ACS 54500 Cryptography and Network Security

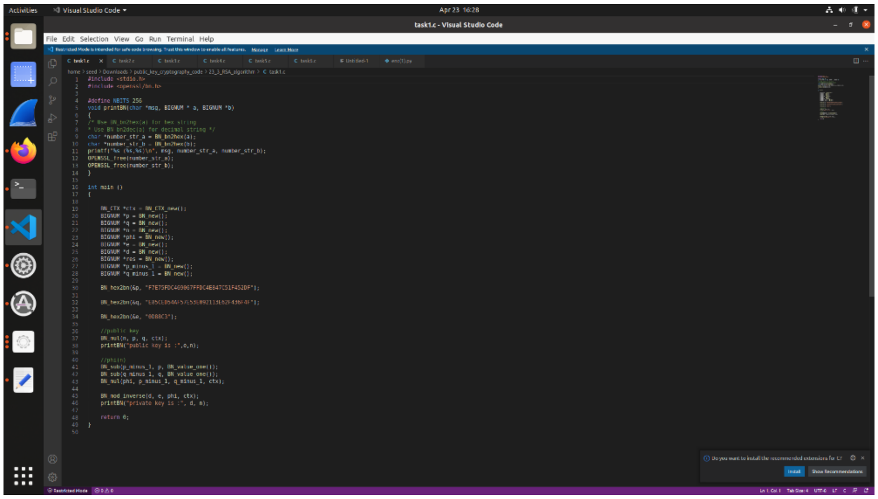
Lab 9: Lab 9: RSA Encryption and Signature Lab

Task 1: Deriving the Private Key

Let p, q, e be three prime numbers. Let n=p*q, we use (e, n) as the public key. Calculate the private key d.

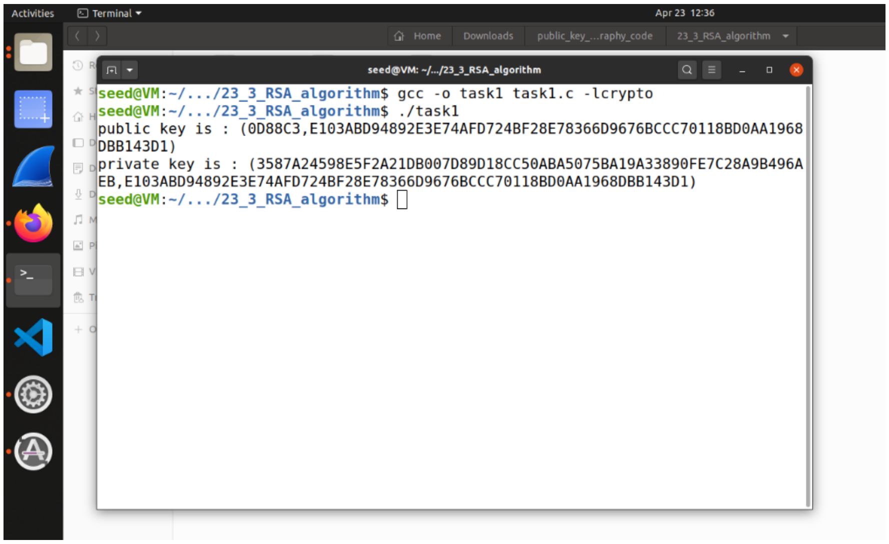
```
p = F7E75FDC469067FFDC4E847C51F452DF
q = E85CED54AF57E53E092113E62F436F4F
e = 0D88C3
```

The hexadecimal values for p, q, and e are listed below. Note that although p and q used in this task are quite large numbers, they are not large enough to be safe.



```
#include <stdio.h>
#include <openssl/bn.h>

#define NBITS 256
void printBN(char *msg, BIGNUM * a, BIGNUM *b)
{
/* Use BN_bn2hex(a) for hex string
* Use BN bn2dec(a) for decimal string */
char *number_str_a = BN_bn2hex(a);
char *number_str_b = BN_bn2hex(b);
printf("%s (%s,%s)\n", msg, number_str_a, number_str_b);
OPENSSL_free(number_str_a);
OPENSSL_free(number_str_b);
}

int main ()
{

    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *p = BN_new();
    BIGNUM *q = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *phi = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *res = BN_new();
    BIGNUM *p_minus_1 = BN_new();
    BIGNUM *q_minus_1 = BN_new();

    BN_hex2bn(&p, "F7E75FDC469067FFDC4E847C51F452DF");

    BN_hex2bn(&q, "E85CED54AF57E53E092113E62F436F4F");

    BN_hex2bn(&e, "0D88C3");

    //public key
    BN_mul(n, p, q, ctx);
    printBN("public key is :",e,n);

    //phi(n)
    BN_sub(p_minus_1, p, BN_value_one());
    BN_sub(q_minus_1, q, BN_value_one());
    BN_mul(phi, p_minus_1, q_minus_1, ctx);

    BN_mod_inverse(d, e, phi, ctx);
    printBN("private key is :", d, n);

    return 0;
}
```

Activities
Visual Studio Code
task1.c - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
task1.c
home > seed > Downloads > public_key_cryptography_code > 23_3_RSA_algorithm > task1.c
Restricted Mode

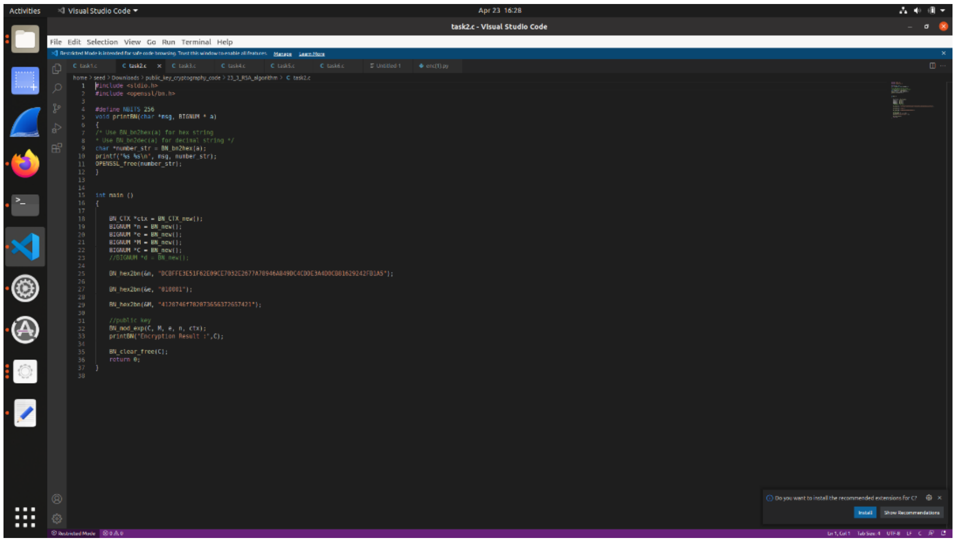


```
seed@VM:~/.../23_3_RSA_algorithm$ gcc -o task1 task1.c -lcrypto
seed@VM:~/.../23_3_RSA_algorithm$ ./task1
public key is : (0D88C3,E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968
DBB143D1)
private key is : (3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496A
EB,E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1)
seed@VM:~/.../23_3_RSA_algorithm$ 
```

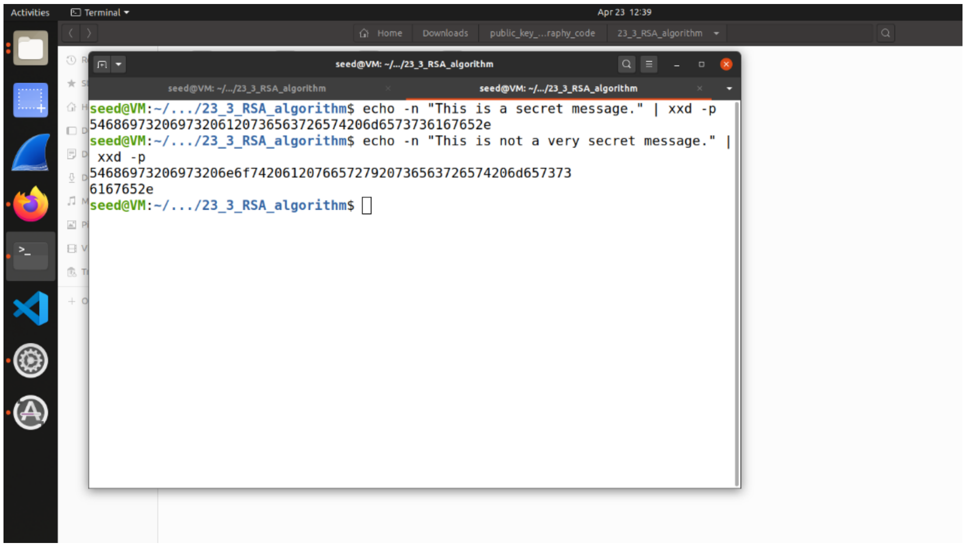
Task 2: Encrypting a Message:

Let (e, n) be the public key. Please encrypt the message

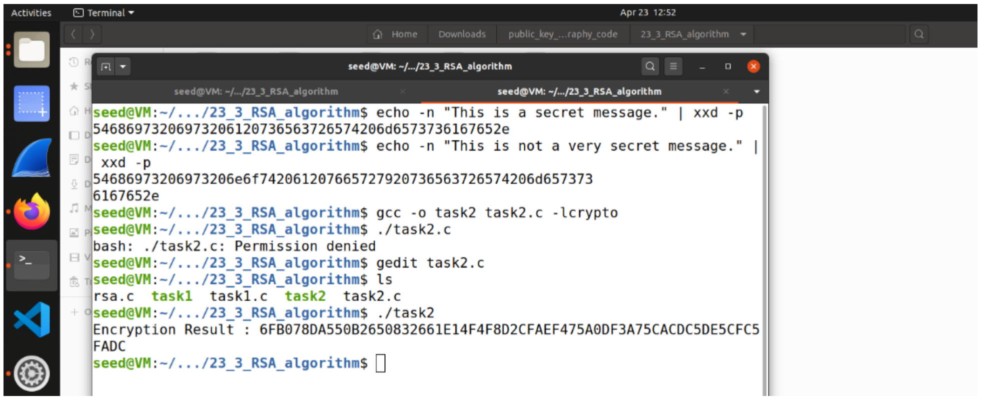
We can encrypt the message by using echo -n “ ”| xxd -p



```
#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 *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *M = BN_new();
    BIGNUM *C = BN_new();
    //BIGNUM *d = BN_new();

    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");

    BN_hex2bn(&e, "010001");

    BN_hex2bn(&M, "4120746f702073656372657421");

    //public key
    BN_mod_exp(C, M, e, n, ctx);
    printBN("Encryption Result :",C);

    BN_clear_free(C);
    return 0;
}
```



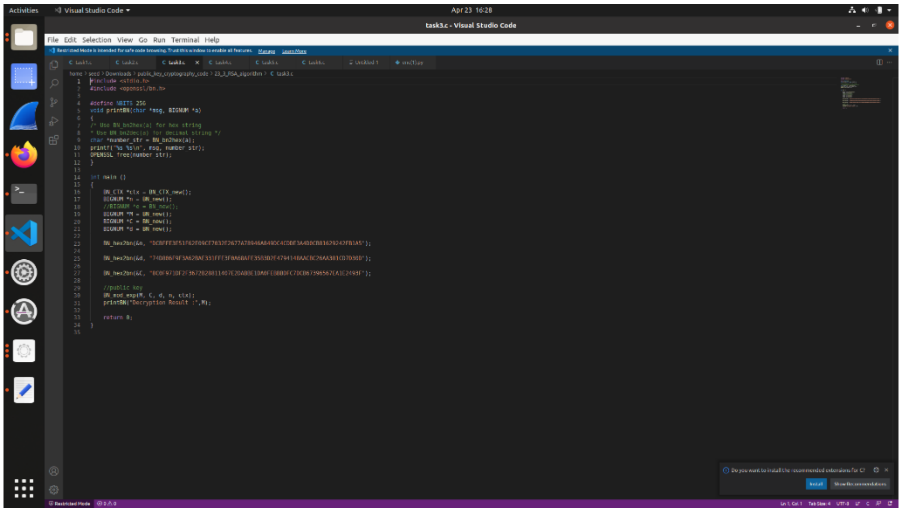
```
seed@VM:~/.../23_3_RSA_algorithm$ echo -n "This is a secret message." | xxd -p
546869732069732061207365637265742206d6573736167652e
seed@VM:~/.../23_3_RSA_algorithm$ echo -n "This is not a very secret message." |
 xxd -p
546869732069732066f742061207665727920736563726574206d657373
6167652e
seed@VM:~/.../23_3_RSA_algorithm$ 
```



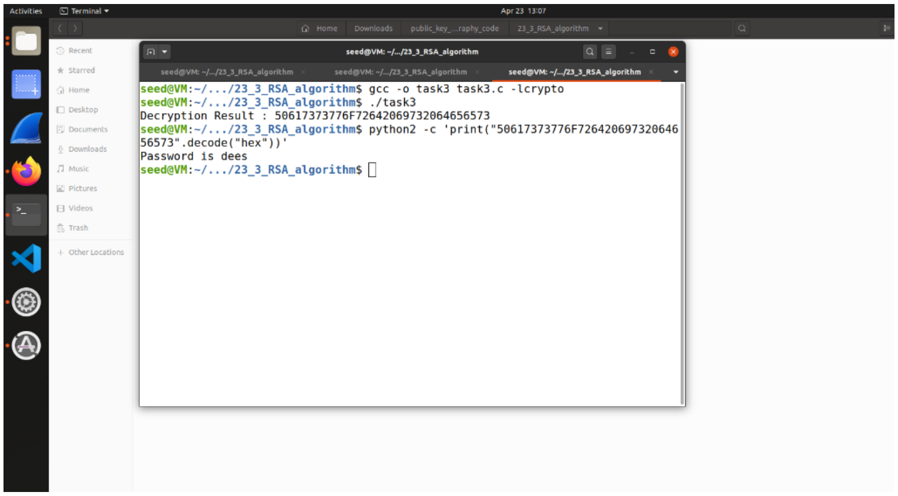
```
seed@VM:~/.../23_3_RSA_algorithm$ echo -n "This is a secret message." | xxd -p
546869732069732061207365637265742206d6573736167652e
seed@VM:~/.../23_3_RSA_algorithm$ echo -n "This is not a very secret message." |
 xxd -p
546869732069732066f742061207665727920736563726574206d657373
6167652e
seed@VM:~/.../23_3_RSA_algorithm$ gcc -o task2 task2.c -lcrypto
seed@VM:~/.../23_3_RSA_algorithm$ ./task2.c
bash: ./task2.c: Permission denied
seed@VM:~/.../23_3_RSA_algorithm$ gedit task2.c
seed@VM:~/.../23_3_RSA_algorithm$ ls
rsa.c  task1  task1.c  task2  task2.c
seed@VM:~/.../23_3_RSA_algorithm$ ./task2
Encryption Result : 6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5CFC5
FADC
seed@VM:~/.../23_3_RSA_algorithm$ 
```

Task 3: Decrypting a Message:

The public/private keys used in this task are the same as the ones used in Task 2. Please decrypt the following ciphertext C, and convert it back to a plain ASCII string



```c
#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 *n = BN_new();
    //BIGNUM *e = BN_new();
    BIGNUM *M = BN_new();
    BIGNUM *C = BN_new();
    BIGNUM *d = BN_new();

    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");

    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

    BN_hex2bn(&C, "8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F");

    //public key
    BN_mod_exp(M, C, d, n, ctx);
    printBN("Decryption Result :",M);

    return 0;
}
```

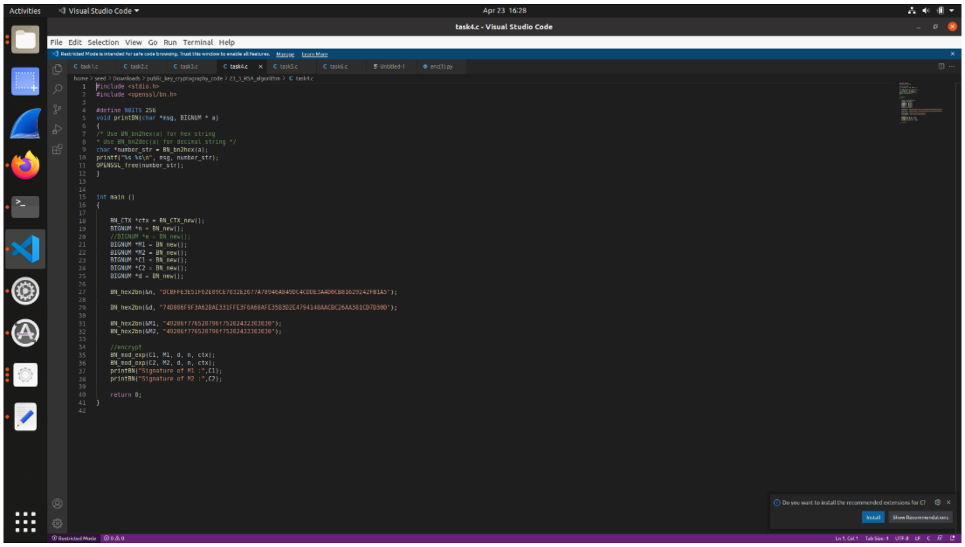


```
seed@VM:~/.../23_3_RSA_algorithm$ gcc -o task3 task3.c -lcrypto
seed@VM:~/.../23_3_RSA_algorithm$ ./task3
Decryption Result : 50617373776F72642069732064656573
seed@VM:~/.../23_3_RSA_algorithm$ python2 -c 'print("50617373776F726420697320646
56573".decode("hex"))'
Password is dees
seed@VM:~/.../23_3_RSA_algorithm$ 
```

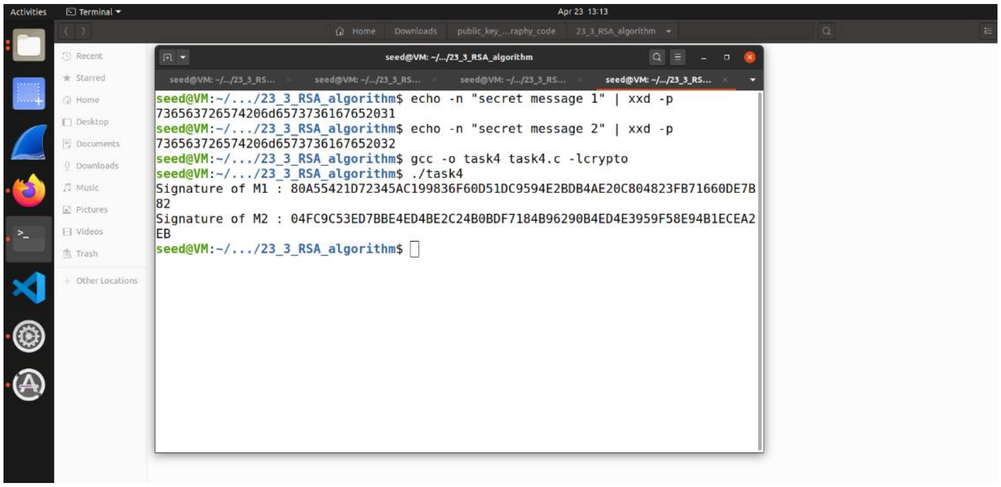
Task 4: Signing a Message

The public/private keys used in this task are the same as the ones used in Task 2.

Generating a Signature for the following message.



```c
#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 *n = BN_new();
    //BIGNUM *e = BN_new();
    BIGNUM *M1 = BN_new();
    BIGNUM *M2 = BN_new();
    BIGNUM *C1 = BN_new();
    BIGNUM *C2 = BN_new();
    BIGNUM *d = BN_new();

    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");

    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

    BN_hex2bn(&M1, "49206f776520796f75202432303030");
    BN_hex2bn(&M2, "49206f776520796f75202433303030");

    //encrypt
    BN_mod_exp(C1, M1, d, n, ctx);
    BN_mod_exp(C2, M2, d, n, ctx);
    printBN("Signature of M1 :",C1);
    printBN("Signature of M2 :",C2);

    return 0;
}
```

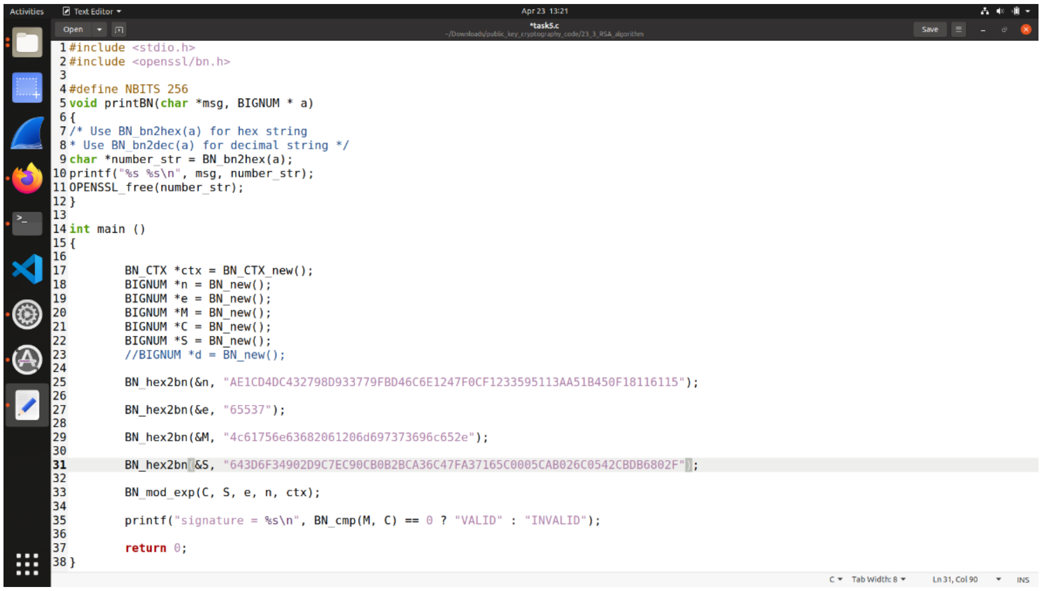


```
seed@VM:~/.../23_3_RSA_algorithm$ echo -n "secret message 1" | xxd -p
736563726574206d6573736167652031
seed@VM:~/.../23_3_RSA_algorithm$ echo -n "secret message 2" | xxd -p
736563726574206d6573736167652032
seed@VM:~/.../23_3_RSA_algorithm$ gcc -o task4 task4.c -lcrypto
seed@VM:~/.../23_3_RSA_algorithm$ ./task4
Signature of M1 : 80A55421D72345AC199836F60D51DC9594E2BDB4AE20C804823FB71660DE7B
82
Signature of M2 : 04FC9C53ED7BBE4ED4BE2C24B0BDF7184B96290B4ED4E3959F58E94B1ECEA2
EB
seed@VM:~/.../23_3_RSA_algorithm$ 
```

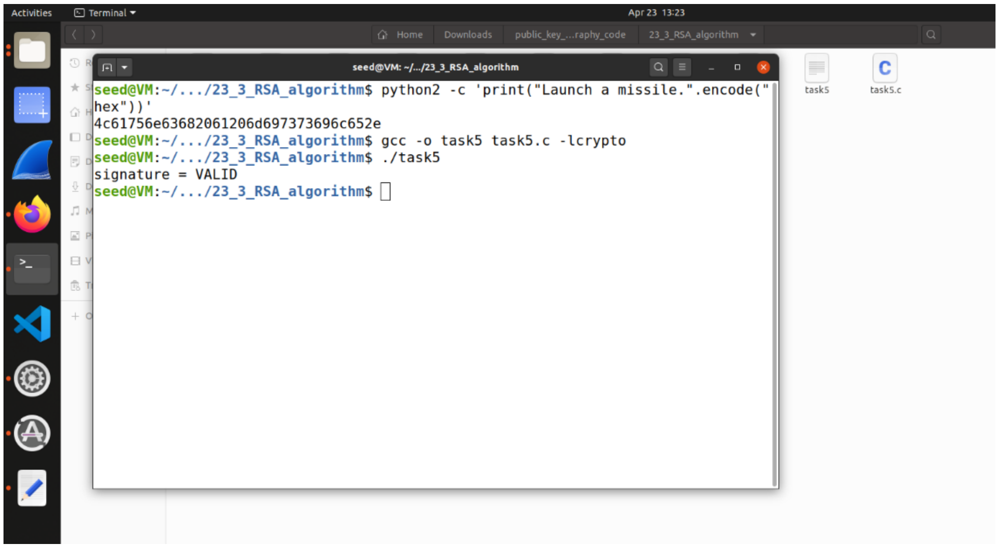
Task 5: Verifying a Signature

Bob receives a message M = "Launch a missile." from Alice, with her signature S.

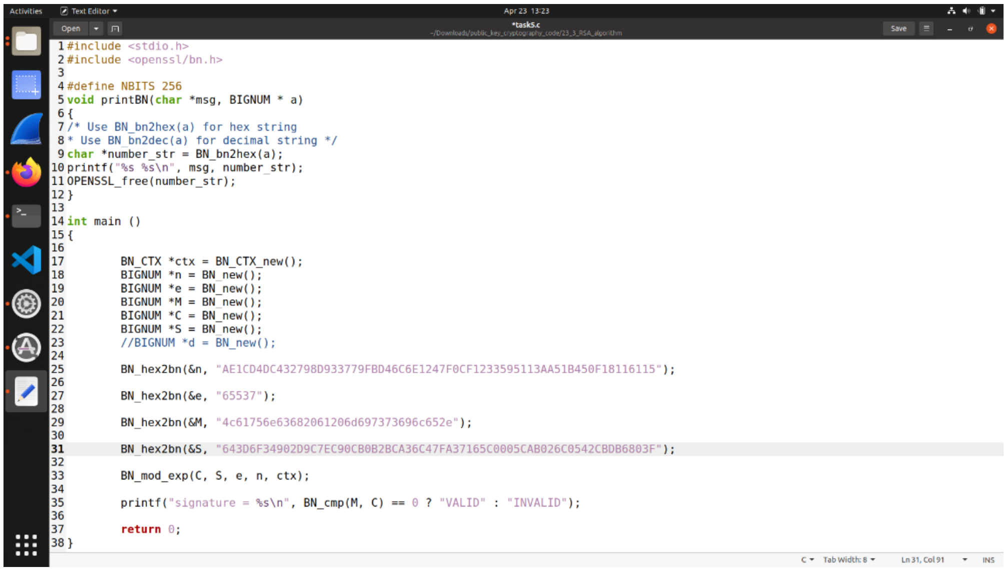
```
M = Launch a missile.
S = 643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F
e = 010001 (this hex value equals to decimal 65537)
n = AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115
```



```
#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 *n = BN_new();
        BIGNUM *e = BN_new();
        BIGNUM *M = BN_new();
        BIGNUM *C = BN_new();
        BIGNUM *S = BN_new();
        //BIGNUM *d = BN_new();

        BN_hex2bn(&n, "AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");

        BN_hex2bn(&e, "65537");

        BN_hex2bn(&M, "4c61756e63682061206d697373696c652e");

        BN_hex2bn(&S, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F");

        BN_mod_exp(C, S, e, n, ctx);

        printf("signature = %s\n", BN_cmp(M, C) == 0 ? "VALID" : "INVALID");

        return 0;
}
```

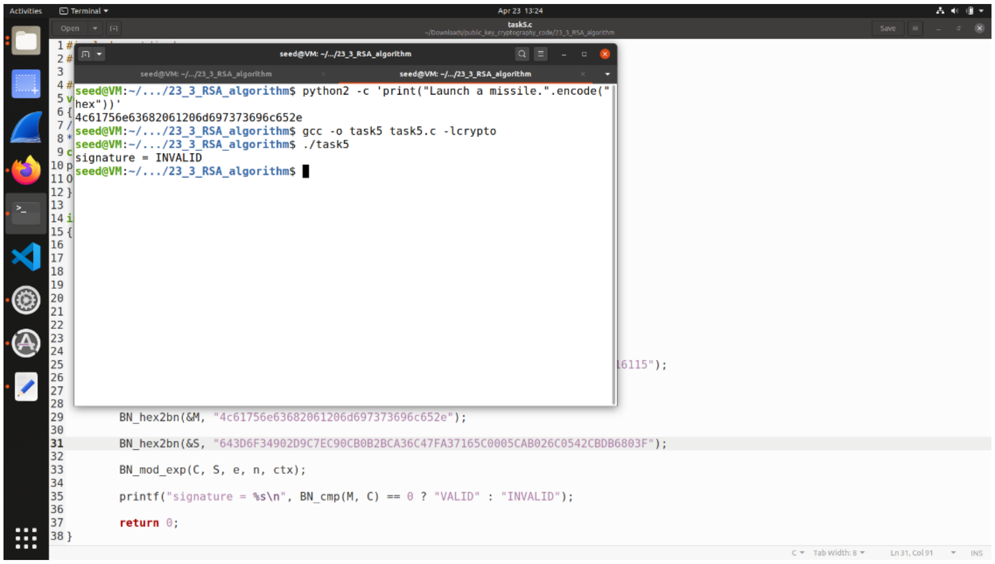


```
seed@VM:~/.../23_3_RSA_algorithm$ python2 -c 'print("Launch a missile.".encode("
hex"))'
4c61756e63682061206d697373696c652e
seed@VM:~/.../23_3_RSA_algorithm$ gcc -o task5 task5.c -lcrypto
seed@VM:~/.../23_3_RSA_algorithm$ ./task5
signature = VALID
seed@VM:~/.../23_3_RSA_algorithm$
```



```c
#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 *n = BN_new();
        BIGNUM *e = BN_new();
        BIGNUM *M = BN_new();
        BIGNUM *C = BN_new();
        BIGNUM *S = BN_new();
        //BIGNUM *d = BN_new();

        BN_hex2bn(&n, "AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");

        BN_hex2bn(&e, "65537");

        BN_hex2bn(&M, "4c61756e63682061206d697373696c652e");

        BN_hex2bn(&S, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F");

        BN_mod_exp(C, S, e, n, ctx);

        printf("signature = %s\n", BN_cmp(M, C) == 0 ? "VALID" : "INVALID");

        return 0;
}
```

Here we can observe that when we changed the hash signature value from 2F to 3F. It shows the verification fails



Activities
Terminal
Apr 23 13:24
Open
task5.c
~/Downloads/public_key_cryptography_code/23_3_RSA_algorithm
Save
seed@VM: ~/.../23_3_RSA_algorithm
seed@VM: ~/.../23_3_RSA_algorithm
seed@VM: ~/.../23_3_RSA_algorithm
C
Tab Width: 8
Ln 31, Col 91
INS

```
seed@VM:~/.../23_3_RSA_algorithm$ python2 -c 'print("Launch a missile.".encode("
hex"))'
4c61756e63682061206d697373696c652e
seed@VM:~/.../23_3_RSA_algorithm$ gcc -o task5 task5.c -lcrypto
seed@VM:~/.../23_3_RSA_algorithm$ ./task5
signature = INVALID
seed@VM:~/.../23_3_RSA_algorithm$
```

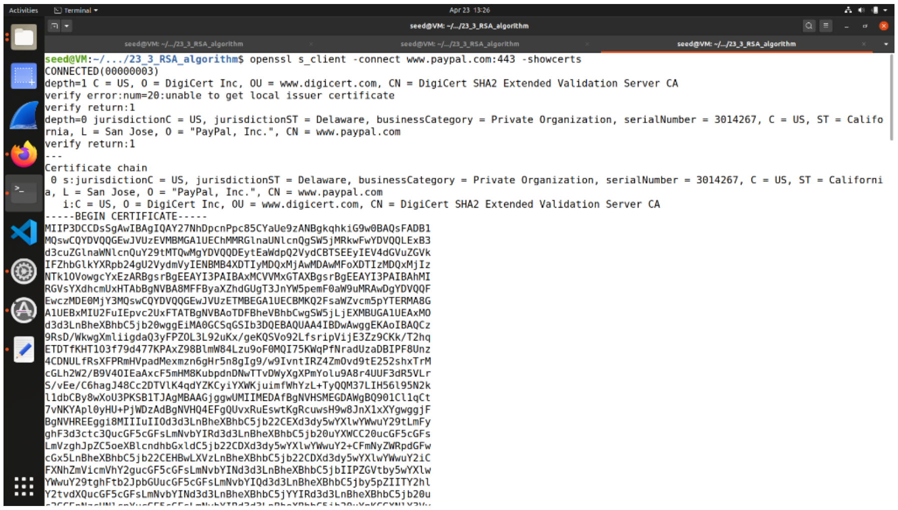
```c
16115");

        BN_hex2bn(&M, "4c61756e63682061206d697373696c652e");

        BN_hex2bn(&S, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F");

        BN_mod_exp(C, S, e, n, ctx);

        printf("signature = %s\n", BN_cmp(M, C) == 0 ? "VALID" : "INVALID");

        return 0;
}
```

Task 6: Manually Verifying an X.509 Certificate:

Downloading a certificate from www.paypal.com Copy and paste each of the certificate between the Begin Certificate and the line containing "END CERTIFICATE" to a file and saved as first one C0.pem and the C1.pem

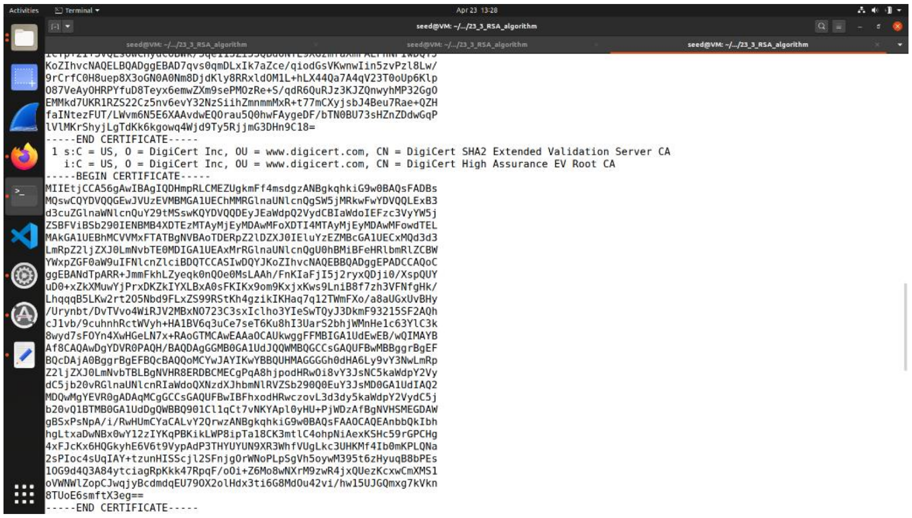
Openssl provides commands to extract certain attributes from x509 certificate. We can extract the value of n using -modulus. There is no specific command to extract e, but we can print out all the fields and can easily find the value of e.

Here we cam find the exponent value.



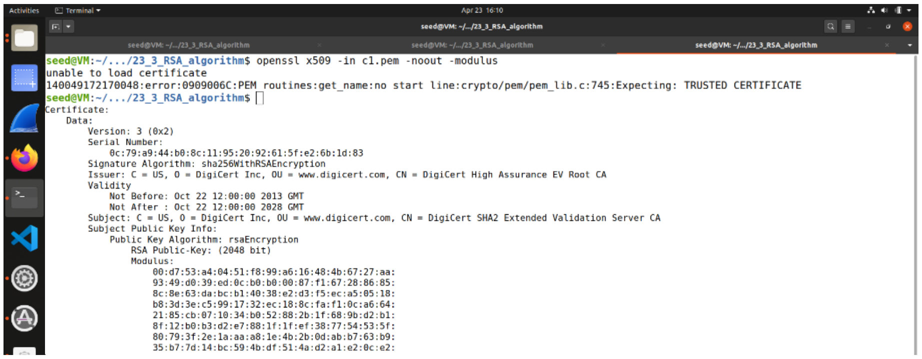
Activities
Terminal
Apr 23 13:26
seed@VM: ~/.../23_3_RSA_algorithm
seed@VM: ~/.../23_3_RSA_algorithm
seed@VM: ~/.../23_3_RSA_algorithm
seed@VM: ~/.../23_3_RSA_algorithm

```
seed@VM:~/.../23_3_RSA_algorithm$ openssl s_client -connect www.paypal.com:443 -showcerts
CONNECTED(00000003)
depth=1 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 Extended Validation Server CA
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=0 jurisdictionC = US, jurisdictionST = Delaware, businessCategory = Private Organization, serialNumber = 3014267, C = US, ST = Califo
rnia, L = San Jose, O = "PayPal, Inc.", CN = www.paypal.com
verify return:1
---
Certificate chain
 0 s:jurisdictionC = US, jurisdictionST = Delaware, businessCategory = Private Organization, serialNumber = 3014267, C = US, ST = Californi
a, L = San Jose, O = "PayPal, Inc.", CN = www.paypal.com
   i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 Extended Validation Server CA
-----BEGIN CERTIFICATE-----
MIIP3DCCDsSgAwIBAgIQAY27NhDpcnPpc85CYaUe9zANBgkqhkiG9w0BAQsFADB1
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMTQwMgYDVQQDEytEaWdpQ2VydCBTSEEyIEV4dGVuZGVk
IFZhbGlkYXRpb24gU2VydmVyIENBMB4XDTIyMDQxMjAwMDAwMFoXDTIzMDQxMjIz
NTk1OVowgcYxEzARBgsrBgEEAYI3PAIBAxMCVVMxGTAXBgsrBgEEAYI3PAIBAhMI
RGVsYXdhcmUxHTAbBgNVBA8MFFByaXZhdGUgT3JnYW5pemF0aW9uMRAwDgYDVQQF
EwczMDE0MjY3MQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8G
A1UEBxMIU2FuIEpvc2UxFTATBgNVBAoTDFBheVBhbCwgSW5jLjEXMBUGA1UEAxMO
d3d3LnBheXBhbC5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCz
9RsD/WkwgXmliigdaQ3yFPZOL3L92uKx/geKQSVo92LfsripVijE3Zz9CKk/T2hq
ETDTfKHT1O3f79d477KPAxZ98BlmW84Lzu9oF0MQI75KWqPfNradUzaDBIPF8Unz
4CDNULfRsXFPRmHVpadMexmzn6gHr5n8gIg9/w9IvntIRZ4ZmOvd9tE252shxTrM
cGLh2W2/B9V4OIEaAxcF5mHM8KubpdnDNwTTvDWyXgXPmYolu9A8r4UUF3dR5VLr
S/vEe/C6hagJ48Cc2DTVlK4qdYZKCyiYXWKjuimfWhYzL+TyQQM37LIH56l95N2k
l1dbCBy8wXoU3PKSB1TJAgMBAAGjggwUMIIMEDAfBgNVHSMEGDAWgBQ901Cl1qCt
7vNKYApl0yHU+PjWDzAdBgNVHQ4EFgQUvxRuEswtKgRcuwsH9w8JnX1xXYgwggjF
BgNVHREEggi8MIIIuIIOd3d3LnBheXBhbC5jb22CEXd3dy5wYXlwYWwuY29tLmFy
ghF3d3ctc3QucGF5cGFsLmNvbYIRd3d3LnBheXBhbC5jb20uYXWCC20ucGF5cGFs
LmVzghJpZC5oeXBlcndhbGxldC5jb22CDXd3dy5wYXlwYWwuY2+CFmNyZWRpdGFw
cGx5LnBheXBhbC5jb22CEHBwLXVzLnBheXBhbC5jb22CDXd3dy5wYXlwYWwuY2iC
FXNhZmVicmVhY2gucGF5cGFsLmNvbYINd3d3LnBheXBhbC5jbIIPZGVtby5wYXlw
YWwuY29tghFtb2JpbGUucGF5cGFsLmNvbYIQd3d3LnBheXBhbC5jby5pZIITY2hl
Y2tvdXQucGF5cGFsLmNvbYINd3d3LnBheXBhbC5jYYIRd3d3LnBheXBhbC5jb20u
```



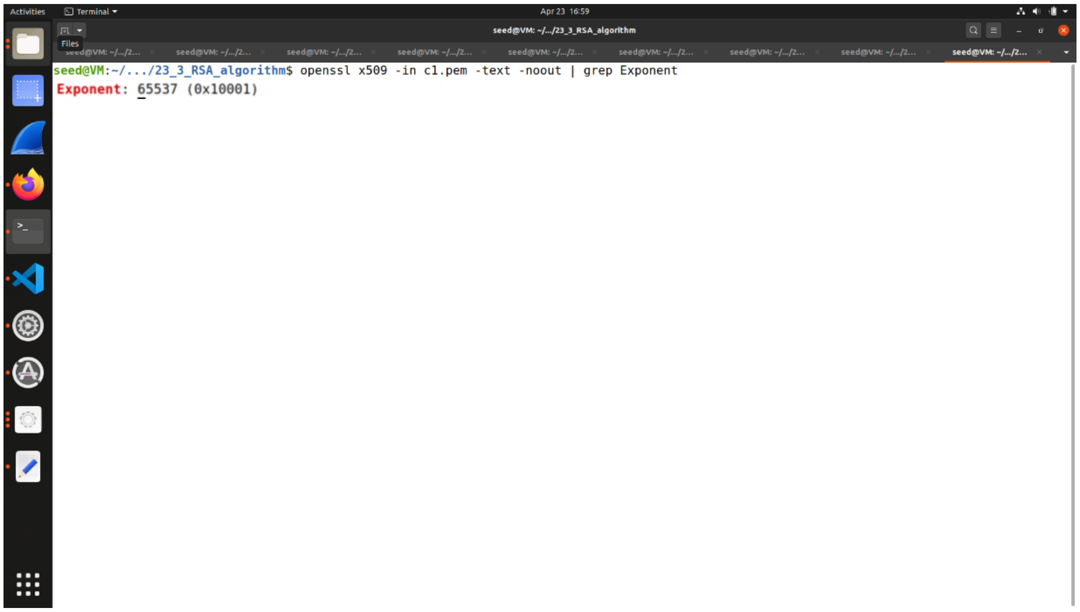
Activities
Terminal
Apr 23 13:28
seed@VM: ~/.../23_3_RSA_algorithm
seed@VM: ~/.../23_3_RSA_algorithm
seed@VM: ~/.../23_3_RSA_algorithm
seed@VM: ~/.../23_3_RSA_algorithm

```
KoZIhvcNAQELBQADggEBAD7qvs0qmDLxIk7aZce/qiodGsVKwnwIin5zvPzl8Lw/
9rCrfC0H8uep8X3oGN0A0Nm8DjdKly8RRxldOM1L+hLX44Qa7A4qV23T0oUp6Klp
O87VeAyOHRPYfuD8Teyx6emwZXm9sePMOzRe+S/qdR6QuRJz3KJZQnwyhMP32GgO
EMMkd7UKR1RZS22Cz5nv6evY32NzSiihZmnmmMxR+t77mCXyjsbJ4Beu7Rae+QZH
faINtezFUT/LWvm6N5E6XAAvdwEQOrau5Q0hwFAygeDF/bTN0BU73sHZnZDdwGqP
lVlMKrShyjLgTdKk6kgowq4Wjd9Ty5RjjmG3DHn9C18=
-----END CERTIFICATE-----
 1 s:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 Extended Validation Server CA
   i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert High Assurance EV Root CA
-----BEGIN CERTIFICATE-----
MIIEtjCCA56gAwIBAgIQDHmpRLCMEZUgkmFf4msdgzANBgkqhkiG9w0BAQsFADBs
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMSswKQYDVQQDEyJEaWdpQ2VydCBIaWdoIEFzc3VyYW5j
ZSBFViBSb290IENBMB4XDTEzMTAyMjEyMDAwMFoXDTI4MTAyMjEyMDAwMFowdTEL
MAkGA1UEBhMCVVMxFTATBgNVBAoTDERpZ2lDZXJ0IEluYzEZMBcGA1UECxMQd3d3
LmRpZ2ljZXJ0LmNvbTE0MDIGA1UEAxMrRGlnaUNlcnQgU0hBMiBFeHRlbmRlZCBW
YWxpZGF0aW9uIFNlcnZlciBDQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
ggEBANdTpARR+JmmFkhLZyeqk0nQOe0MsLAAh/FnKIaFjI5j2ryxQDji0/XspQUY
uD0+xZkXMuwYjPrxDKZkIYXLBxA0sFKIKx9om9KxjxKws9LniB8f7zh3VFNfgHk/
LhqqqB5LKw2rt2O5Nbd9FLxZS99RStKh4gzikIKHaq7q12TWmFXo/a8aUGxUvBHy
/Urynbt/DvTVvo4WiRJV2MBxNO723C3sxIclho3YIeSwTQyJ3DkmF93215SF2AQh
cJ1vb/9cuhnhRctWVyh+HA1BV6q3uCe7seT6Ku8hI3UarS2bhjWMnHe1c63YlC3k
8wyd7sFOYn4XwHGeLN7x+RAoGTMCAwEAAaOCAUkwggFFMBIGA1UdEwEB/wQIMAYB
Af8CAQAwDgYDVR0PAQH/BAQDAgGGMB0GA1UdJQQWMBQGCCsGAQUFBwMBBggrBgEF
BQcDAjA0BggrBgEFBQcBAQQoMCYwJAYIKwYBBQUHMAGGGGh0dHA6Ly9vY3NwLmRp
Z2ljZXJ0LmNvbTBLBgNVHR8ERDBCMECgPqA8hjpodHRwOi8vY3JsNC5kaWdpY2Vy
dC5jb20vRGlnaUNlcnRIaWdoQXNzdXJhbmNlRVZSb290Q0EuY3JsMD0GA1UdIAQ2
MDQwMgYEVR0gADAqMCgGCCsGAQUFBwIBFhxodHRwczovL3d3dy5kaWdpY2VydC5j
b20vQ1BTMB0GA1UdDgQWBBQ901Cl1qCt7vNKYApl0yHU+PjWDzAfBgNVHSMEGDAW
gBSxPsNpA/i/RwHUmCYaCALvY2QrwzANBgkqhkiG9w0BAQsFAAOCAQEAnbbQkIbh
hgLtxaDwNBx0wY12zIYKqPBKikLWP8ipTa18CK3mtlC4ohpNiAexKSHc59rGPCHg
4xFJcKx6HQGkyhE6V6t9VypAdP3THYUYUN9XR3WhfVUgLkc3UHKMf4Ib0mKPLQNa
2sPIoc4sUqIAY+tzunHISScjl2SFnjgOrWNoPLpSgVh5oywM395t6zHyuqB8bPEs
1OG9d4Q3A84ytciagRpKkk47RpqF/oOi+Z6Mo8wNXrM9zwR4jxQUezKcxwCmXMS1
oVWNWlZopCJwqjyBcdmdqEU79OX2olHdx3ti6G8MdOu42vi/hw15UJGQmxg7kVkn
8TUoE6smftX3eg==
-----END CERTIFICATE-----
```



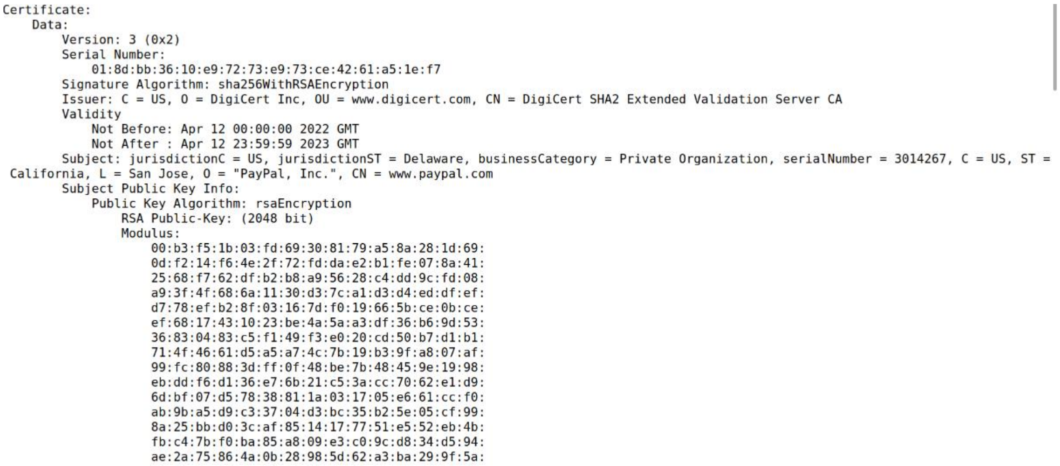
Activities
Terminal
Apr 23 16:10
seed@VM: ~/.../23_3_RSA_algorithm
seed@VM: ~/.../23_3_RSA_algorithm
seed@VM: ~/.../23_3_RSA_algorithm
seed@VM: ~/.../23_3_RSA_algorithm

```
seed@VM:~/.../23_3_RSA_algorithm$ openssl x509 -in c1.pem -noout -modulus
unable to load certificate
140049172170048:error:0909006C:PEM routines:get_name:no start line:crypto/pem/pem_lib.c:745:Expecting: TRUSTED CERTIFICATE
seed@VM:~/.../23_3_RSA_algorithm$ 
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            0c:79:a9:44:b0:8c:11:95:20:92:61:5f:e2:6b:1d:83
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert High Assurance EV Root CA
        Validity
            Not Before: Oct 22 12:00:00 2013 GMT
            Not After : Oct 22 12:00:00 2028 GMT
        Subject: C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 Extended Validation Server CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:d7:53:a4:04:51:f8:99:a6:16:48:4b:67:27:aa:
                    93:49:d0:39:ed:0c:b0:b0:00:87:f1:67:28:86:85:
                    8c:8e:63:da:bc:b1:40:38:e2:d3:f5:ec:a5:05:18:
                    b8:3d:3e:c5:99:17:32:ec:18:8c:fa:f1:0c:a6:64:
                    21:85:cb:07:10:34:b0:52:88:2b:1f:68:9b:d2:b1:
                    8f:12:b0:b3:d2:e7:88:1f:1f:ef:38:77:54:53:5f:
                    80:79:3f:2e:1a:aa:a8:1e:4b:2b:0d:ab:b7:63:b9:
                    35:b7:7d:14:bc:59:4b:df:51:4a:d2:a1:e2:0c:e2:
```

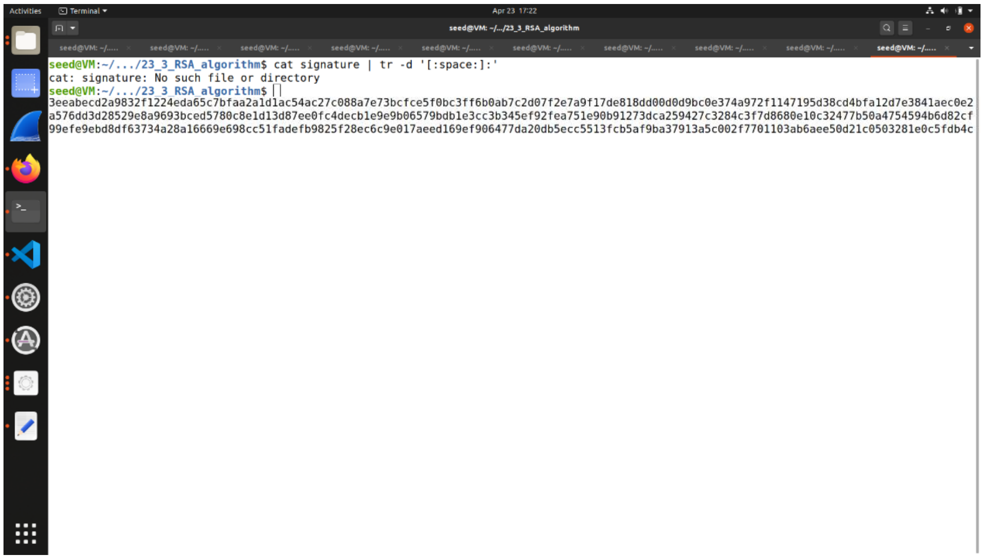


```
seed@VM:~/.../23_3_RSA_algorithm$ openssl x509 -in c1.pem -text -noout | grep Exponent
Exponent: 65537 (0x10001)
```

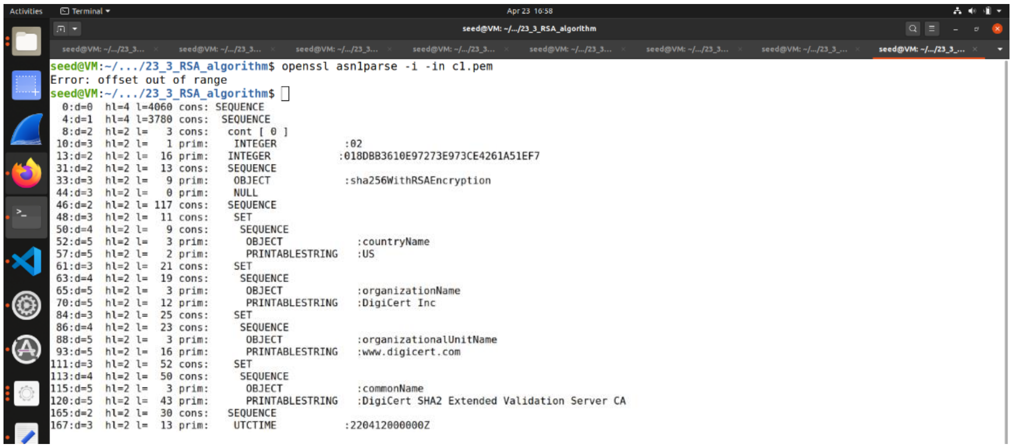
Extract the body of the server's certificate.



```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            01:8d:bb:36:10:e9:72:73:e9:73:ce:42:61:a5:1e:f7
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 Extended Validation Server CA
        Validity
            Not Before: Apr 12 00:00:00 2022 GMT
            Not After : Apr 12 23:59:59 2023 GMT
        Subject: jurisdictionC = US, jurisdictionST = Delaware, businessCategory = Private Organization, serialNumber = 3014267, C = US, ST =
 California, L = San Jose, O = "PayPal, Inc.", CN = www.paypal.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:b3:f5:1b:03:fd:69:30:81:79:a5:8a:28:1d:69:
                    0d:f2:14:f6:4e:2f:72:fd:da:e2:b1:fe:07:8a:41:
                    25:68:f7:62:df:b2:b8:a9:56:28:c4:dd:9c:fd:08:
                    a9:3f:4f:68:6a:11:30:d3:7c:a1:d3:d4:ed:df:ef:
                    d7:78:ef:b2:8f:03:16:7d:f0:19:66:5b:ce:0b:ce:
                    ef:68:17:43:10:23:be:4a:5a:a3:df:36:b6:9d:53:
                    36:83:04:83:c5:f1:49:f3:e0:20:cd:50:b7:d1:b1:
                    71:4f:46:61:d5:a5:a7:4c:7b:19:b3:9f:a8:07:af:
                    99:fc:80:88:3d:ff:0f:48:be:7b:48:45:9e:19:98:
                    eb:dd:f6:d1:36:e7:6b:21:c5:3a:cc:70:62:e1:d9:
                    6d:bf:07:d5:78:38:81:1a:03:17:05:e6:61:cc:f0:
                    ab:9b:a5:d9:c3:37:04:d3:bc:35:b2:5e:05:cf:99:
                    8a:25:bb:d0:3c:af:85:14:17:77:51:e5:52:eb:4b:
                    fb:c4:7b:f0:ba:85:a8:09:e3:c0:9c:d8:34:d5:94:
                    ae:2a:75:86:4a:0b:28:98:5d:62:a3:ba:29:9f:5a:
```

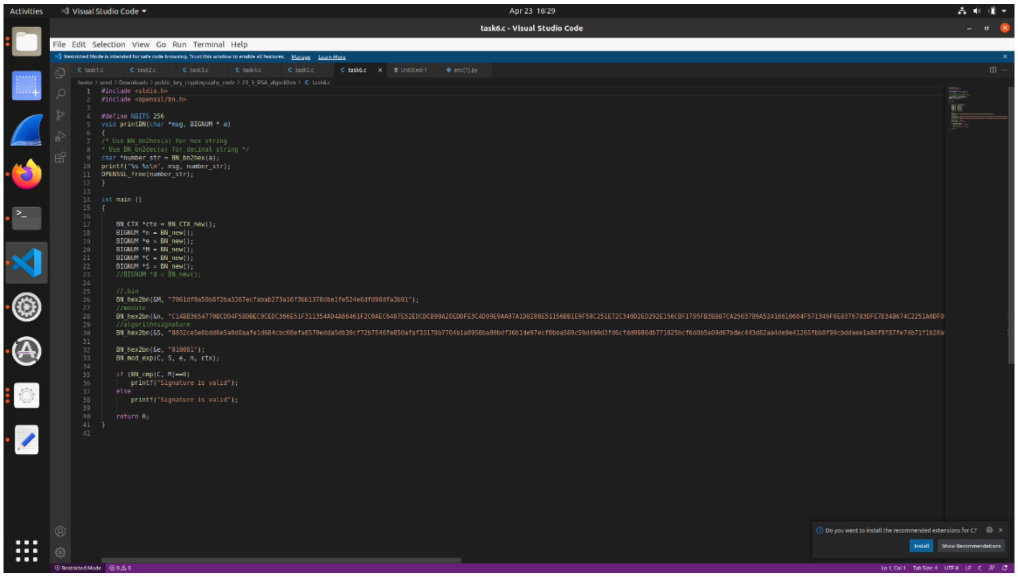


```
seed@VM:~/.../23_3_RSA_algorithm$ cat signature | tr -d '[:space:]:'
cat: signature: No such file or directory
seed@VM:~/.../23_3_RSA_algorithm$ 
3eeabecd2a9832f1224eda65c7bfaa2a1d1ac54ac27c088a7e73bcfce5f0bc3ff6b0ab7c2d07f2e7a9f17de818dd00d0d9bc0e374a972f1147195d38cd4bfa12d7e3841aec0e2
a576dd3d28529e8a9693bced5780c8e1d13d87ee0fc4decb1e9e9b06579bdb1e3cc3b345ef92fea751e90b91273dca259427c3284c3f7d8680e10c32477b50a4754594b6d82cf
99efe9ebd8df63734a28a16669e698cc51fadefb9825f28ec6c9e017aeed169ef906477da20db5ecc5513fcb5af9ba37913a5c002f7701103ab6aee50d21c0503281e0c5fdb4c
```

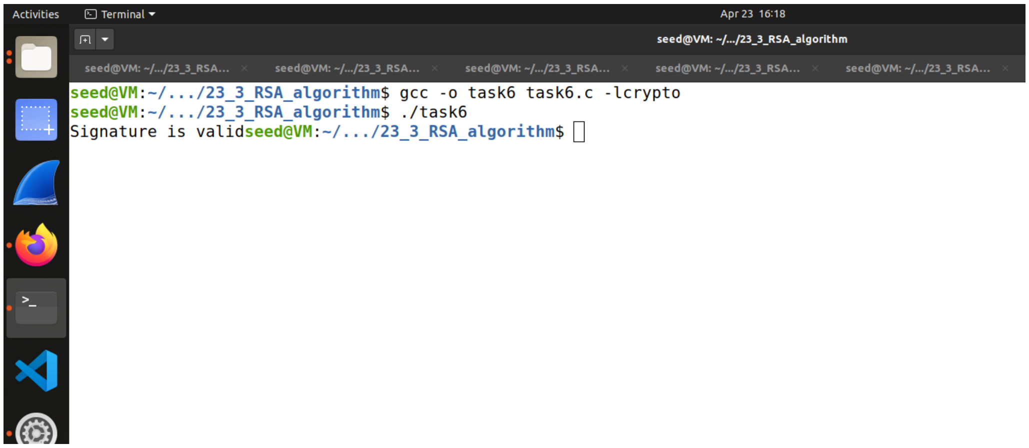


```
seed@VM:~/.../23_3_RSA_algorithm$ openssl asnlparse -i -in cl.pem
Error: offset out of range
seed@VM:~/.../23_3_RSA_algorithm$ 
    0:d=0  hl=4 l=4060 cons: SEQUENCE
    4:d=1  hl=4 l=3780 cons:  SEQUENCE
    8:d=2  hl=2 l=   3 cons:   cont [ 0 ]
   10:d=3  hl=2 l=   1 prim:    INTEGER           :02
   13:d=2  hl=2 l=  16 prim:   INTEGER           :018DBB3610E97273E973CE4261A51EF7
   31:d=2  hl=2 l=  13 cons:   SEQUENCE
   33:d=3  hl=2 l=   9 prim:    OBJECT            :sha256WithRSAEncryption
   44:d=3  hl=2 l=   0 prim:    NULL
   46:d=2  hl=2 l= 117 cons:   SEQUENCE
   48:d=3  hl=2 l=  11 cons:    SET
   50:d=4  hl=2 l=   9 cons:     SEQUENCE
   52:d=5  hl=2 l=   3 prim:      OBJECT            :countryName
   57:d=5  hl=2 l=   2 prim:      PRINTABLESTRING   :US
   61:d=3  hl=2 l=  21 cons:    SET
   63:d=4  hl=2 l=  19 cons:     SEQUENCE
   65:d=5  hl=2 l=   3 prim:      OBJECT            :organizationName
   70:d=5  hl=2 l=  12 prim:      PRINTABLESTRING   :DigiCert Inc
   84:d=3  hl=2 l=  25 cons:    SET
   86:d=4  hl=2 l=  23 cons:     SEQUENCE
   88:d=5  hl=2 l=   3 prim:      OBJECT            :organizationalUnitName
   93:d=5  hl=2 l=  16 prim:      PRINTABLESTRING   :www.digicert.com
  111:d=3  hl=2 l=  52 cons:    SET
  113:d=4  hl=2 l=  50 cons:     SEQUENCE
  115:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
  120:d=5  hl=2 l=  43 prim:      PRINTABLESTRING   :DigiCert SHA2 Extended Validation Server CA
  165:d=2  hl=2 l=  30 cons:   SEQUENCE
  167:d=3  hl=2 l=  13 prim:    UTCTIME           :220412000000Z
```

Verify the signature:



```c
#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 *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *M = BN_new();
    BIGNUM *C = BN_new();
    BIGNUM *S = BN_new();
    //BIGNUM *d = BN_new();

    //.bin
    BN_hex2bn(&M, "7061df0a50b8f2ba3367ecfabab273a16f3bb1378dbe1fe524e6dfd98dfa3b91");
    //module
    BN_hex2bn(&n, "C14BB3654770BCDD4F58DBEC9CEDC366E51F311354AD4A66461F2C0AEC6407E52EDCDCB90A20EDDFE3C4D09E9AA97A1D8288E51156DB1E9F58C251E72C340D2ED292E156CBF1795FB3BB87CA25037B9A52416610604F571349F0E8376783DFE7D34B674C2251A6DF0
    //algorithmsignature
    BN_hex2bn(&S, "8032ce5e0bdd6e5a0d0aafe1d684cbc08efa8570edda5db30cf72b7540fe850afaf33178b7704b1a8958ba88bdf36b1de97ecf0bba589c59d490d3fd6cfdd0986db771825bcf6d0b5a09d07bdec443d82aa4de9e41265fbb8f99cbddaee1a86f9f87fe74b71f1b20a

    BN_hex2bn(&e, "010001");
    BN_mod_exp(C, S, e, n, ctx);

    if (BN_cmp(C, M)==0)
        printf("Signature is valid");
    else
        printf("Signature is valid");

    return 0;
}
```



```
seed@VM:~/.../23_3_RSA_algorithm$ gcc -o task6 task6.c -lcrypto
seed@VM:~/.../23_3_RSA_algorithm$ ./task6
Signature is validseed@VM:~/.../23_3_RSA_algorithm$ 
```