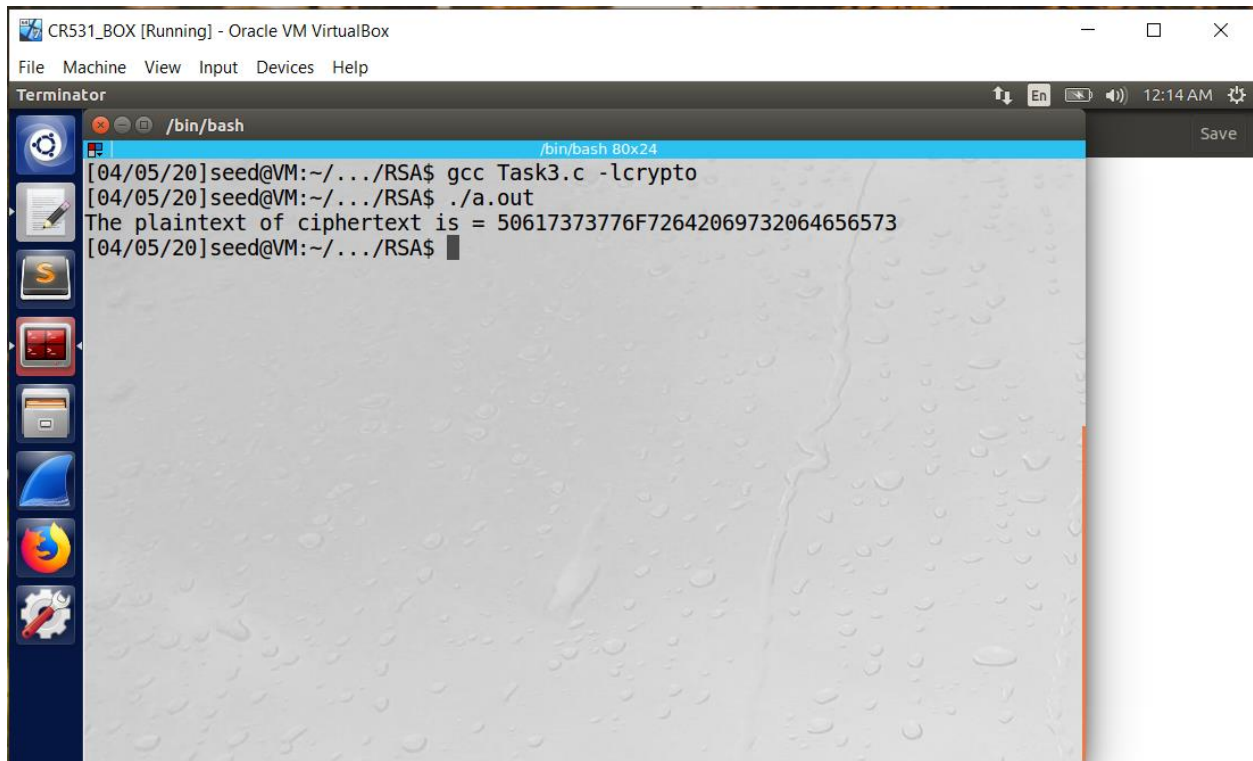
```
$ python -c 'print("4120746f702073656372657421".decode("hex"))'
```



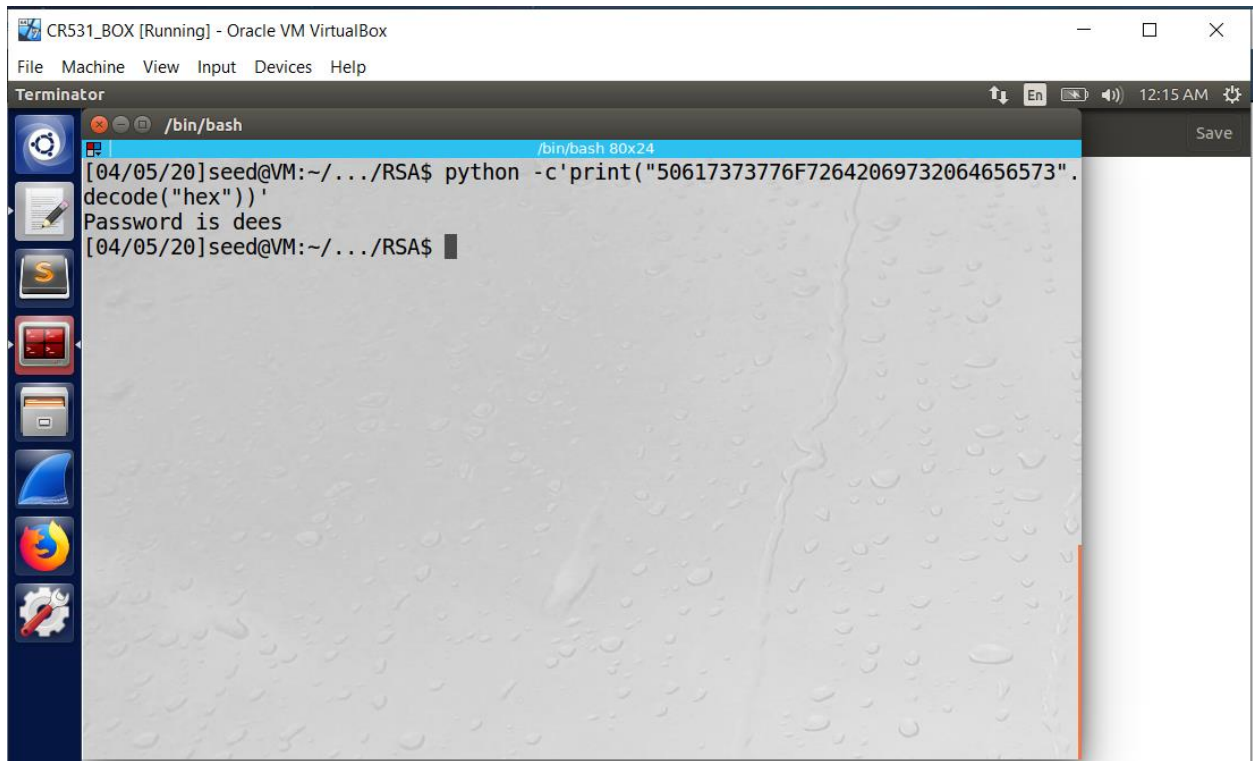
## Code:



## Output:



```
CR531_BOX [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
/bin/bash 80x24
[04/05/20]seed@VM:~/.../RSA$ gcc Task3.c -lcrypto
[04/05/20]seed@VM:~/.../RSA$ ./a.out
The plaintext of ciphertext is = 50617373776F72642069732064656573
[04/05/20]seed@VM:~/.../RSA$
```



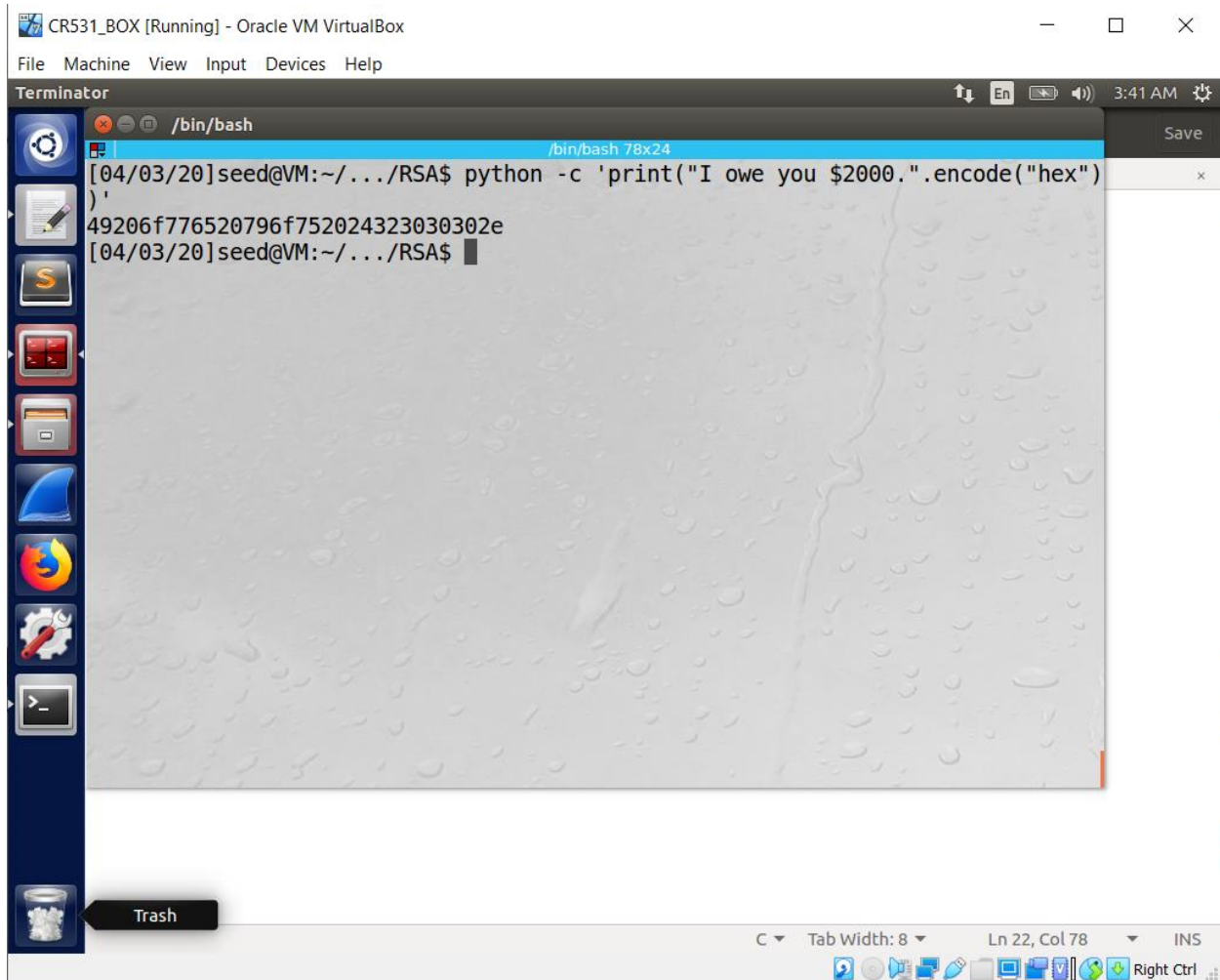
```
CR531_BOX [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
/bin/bash 80x24
[04/05/20]seed@VM:~/.../RSA$ python -c'print("50617373776F72642069732064656573".
decode("hex"))'
Password is dees
[04/05/20]seed@VM:~/.../RSA$
```

#### Task 4: Signing a Message

M = I owe you \$2000.

**Deliverable.** Your code should print out “The signature for the message in hex is ...”, where the dots should be replaced with the actual values.

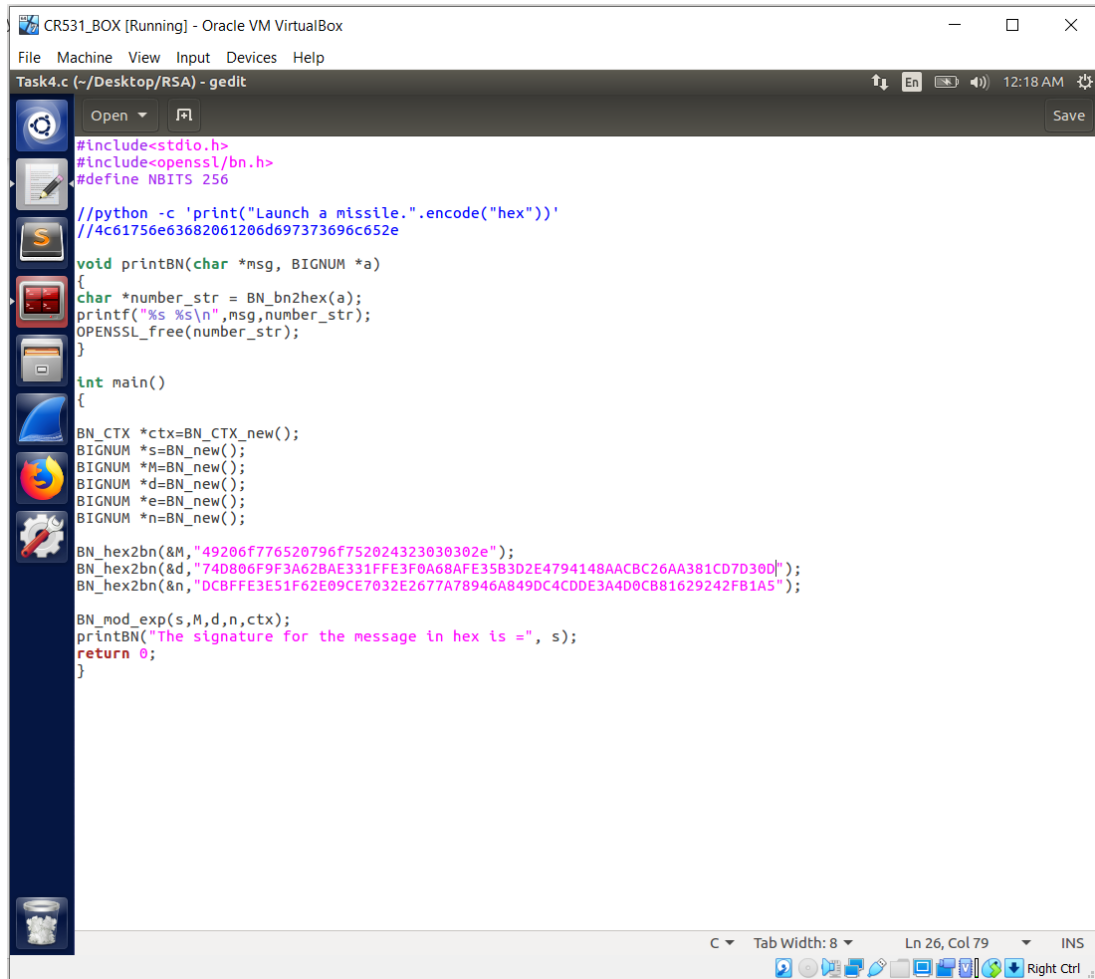
```
$ python -c 'print("I owe you $2000.".decode("hex"))'
```



```
CR531_BOX [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
/bin/bash 78x24
[04/03/20]seed@VM:~/.../RSA$ python -c 'print("I owe you $2000.".encode("hex"))'
49206f776520796f752024323030302e
[04/03/20]seed@VM:~/.../RSA$
```



## Code:



The screenshot shows a C program named Task4.c in a gedit editor within an Oracle VM VirtualBox window. The code includes headers for stdio.h and openssl/bn.h, and defines NBITS as 256. It contains a function printBN that takes a character pointer and a BIGNUM pointer, converts the BIGNUM to a hexadecimal string, and prints it. The main function initializes BN\_CTX, BIGNUM variables s, M, d, n, e, and n, and then calls BN\_hex2bn to convert hexadecimal strings to BIGNUMs. It then calls BN\_mod\_exp to calculate the signature and prints it.

```
#include<stdio.h>
#include<openssl/bn.h>
#define NBITS 256

//python -c 'print("Launch a missile.".encode("hex"))'
//4c61756e63682061206d697373696c652e

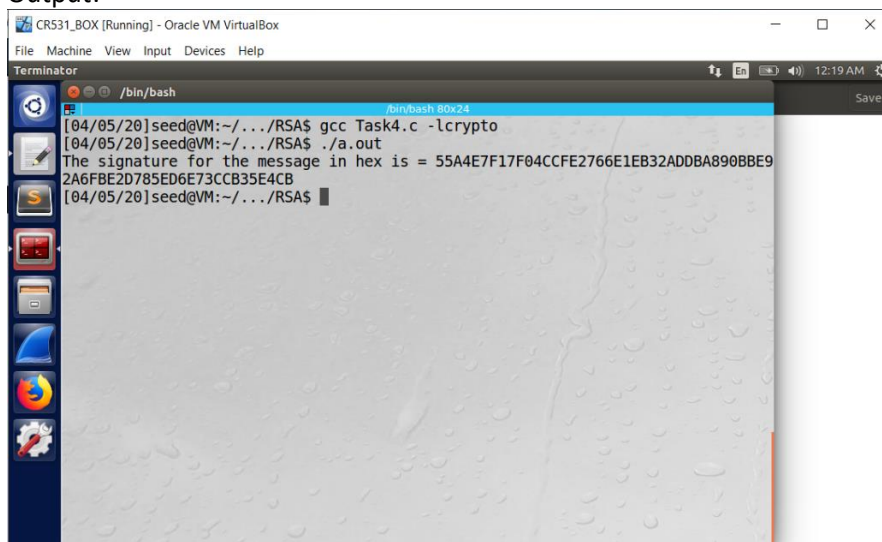
void printBN(char *msg, BIGNUM *a)
{
    char *number_str = BN_bn2hex(a);
    printf("%s %s\n",msg,number_str);
    OPENSSL_free(number_str);
}

int main()
{
    BN_CTX *ctx=BN_CTX_new();
    BIGNUM *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,"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDD0E3A4D0CB81629242FB1A5");

    BN_mod_exp(s,M,d,n,ctx);
    printBN("The signature for the message in hex is =", s);
    return 0;
}
```

## Output:



The screenshot shows the output of the program in a terminal window. The user runs the program using gcc and ./a.out. The output displays the hexadecimal message and the calculated signature.

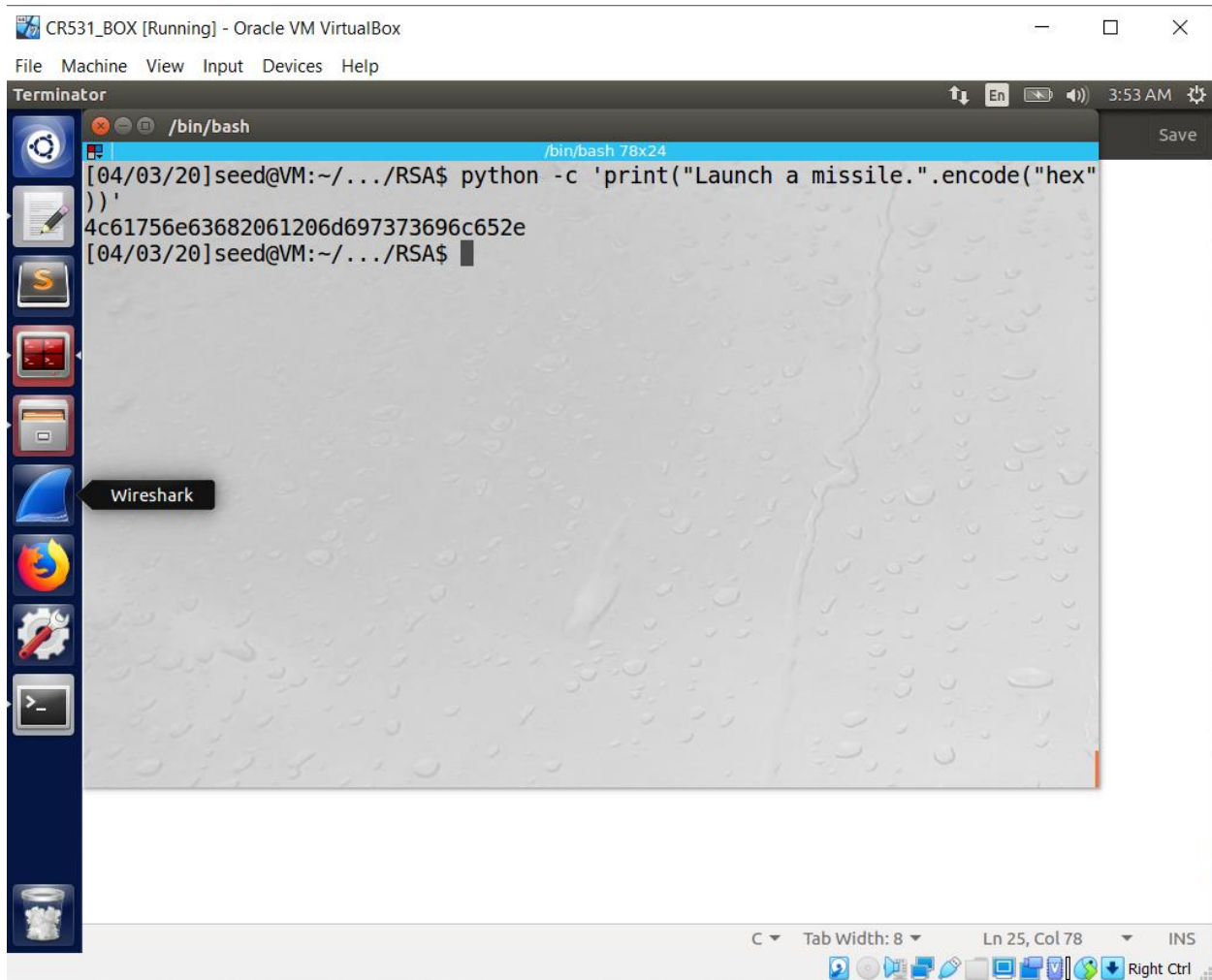
```
[04/05/20]seed@VM:~/.../RSA$ gcc Task4.c -lcrypto
[04/05/20]seed@VM:~/.../RSA$ ./a.out
The signature for the message in hex is = 55A4E7F17F04CCFE2766E1EB32ADDBA890BBE9
2A6FBE2D785ED6E73CCB35E4CB
[04/05/20]seed@VM:~/.../RSA$
```



## Task 5: Verifying a Signature

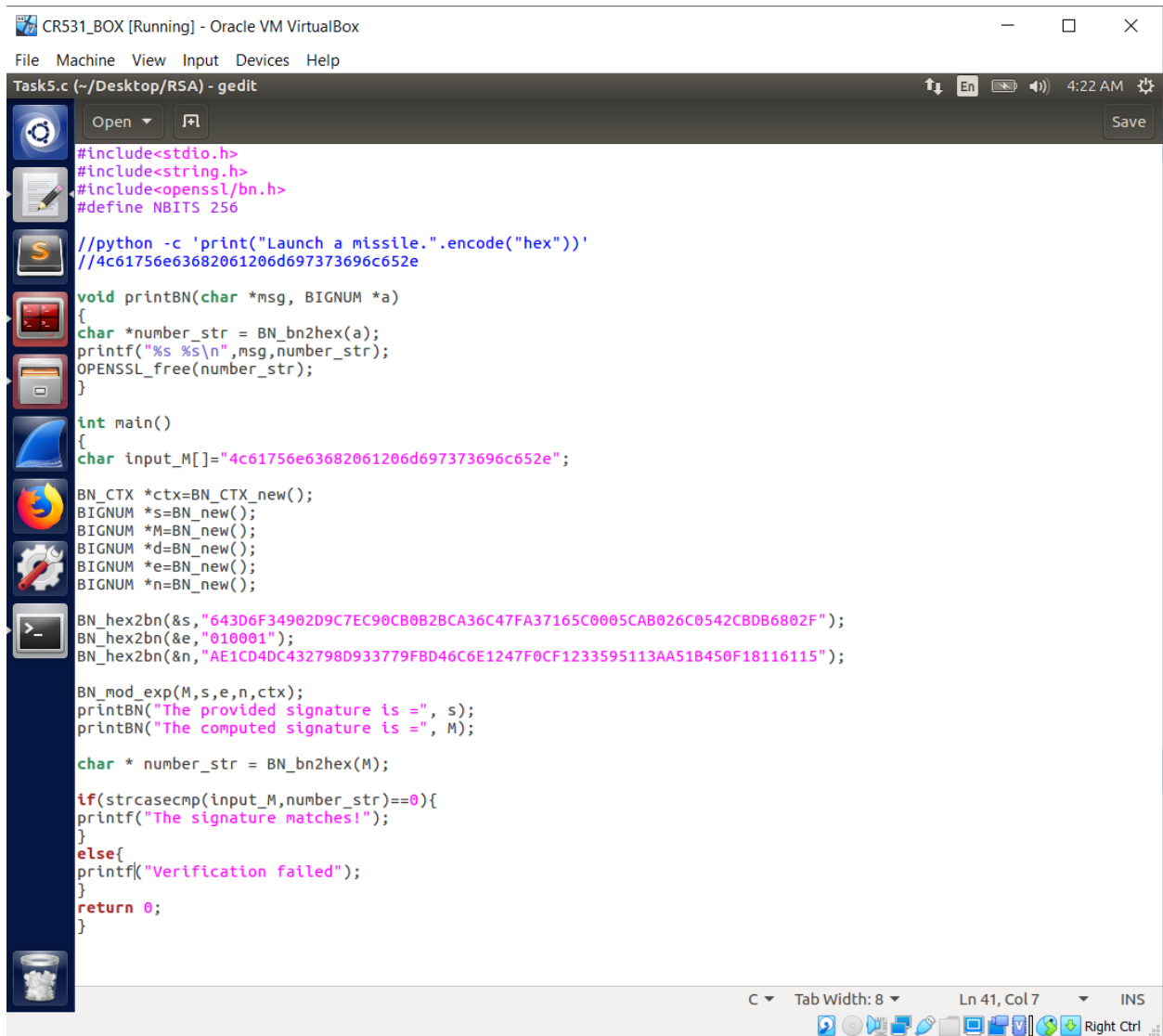
M = Launch a missile.

**Deliverable.** Your code should output the message "The provided signature is ... and the computed signature is...". Replace the dots with the actual values. If S and the computed signature are equal, output "The signature matches!", Otherwise, print "Verification failed".



```
CR531_BOX [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
/bin/bash 78x24
[04/03/20]seed@VM:~/.../RSA$ python -c 'print("Launch a missile.".encode("hex"))'
4c61756e63682061206d697373696c652e
[04/03/20]seed@VM:~/.../RSA$
```

## Code:



The screenshot shows a virtual machine window titled "CR531\_BOX [Running] - Oracle VM VirtualBox". Inside the VM, a terminal window titled "Task5.c (~/Desktop/RSA) - gedit" is open. The terminal displays a C program for RSA signature verification. The program includes headers for stdio, string, and openssl/bn.h, and defines NBITS as 256. It uses Python to print a hex string and then implements a C function printBN to print a BIGNUM in hex. The main function initializes BN\_CTX, BN, and BIGNUM variables, sets up a context, and performs a modular exponentiation. It then compares the provided signature with the computed signature and prints the result.

```
#include<stdio.h>
#include<string.h>
#include<openssl/bn.h>
#define NBITS 256

//python -c 'print("Launch a missile.".encode("hex"))'
//4c61756e63682061206d697373696c652e

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()
{
    char input_M[]="4c61756e63682061206d697373696c652e";

    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();

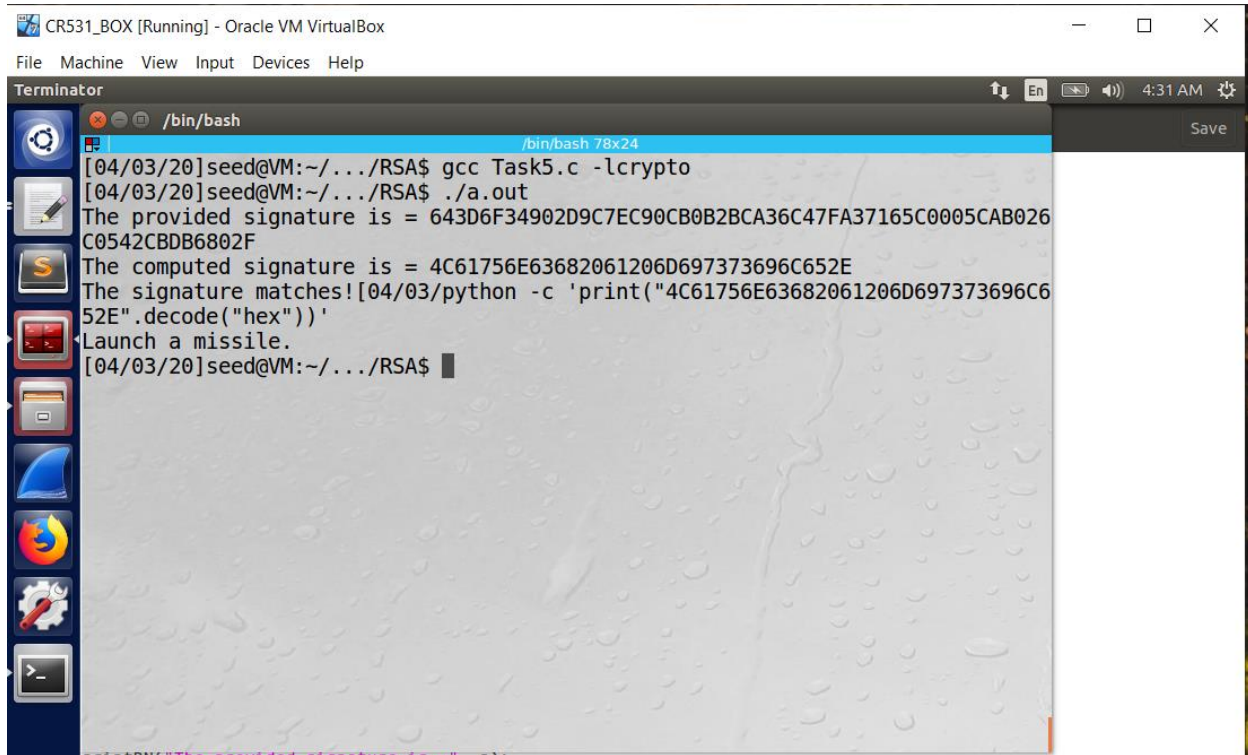
    BN_hex2bn(&s,"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBD86802F");
    BN_hex2bn(&e,"010001");
    BN_hex2bn(&n,"AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");

    BN_mod_exp(M,s,e,n,ctx);
    printBN("The provided signature is ", s);
    printBN("The computed signature is ", M);

    char * number_str = BN_bn2hex(M);

    if(strcasecmp(input_M,number_str)==0){
        printf("The signature matches!");
    }
    else{
        printf("Verification failed");
    }
    return 0;
}
```

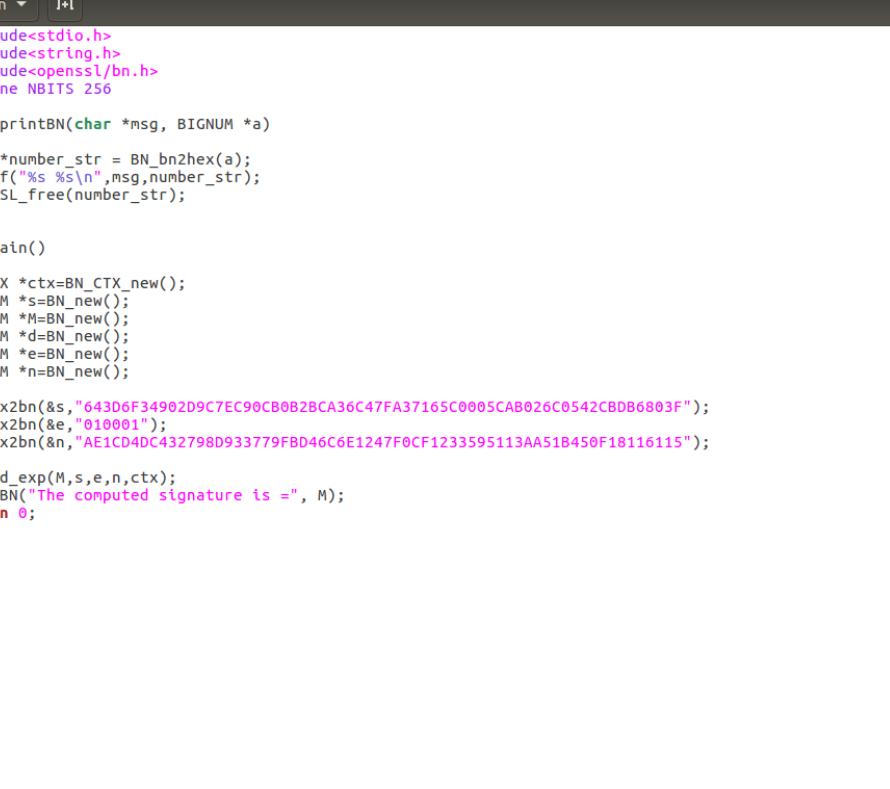
## Output:



```
CR531_BOX [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminator
/bin/bash
/bin/bash 78x24
[04/03/20]seed@VM:~/.../RSA$ gcc Task5.c -lcrypto
[04/03/20]seed@VM:~/.../RSA$ ./a.out
The provided signature is = 643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F
The computed signature is = 4C61756E63682061206D697373696C652E
The signature matches! [04/03/python -c 'print("4C61756E63682061206D697373696C652E".decode("hex"))']
Launch a missile.
[04/03/20]seed@VM:~/.../RSA$
```

Then, add another section to your code where you repeat this task but for a corrupted signature such that the last byte of the provided signature S changes from 2F to 3F, i.e, there is only one bit of change.

Code:



```
#include<stdio.h>
#include<string.h>
#include<openssl/bn.h>
#define NBITS 256

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 *s=BN_new();
    BIGNUM *M=BN_new();
    BIGNUM *d=BN_new();
    BIGNUM *e=BN_new();
    BIGNUM *n=BN_new();

    BN_hex2bn(&s,"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542C8DB6803F");
    BN_hex2bn(&e,"010001");
    BN_hex2bn(&n,"AE1CD40C432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");

    BN_mod_exp(M,s,e,n,ctx);
    printBN("The computed signature is =", M);
    return 0;
}
```

**Output:**

[illegible]