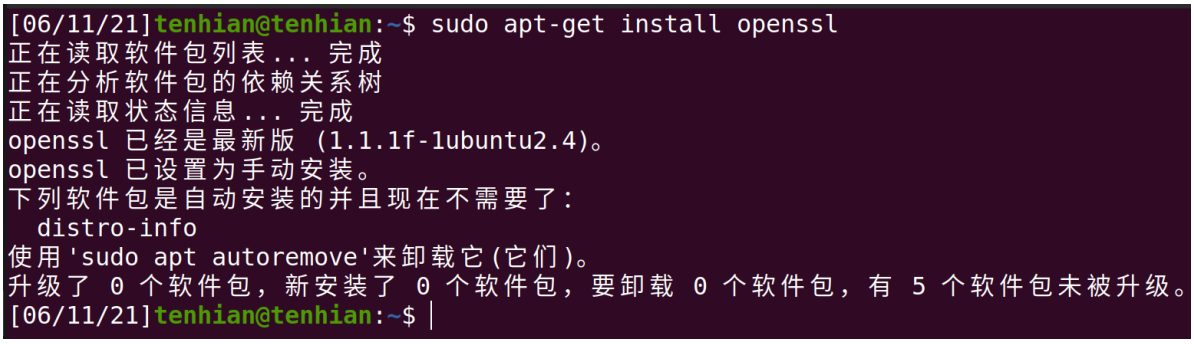
实验2流程

produced by TenHian

首先检查你的Ubuntu上是否装有openssl库的环境

我们在终端输入`openssl`

ctrl+c退出，则为已装好openssl库，若未如图所示，可执行`sudo apt-get install openssl`命令安装，如下图所示



Task 1: Deriving the Private Key

Let p, q, and e be three prime numbers. Let n = p*q. We will use (e, n) as the public key. Please calculate the private key d. The hexadecimal values of p, q, and e are listed in the following. It should be noted that although p and q used in this task are

quite large numbers, they are not large enough to be secure. We intentionally make them small for the sake of simplicity. In practice, these numbers should be at least 512 bits long (the one used here are only 128 bits).

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

p,q,e是三个素数，n=p*q，用（e,n）作为公钥,请计算私钥 d，p,e,q的十六进制值如下所示。这里只是示例选取的p,q不够大，实际密钥应至少512位长，这里只有128位

我们写出如下代码

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

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 *p1=BN_new();
    BIGNUM *q=BN_new();
    BIGNUM *q1=BN_new();
    BIGNUM *e=BN_new();
    BIGNUM *x=BN_new();
    BIGNUM *d=BN_new();
    BIGNUM *one=BN_new();

    //赋值p q e
    BN_hex2bn(&p,"F7E75FDC469067FFDC4E847C51F452DF");
    BN_hex2bn(&q,"E85CED54AF57E53E092113E62F436F4F");
    BN_hex2bn(&e,"0D88C3");

    //one这个变量是1
    BN_hex2bn(&one,"1");

    //p1=p-1
    BN_sub(p1,p,one);
```

```
    //q1=q-1
    BN_sub(q1,q,one);

    //x=p1*q1
    BN_mul(x,p1,q1,ctx);

    //e*d mod x = 1
    BN_mod_inverse(d,e,x,ctx);

    printBN("私钥d:",d);

    return 0;
}
```

编译运行

```
[06/11/21]tenhian@tenhian:~/.../RSA$ gcc task1.c -lcrypto
[06/11/21]tenhian@tenhian:~/.../RSA$ ./a.out
私钥d: 3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
[06/11/21]tenhian@tenhian:~/.../RSA$
```

Task 2: Encrypting a Message

Let (e, n) be the public key. Please encrypt the message "`A top secret!`" (the quotations are not included). We need to convert this ASCII string to a hex string, and then convert the hex string to a BIGNUM using the hex-to-bn API `BN hex2bn()`. The following `python` command can be used to convert a plain ASCII string to a hex string.

```
$ python -c 'print("A top secret!".encode("hex"))'
4120746f702073656372657421
```

The public keys are listed in the followings (hexadecimal). We also provide the private key d to help you verify your encryption result.

```
n = DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5
e = 010001 (this hex value equals to decimal 65537)
M = A top secret!
d = 74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D
```

用 (e,n) 作为公钥，加密消息`A top secret!`用下面的Python命令能将这条消息转换成16进制代码

已知该消息为 4120746f702073656372657421

代码

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

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 *M=BN_new();//明文
    BIGNUM *e=BN_new();
    BIGNUM *n=BN_new();
    BIGNUM *Eres=BN_new();//密文
    BIGNUM *Dres=BN_new();//解密之后明文
    BIGNUM *p=BN_new();
    BIGNUM *q=BN_new();
    BIGNUM *d=BN_new();

    //赋值m e p q
    BN_hex2bn(&M,"4120746f702073656372657421");
    BN_hex2bn(&e,"0D88C3");
    BN_hex2bn(&p,"F7E75FDC469067FFDC4E847C51F452DF");
    BN_hex2bn(&q,"E85CED54AF57E53E092113E62F436F4F");

    //n=p*q
    BN_mul(n,p,q,ctx);
    printBN("n:\n",n);

    //d我们在task1已求出

BN_hex2bn(&d,"3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB")
;

    //加密 M^e mod n
    BN_mod_exp(Eres,M,e,n,ctx);
    printBN("密文:",Eres);

    //解密 Eres^d mod n
```

```
    BN_mod_exp(Dres,Eres,d,n,ctx);
    printBN("明文:",Dres);

    return 0;
}
```

编译运行

```
[06/11/21]tenhian@tenhian:~/.../RSA$ gcc task2.c -lcrypto
[06/11/21]tenhian@tenhian:~/.../RSA$ ./a.out
n:
 E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1
密文： 90A81343DFE08415EDF79337CDE00457BAB56AFFA1B0CE5647BF9025665B396A
明文： 4120746F702073656372657421
[06/11/21]tenhian@tenhian:~/.../RSA$
```

pdf中最后给出了n和d的参考值，但在上程序中我们用的是自己算出来的，只要明文值相同就算成功

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 = 8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F
```

You can use the following python command to convert a hex string back to to a plain ASCII string.

```
$ python -c 'print("4120746f702073656372657421".decode("hex"))''
A top secret!
```

破译密文c，公私钥使用task2给出的

已知密文
c: 8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F

代码:

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

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 *Dres=BN_new();//解密之后明文
    BIGNUM *c=BN_new();
    BIGNUM *d=BN_new();
    BIGNUM *p=BN_new();
    BIGNUM *q=BN_new();


BN_hex2bn(&n,"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5")
;


BN_hex2bn(&c,"8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F")
;
    //d使用task2给的

BN_hex2bn(&d,"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D")
;

    //破译 c^d mod n
    BN_mod_exp(Dres,c,d,n,ctx);
    printBN("明文:",Dres);

    return 0;
}
```

编译运行

```
[06/11/21]tenhian@tenhian:~/.../RSA$ gcc task3.c -lcrypto
[06/11/21]tenhian@tenhian:~/.../RSA$ ./a.out
明文: 50617373776F72642069732064656573
[06/11/21]tenhian@tenhian:~/.../RSA$
```

明文: 50617373776F72642069732064656573

用Python命令转换为ascii

```
[06/11/21]tenhian@tenhian:~/.../RSA$ python -c 'print("50617373776F72642069732064656573".decode("hex"))'
Password is dees
[06/11/21]tenhian@tenhian:~/.../RSA$
```

Task 4: Signing a Message

The public/private keys used in this task are the same as the ones used in Task 2. Please generate a signature for the following message (please directly sign this message, instead of signing its hash value):

```
M = I owe you $2000.
```

Please make a slight change to the message M, such as changing $2000 to $3000, and sign the modified
message. Compare both signatures and describe what you observe.

直接签名消息M，同样的方式签名`M = I owe you $3000.`比较不同，公私钥使用task2给出的

先将该消息转为16进制

```
[06/11/21]tenhian@tenhian:~/.../RSA$ python -c 'print("I owe you $2000.".encode("hex"))'
49206f776520796f752024323030302e
[06/11/21]tenhian@tenhian:~/.../RSA$
```

```
M = 49206f776520796f752024323030302e
```

代码:

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

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 *M=BN_new();//明文
    BIGNUM *n=BN_new();
    BIGNUM *res=BN_new();//签名了的消息
    BIGNUM *d=BN_new();

    BN_hex2bn(&M,"49206f776520796f752024323030302e");

BN_hex2bn(&n,"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5")
;

BN_hex2bn(&d,"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D")
;

    //签名
    BN_mod_exp(res,M,d,n,ctx);

    printBN("签名后:",res);

    return 0;
}
```

编译运行

```
[06/11/21]tenhian@tenhian:~/.../RSA$ gcc task4.c -lcrypto
[06/11/21]tenhian@tenhian:~/.../RSA$ ./a.out
签名后: 55A4E7F17F04CCFE2766E1EB32ADDBA890BBE92A6FBE2D785ED6E73CCB35E4CB
[06/11/21]tenhian@tenhian:~/.../RSA$
```

```
签名后: 55A4E7F17F04CCFE2766E1EB32ADDBA890BBE92A6FBE2D785ED6E73CCB35E4CB
```

改成$3000

```
[06/11/21]tenhian@tenhian:~/.../RSA$ python -c 'print("I owe you $3000.".encode("hex"))'
49206f776520796f752024333030302e
[06/11/21]tenhian@tenhian:~/.../RSA$
```

```
M = 49206f776520796f752024333030302e
```

修改代码

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

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 *M=BN_new();//明文
    BIGNUM *n=BN_new();
```

```
    BIGNUM *res=BN_new();//签名了的消息
    BIGNUM *d=BN_new();

    BN_hex2bn(&M,"49206f776520796f752024333030302e");

BN_hex2bn(&n,"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5")
;

BN_hex2bn(&d,"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D")
;

    //签名
    BN_mod_exp(res,M,d,n,ctx);

    printBN("签名后:",res);

    return 0;
}
```

编译运行

```
[06/11/21]tenhian@tenhian:~/.../RSA$ gcc task4.c -lcrypto
[06/11/21]tenhian@tenhian:~/.../RSA$ ./a.out
签名后: BCC20FB7568E5D48E434C387C06A6025E90D29D848AF9C3EBAC0135D99305822
[06/11/21]tenhian@tenhian:~/.../RSA$
```

```
签名后: BCC20FB7568E5D48E434C387C06A6025E90D29D848AF9C3EBAC0135D99305822
```

可以看出有很大差别

Task 5: Verifying a Signature

Bob receives a message M = "Launch a missile." from Alice, with her signature S. We know that Alice's public key is (e, n). Please verify whether the signature is indeed Alice's or not. The public key and signature (hexadecimal) are listed in the following:

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

Suppose that the signature above is corrupted, such that the last byte of the signature changes from 2F
to 3F, i.e, there is only one bit of change. Please repeat this task, and describe what will happen to the verification process.

B收到了A发来的消息 Launch a missile. 这个消息以S 为签名，以(e,n)为公钥，请确认是否是A本人签发

假设上文的签名已损坏，将最后一位由2F改为3F，重复验证过程

先获取M的16进制形式

```
[06/11/21]tenhian@tenhian:~/.../RSA$ python -c 'print("Launch a missile.".encode("hex"))'
4c61756e63682061206d697373696c652e
[06/11/21]tenhian@tenhian:~/.../RSA$
```

```
M = 4c61756e63682061206d697373696c652e
```

代码:

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

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();//签名
    BIGNUM *n=BN_new();
    BIGNUM *e=BN_new();
    BIGNUM *M=BN_new();//明文16进制
    BIGNUM *Mi=BN_new();//验证

    BN_hex2bn(&M,"4c61756e63682061206d697373696c652e");

BN_hex2bn(&n,"AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115")
;
```

```
BN_hex2bn(&S,"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F")
;
    BN_hex2bn(&e,"010001");

    printBN("M原文:",M);

    //验证    S^e mod n
    BN_mod_exp(Mi,S,e,n,ctx);
    printBN("Mi验证:",Mi);

    if(BN_cmp(Mi,M)==0)
    {
        printf("Alice签名\n");
    }
    else
    {
        printf("非Alice签名\n");
    }

    return 0;
}
```

编译运行

```
[06/11/21]tenhian@tenhian:~/.../RSA$ gcc task5.c -lcrypto
[06/11/21]tenhian@tenhian:~/.../RSA$ ./a.out
M原文: 4C61756E63682061206D697373696C652E
Mi验证: 4C61756E63682061206D697373696C652E
Alice签名
[06/11/21]tenhian@tenhian:~/.../RSA$
```

修改S 将最后一位由2F改为3F

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

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();//签名
    BIGNUM *n=BN_new();
    BIGNUM *e=BN_new();
    BIGNUM *M=BN_new();//明文16进制
    BIGNUM *Mi=BN_new();//验证

    BN_hex2bn(&M,"4c61756e63682061206d697373696c652e");
```

```
BN_hex2bn(&n,"AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115")
;

BN_hex2bn(&S,"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F")
;
    BN_hex2bn(&e,"010001");

    printBN("M原文:",M);

    //验证    S^e mod n
    BN_mod_exp(Mi,S,e,n,ctx);
    printBN("Mi验证:",Mi);

    if(BN_cmp(Mi,M)==0)
    {
        printf("Alice签名\n");
    }
    else
    {
        printf("非Alice签名\n");
    }

    return 0;
}
```

编译运行

```
[06/11/21]tenhian@tenhian:~/.../RSA$ gcc task5.c -lcrypto
[06/11/21]tenhian@tenhian:~/.../RSA$ ./a.out
M原文： 4C61756E63682061206D697373696C652E
Mi验证： 91471927C80DF1E42C154FB4638CE8BC726D3D66C83A4EB6B7BE0203B41AC294
非Alice签名
[06/11/21]tenhian@tenhian:~/.../RSA$
```

可见，修改了签名后的消息一位，就会与原文产生很大差异，足以判别是否为本人签发

Task 6: Manually Verifying an X.509 Certificate

In this task, we will manually verify an X.509 certificate using our program. An X.509 contains data about a public key and an issuer's signature on the data. We will download a real X.509 certificate from a

web server, get its issuer's public key, and then use this public key to verify the signature on the certificate.

在此任务中，我们将用程序验证一个X.509证书。一个X.509证书包含公钥和发行者签名，我们将会从网上下载一个X.509证书，使用其公钥验证其签名

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

从www.example.org下载证书，让我们不要用前面的示例服务器

那就用 www.baidu.com 端口号:443

执行命令`openssl s_client -connect www.baidu.com:443 -showcerts`

```
[06/11/21]tenhian@tenhian:~/.../RSA$ openssl s_client -connect www.baidu.com:443 -showcerts
CONNECTED(00000003)
depth=2 C = BE, O = GlobalSign nv-sa, OU = Root CA, CN = GlobalSign Root CA
verify return:1
depth=1 C = BE, O = GlobalSign nv-sa, CN = GlobalSign Organization Validation CA - SHA256 - G2
verify return:1
depth=0 C = CN, ST = beijing, L = beijing, OU = service operation department, O = "Beijing Baidu
 Netcom Science Technology Co., Ltd", CN = baidu.com
verify return:1
---
Certificate chain
 0 s:C = CN, ST = beijing, L = beijing, OU = service operation department, O = "Beijing Baidu Ne
tcom Science Technology Co., Ltd", CN = baidu.com
   i:C = BE, O = GlobalSign nv-sa, CN = GlobalSign Organization Validation CA - SHA256 - G2
-----BEGIN CERTIFICATE-----
MIIKLjCCCRagAwIBAgIMclh4Nm6fVugdQYhIMA0GCSqGSIb3DQEBCwUAMGYxCzAJ
BgNVBAYTAkJFMRkwFwYDVQQKExBHbG9iYWxTaWduIG52LXNhMTwwOgYDVQQDEzNH
bG9iYWxTaWduIE9yZ2FuaXphdGlvbiBWYWxpZGF0aW9uIENBIC0gU0hBMjU2IC0g
RzIwHhcNMjAwNDAyMDcwNDU4WhcNMjEwNzI2MDUzMTAyWjCBpzELMAkGA1UEBhMC
Q04xEDAOBgNVBAgTB2JlaWppbmcxEDAOBgNVBAcTB2JlaWppbmcxJTAjBgNVBAsT
HHNlcnZpY2Ugb3BlcmF0aW9uIGRlcGFydG1lbnQxOTA3BgNVBAoTMEJlaWppbmcg
QmFpZHUgTmV0Y29tIFNjaWVuY2UgVGVjaG5vbG9neSBDby4sIEx0ZDESMBAGA1UE
```

分别将两段BEGIN到END复制到c0.pem和c1.pem

得到c0.pem和c1.pem

c0.pem | c1.pem | task5.c | task4.c | task3.c

```
-----BEGIN CERTIFICATE-----
MIIKLjCCCRagAwIBAgIMclh4Nm6fVugdQYhIMA0GCSqGSIb3DQEBCwUAMGYxCzAJ
BgNVBAYTAkJFMRkwFwYDVQQKExBHbG9iYWxTaWduIG52LXNhMTwwOgYDVQQDEzNH
bG9iYWxTaWduIE9yZ2FuaXphdGlvbiBWYWxpZGF0aW9uIENBIC0gU0hBMjU2IC0g
RzIwHhcNMjAwNDAyMDcwNDU4WhcNMjEwNzI2MDUzMTAyWjCBpzELMAkGA1UEBhMC
Q04xEDAOBgNVBAgTB2JlaWppbmcxEDAOBgNVBAcTB2JlaWppbmcxJTAjBgNVBAsT
HHNlcnZpY2Ugb3BlcmF0aW9uIGRlcGFydG1lbnQxOTA3BgNVBAoTMEJlaWppbmcg
QmFpZHUgTmV0Y29tIFNjaWVuY2UgVGVjaG5vbG9neSBDby4sIEx0ZDESMBAGA1UE
AxMJYmFpZHUuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwamw
rkca0lfrHRUfblyy5PgLINvqAN8p/6RriSZLnyMv7FewirhGQCp+vNxaRZdPrUEO
vCCGSwxdVSFH4jE8V6fsmUfrRw1y18gWVHXv00URD0vOYHpGXCh0ro4bvthwZnuo
k0ko0qN2lFXefCfyD/eYDK2G2sau/Z/w2YEympfjIe4EkpbkeBHlxBAOEDF6Speg
68ebxNqJN6nDN9dWsX9Sx9kmCtavOBaxbftzebFoeQOQ64h7jEiRmFGlB5SGpXhG
eY9Ym+k1Wafxe1cxCpDPJM4NJOeSsmrp5pY3Crh8hy900lzoSwpfZhinQYbPJqYI
jqVJF5JTs5Glz1OwMQIDAQABo4IGmDCCBpQwDgYDVR0PAQH/BAQDAgWgMIGgBggr
BgEFBQcBAQSBkzCBkDBNBggrBgEFBQcwAoZBaHR0cDovL3NlY3VyZS5nbG9iYWxz
aWduLmNvbS9jYWNlcnQvZ3Nvcmdhbml6YXRpb252YWxzaGEyZzJyMS5jcnQwPwYI
KwYBBQUHMAGGM2h0dHA6Ly9vY3NwMi5nbG9iYWxzaWduLmNvbS9nc29yZ2FuaXph
dGlvbnZhbHNoYTJnMjBWBgNVHSAETzBNMEEGCSsGAQQBoDIBFDA0MDIGCCsGAQUF
BwIBFiZodHRwczovL3d3dy5nbG9iYWxzaWduLmNvbS9yZXBvc2l0b3J5LzAIBgZn
gQwBAgIwCQYDVR0TBAIwADBJBgNVHR8EQjBAMD6gPKA6hjhodHRwOi8vY3JsLmds
b2JhbHNpZ24uY29tL2dzL2dzb3JnYW5pemF0aW9udmFsc2hhMmcyLmNybDCCA04G
A1UdEQSCA0UwggNBggliYWlkdS5jb22CDGJhaWZ1YmFvLmNvbYIMd3d3LmJhaWR1
LmNughB3d3cuYmFpZHUuY29tLmNugg9tY3QueS5udW9taS5jb22CC2Fwb2xsby5h
dXRvggZkd3ouY26CCyouYmFpZHUuY29tgg4qLmJhaWZ1YmFvLmNvbYIRKi5iYWlk
dXN0YXRpYy5jb22CDiouYmRzdGF0aWMuY29tggsqLmJkaW1nLmNvbYIMKi5oYW8x
MjMuY29tggsqLm51b21pLmNvbYINKi5jaHVhbmtlLmNvbYINKi50cnVzdGdvLmNv
bYIPKi5iY2UuYmFpZHUuY29tghAqLmV5dW4uYmFpZHUuY29tgg8qLm1hcC5iYWlk
dS5jb22CDyoubWJkLmJhaWR1LmNvbYIRKi5mYW55aS5iYWlkdS5jb22CDiouYmFp
ZHViY2UuY29tggwqLm1pcGNkbi5jb22CECoubmV3cy5iYWlkdS5jb22CDiouYmFp
```

c0.pem | c1.pem | task5.c | task4.c | task3.c

```
-----BEGIN CERTIFICATE-----
MIIEaTCCA1GgAwIBAgILBAAAAAABRE7wQkcwDQYJKoZIhvcNAQELBQAwVzELMAkG
A1UEBhMCQkUxGTAXBgNVBAoTEEdsb2JhbFNpZ24gbnYtc2ExEDAOBgNVBAsTB1Jv
b3QgQ0ExGzAZBgNVBAMTEkdsb2JhbFNpZ24gUm9vdCBDQTAeFw0xNDAyMjAxMDAw
MDBaFw0yNDAyMjAxMDAwMDBaMGYxCzAJBgNVBAYTAkJFMRkwFwYDVQQKExBHbG9i
YWxTaWduIG52LXNhMTwwOgYDVQQDEzNHbG9iYWxTaWduIE9yZ2FuaXphdGlvbiBW
YWxpZGF0aW9uIENBIC0gU0hBMjU2IC0gRzIwggEiMA0GCSqGSIb3DQEBAQUAA4IB
DwAwggEKAoIBAQDHDmw/I5N/zHClnSDDDlM/fsBOwphJykfVI+8DNIV0yKMCLkZc
C33JiJ1Pi/D4nGyMVTXbv/Kz6vvjVudKRtkTIso21ZvBqOOWQ5PyDLzm+ebomchj
SHh/VzZpGhkdWtHUfcKc1H/hgBKueuqI6lfYygoKOhJJomIZeg0k9zfrtHOSewUj
mxK1zusp36QUArkBpdSmnENkiN74fv7j9R7l/tyjqORmMdlMJekYuYlZCa7pnRxt
Nw9KHjUgKOKv1CGLAcRFrW4rY6uSa2EKTSDtc7p8zv4WtdufgPDWi2zZCHlKT3hl
2pK8vjX5s8T5J4BO/5ZS5gIg4Qdz6V0rvbLxAgMBAAGjggElMIIBITAOBgNVHQ8B
Af8EBAMCAQYwEgYDVR0TAQH/BAgwBgEB/wIBADAdBgNVHQ4EFgQUlt5h8b0cFilT
HMDMfTuDAEDmGnwwRwYDVR0gBEAwPjA8BgRVHSAAMDQwMgYIKwYBBQUHAgEWJmh0
dHBzOi8vd3d3Lmdsb2JhbHNpZ24uY29tL3JlcG9zaXRvcnkvMDMGA1UdHwQsMCow
KKAmoCSGImh0dHA6Ly9jcmwuZ2xvYmFsc2lnbi5uZXQvcm9vdC5jcmwwPQYIKwYB
BQUHAQEEMTAvMC0GCCsGAQUFBzABhiFodHRwOi8vb2NzcC5nbG9iYWxzaWduLmNv
bS9yb290cjEwHwYDVR0jBBgwFoAUYHtmGkUNl8qJUC99BM00qP/8/UswDQYJKoZI
hvcNAQELBQADggEBAEYq7l69rgFgNzERhnF0tkZJyBAW/i9iIxerH4f4gu3K3w4s
32R1juUYcqeMOovJrKV3UPfvnqTgoI8UV6MqX+x+bRDmuo2wCId2Dkyy2VG7EQLy
XN0cvfNVlg/UBsD84iOKJHDTu/B5GqdhcIOKrwbFINihY9Bsrk8y1658GEV1BSl3
30JAZGSGvip2CTFvHST0mdCF/vIhCPnG9vHQWe3WVjwIKANnuvD58ZAWR65n5ryA
SOlCdjSXVWkkDoPWoC209fN5ikkodBpBocLTJIg1MGCUF7ThBCIxPTsvFwayuJ2G
K1pp74P1S8SqtCr4fKGxhZSM9AyHDPSsQPhZSZg=
-----END CERTIFICATE-----
```

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

从发行者证书中提取公钥 (e,n)

获取n的值

执行命令`openssl x509 -in c1.pem -noout -modulus`

```
[06/11/21]tenhian@tenhian:~/.../RSA$ openssl x509 -in c1.pem -noout -modulus
Modulus=C70E6C3F23937FCC70A59D20C30E533F7EC04EC29849CA47D523EF03348574C8A3022E465C0B7DC9889D4F8B
F0F89C6C8C5535DBBFF2B3EAFBE356E74A46D91322CA36D59BC1A8E3964393F20CBCE6F9E6E899C86348787F5736691A
191D5AD1D47DC29CD47FE18012AE7AEA88EA57D8CA0A0A3A1249A262197A0D24F737EBB473927B05239B12B5CEEB29DF
A41402B901A5D4A69C436488DEF87EFEE3F51EE5FEDCA3A8E46631D94C25E918B9895909AEE99D1C6D370F4A1E352028
E2AFD4218B01C445AD6E2B63AB926B610A4D20ED73BA7CCEFE16B5DB9F80F0D68B6CD908794A4F7865DA92BCBE35F9B3
C4F927804EFF9652E60220E10773E95D2BBDB2F1
[06/11/21]tenhian@tenhian:~/.../RSA$
```

```
Modulus=
C70E6C3F23937FCC70A59D20C30E533F7EC04EC29849CA47D523EF03348574C8A3022E465C0B7DC9
889D4F8BF0F89C6C8C5535DBBFF2B3EAFBE356E74A46D91322CA36D59BC1A8E3964393F20CBCE6F9
E6E899C86348787F5736691A191D5AD1D47DC29CD47FE18012AE7AEA88EA57D8CA0A0A3A1249A262
197A0D24F737EBB473927B05239B12B5CEEB29DFA41402B901A5D4A69C436488DEF87EFEE3F51EE5
FEDCA3A8E46631D94C25E918B9895909AEE99D1C6D370F4A1E352028E2AFD4218B01C445AD6E2B63
AB926B610A4D20ED73BA7CCEFE16B5DB9F80F0D68B6CD908794A4F7865DA92BCBE35F9B3C4F92780
4EFF9652E60220E10773E95D2BBDB2F1
```

获取e的值

执行命令`openssl x509 -in c1.pem -text -noout`

```
tenhian@tenhian: ~/.../RSA
                f0:f8:9c:6c:8c:55:35:db:bf:f2:b3:ea:fb:e3:56:
                e7:4a:46:d9:13:22:ca:36:d5:9b:c1:a8:e3:96:43:
                93:f2:0c:bc:e6:f9:e6:e8:99:c8:63:48:78:7f:57:
                36:69:1a:19:1d:5a:d1:d4:7d:c2:9c:d4:7f:e1:80:
                12:ae:7a:ea:88:ea:57:d8:ca:0a:0a:3a:12:49:a2:
                62:19:7a:0d:24:f7:37:eb:b4:73:92:7b:05:23:9b:
                12:b5:ce:eb:29:df:a4:14:02:b9:01:a5:d4:a6:9c:
                43:64:88:de:f8:7e:fe:e3:f5:1e:e5:fe:dc:a3:a8:
                e4:66:31:d9:4c:25:e9:18:b9:89:59:09:ae:e9:9d:
                1c:6d:37:0f:4a:1e:35:20:28:e2:af:d4:21:8b:01:
                c4:45:ad:6e:2b:63:ab:92:6b:61:0a:4d:20:ed:73:
                ba:7c:ce:fe:16:b5:db:9f:80:f0:d6:8b:6c:d9:08:
                79:4a:4f:78:65:da:92:bc:be:35:f9:b3:c4:f9:27:
                80:4e:ff:96:52:e6:02:20:e1:07:73:e9:5d:2b:bd:
                b2:f1
            Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Subject Key Identifier:
```

```
Exponent: 65537 (0x10001)
```

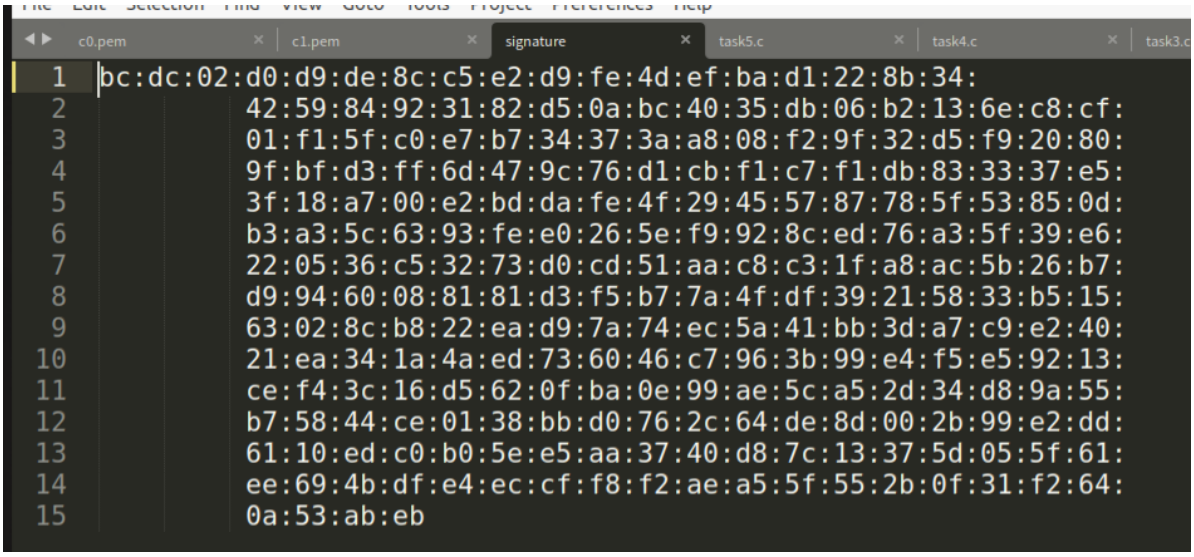
Step 3: Extract the signature from the server's certificate.

获取签名值

执行命令`openssl x509 -in c0.pem -text -noout`

```
    Signature Algorithm: sha256WithRSAEncryption
         bc:dc:02:d0:d9:de:8c:c5:e2:d9:fe:4d:ef:ba:d1:22:8b:34:
         42:59:84:92:31:82:d5:0a:bc:40:35:db:06:b2:13:6e:c8:cf:
         01:f1:5f:c0:e7:b7:34:37:3a:a8:08:f2:9f:32:d5:f9:20:80:
         9f:bf:d3:ff:6d:47:9c:76:d1:cb:f1:c7:f1:db:83:33:37:e5:
         3f:18:a7:00:e2:bd:da:fe:4f:29:45:57:87:78:5f:53:85:0d:
         b3:a3:5c:63:93:fe:e0:26:5e:f9:92:8c:ed:76:a3:5f:39:e6:
         22:05:36:c5:32:73:d0:cd:51:aa:c8:c3:1f:a8:ac:5b:26:b7:
         d9:94:60:08:81:81:d3:f5:b7:7a:4f:df:39:21:58:33:b5:15:
         63:02:8c:b8:22:ea:d9:7a:74:ec:5a:41:bb:3d:a7:c9:e2:40:
         21:ea:34:1a:4a:ed:73:60:46:c7:96:3b:99:e4:f5:e5:92:13:
         ce:f4:3c:16:d5:62:0f:ba:0e:99:ae:5c:a5:2d:34:d8:9a:55:
         b7:58:44:ce:01:38:bb:d0:76:2c:64:de:8d:00:2b:99:e2:dd:
         61:10:ed:c0:b0:5e:e5:aa:37:40:d8:7c:13:37:5d:05:5f:61:
         ee:69:4b:df:e4:ec:cf:f8:f2:ae:a5:5f:55:2b:0f:31:f2:64:
         0a:53:ab:eb
```

将这一段复制成文件`signature`



执行命令`cat signature | tr -d '[:space:]:'`

```
[06/11/21]tenhian@tenhian:~/.../RSA$ cat signature | tr -d '[:space:]:'
bcdc02d0d9de8cc5e2d9fe4defbad1228b34425984923182d50abc4035db06b2136ec8cf01f15fc0e7b734373aa808f2
9f32d5f920809fbfd3ff6d479c76d1cbf1c7f1db833337e53f18a700e2bddafe4f29455787785f53850db3a35c6393fe
e0265ef9928ced76a35f39e6220536c53273d0cd51aac8c31fa8ac5b26b7d99460088181d3f5b77a4fdf39215833b515
63028cb822ead97a74ec5a41bb3da7c9e24021ea341a4aed736046c7963b99e4f5e59213cef43c16d5620fba0e99ae5c
a52d34d89a55b75844ce0138bbd0762c64de8d002b99e2dd6110edc0b05ee5aa3740d87c13375d055f61ee694bdfe4ec
cff8f2aea55f552b0f31f2640a53abeb[06/11/21]tenhian@tenhian:~/.../RSA$
```

```
bcdc02d0d9de8cc5e2d9fe4defbad1228b34425984923182d50abc4035db06b2136ec8cf01f15fc0
e7b734373aa808f29f32d5f920809fbfd3ff6d479c76d1cbf1c7f1db833337e53f18a700e2bddafe
4f29455787785f53850db3a35c6393fee0265ef9928ced76a35f39e6220536c53273d0cd51aac8c3
1fa8ac5b26b7d99460088181d3f5b77a4fdf39215833b51563028cb822ead97a74ec5a41bb3da7c9
e24021ea341a4aed736046c7963b99e4f5e59213cef43c16d5620fba0e99ae5ca52d34d89a55b758
44ce0138bbd0762c64de8d002b99e2dd6110edc0b05ee5aa3740d87c13375d055f61ee694bdfe4ec
cff8f2aea55f552b0f31f2640a53abeb
```

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

获取证书主体

执行命令 `openssl asn1parse -i -in c0.pem -strparse 4 -out c0_body.bin -noout`

执行命令 `sha256sum c0_body.bin`

```
[06/11/21]tenhian@tenhian:~/.../RSA$ openssl asn1parse -i -in c0.pem -strparse 4 -out c0_body.bi
n -noout
[06/11/21]tenhian@tenhian:~/.../RSA$ sha256sum c0_body.bin
8afeec4c6ac9dfb5c5c946fbd43adfa5d7af9d4f8152e18135e2f1d0f0bd8083  c0_body.bin
[06/11/21]tenhian@tenhian:~/.../RSA$
```

```
8afeec4c6ac9dfb5c5c946fbd43adfa5d7af9d4f8152e18135e2f1d0f0bd8083
```

Step 5: Verify the signature.

验证签名

代码:

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

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();//签名
    BIGNUM *n=BN_new();
    BIGNUM *e=BN_new();
    BIGNUM *M=BN_new();//明文16进制
    BIGNUM *Mi=BN_new();//验证


BN_hex2bn(&M,"8afeec4c6ac9dfb5c5c946fbd43adfa5d7af9d4f8152e18135e2f1d0f0bd8083")
;

BN_hex2bn(&n,"C70E6C3F23937FCC70A59D20C30E533F7EC04EC29849CA47D523EF03348574C8A3
022E465C0B7DC9889D4F8BF0F89C6C8C5535DBBFF2B3EAFBE356E74A46D91322CA36D59BC1A8E396
4393F20CBCE6F9E6E899C86348787F5736691A191D5AD1D47DC29CD47FE18012AE7AEA88EA57D8CA
0A0A3A1249A262197A0D24F737EBB473927B05239B12B5CEEB29DFA41402B901A5D4A69C436488DE
F87EFEE3F51EE5FEDCA3A8E46631D94C25E918B9895909AEE99D1C6D370F4A1E352028E2AFD4218B
01C445AD6E2B63AB926B610A4D20ED73BA7CCEFE16B5DB9F80F0D68B6CD908794A4F7865DA92BCBE
35F9B3C4F927804EFF9652E60220E10773E95D2BBDB2F1");
```

```
BN_hex2bn(&S,"bcdc02d0d9de8cc5e2d9fe4defbad1228b34425984923182d50abc4035db06b213
6ec8cf01f15fc0e7b734373aa808f29f32d5f920809fbfd3ff6d479c76d1cbf1c7f1db833337e53f
18a700e2bddafe4f29455787785f53850db3a35c6393fee0265ef9928ced76a35f39e6220536c532
73d0cd51aac8c31fa8ac5b26b7d99460088181d3f5b77a4fdf39215833b51563028cb822ead97a74
ec5a41bb3da7c9e24021ea341a4aed736046c7963b99e4f5e59213cef43c16d5620fba0e99ae5ca5
2d34d89a55b75844ce0138bbd0762c64de8d002b99e2dd6110edc0b05ee5aa3740d87c13375d055f
61ee694bdfe4eccff8f2aea55f552b0f31f2640a53abeb");
    BN_hex2bn(&e,"010001");

    printBN("M原文:",M);

    //验证    S^e mod n
    BN_mod_exp(Mi,S,e,n,ctx);
    printBN("Mi验证:",Mi);

    return 0;
}
```

编译运行

```
[06/11/21]tenhian@tenhian:~/.../RSA$ gcc task6.c -lcrypto
[06/11/21]tenhian@tenhian:~/.../RSA$ ./a.out
M原文： 8AFEEC4C6AC9DFB5C5C946FBD43ADFA5D7AF9D4F8152E18135E2F1D0F0BD8083
Mi验证： 01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF003031300D0609608648016503040201050004208AFEEC4C6AC9DFB5C5C946FBD4
3ADFA5D7AF9D4F8152E18135E2F1D0F0BD8083
[06/11/21]tenhian@tenhian:~/.../RSA$
```

文段末尾与原文相同，签名有效