Task 1

(attempted to compile, could not resolve)

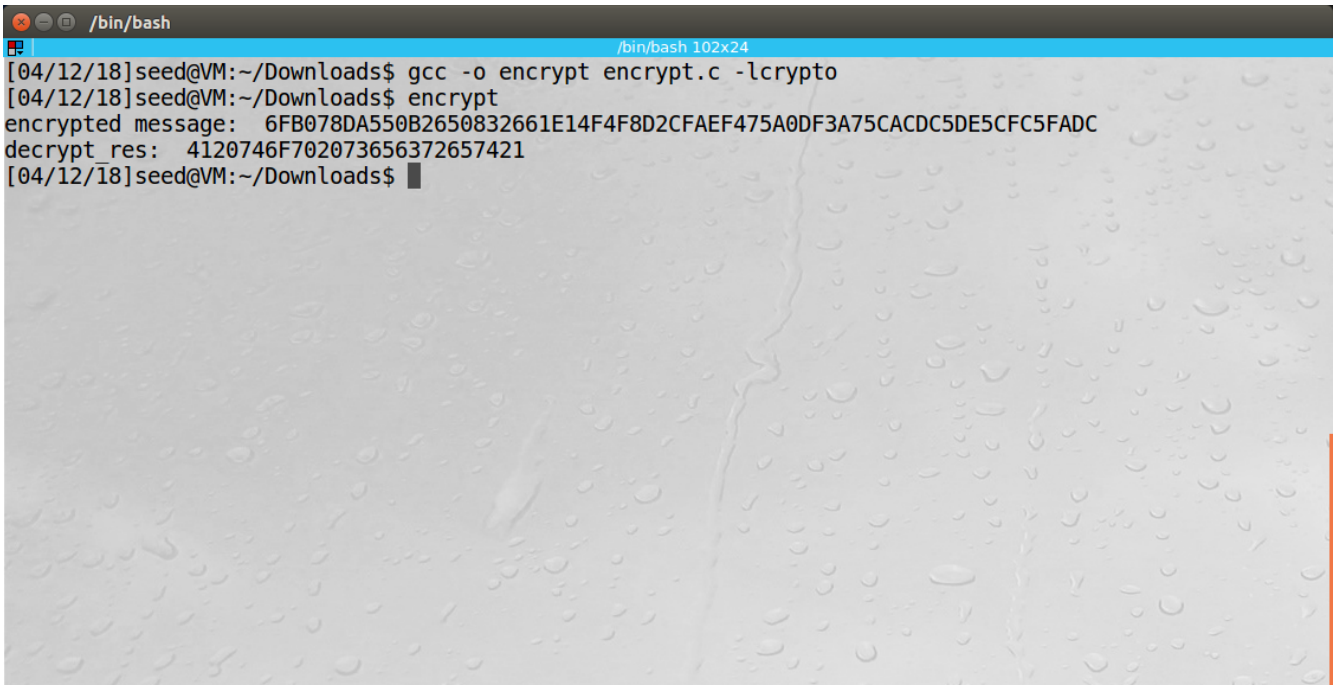
```
/bin/bash
/bin/bash 102x24
[04/12/18]seed@VM:~/Downloads$ gcc -o rsa rsa.c -lcrypto
rsa.c: In function ‘main’:
rsa.c:39:8: warning: assignment makes pointer from integer without a cast [-Wint-conversion]
  sub_p = BN_sub(p, p, one);
        ^
rsa.c:40:8: warning: assignment makes pointer from integer without a cast [-Wint-conversion]
  sub_q = BN_sub(q, q, one);
        ^
rsa.c:42:6: warning: assignment makes pointer from integer without a cast [-Wint-conversion]
  phi = BN_mul(phi, sub_p, sub_q, ctx);
      ^
[04/12/18]seed@VM:~/Downloads$ rsa
p * q =  E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1
Segmentation fault (core dumped)
[04/12/18]seed@VM:~/Downloads$
```

Code:

```
/* Calculate the private key. */

#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 *p = BN_new();
        BIGNUM *q = BN_new();
        BIGNUM *e = BN_new();
```

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

    // calculate n first
    BIGNUM *n = BN_new();
    BN_mul(n, p, q, ctx);
    printBN("p * q = ", n);

    // calculate phi which is p-1 * q-1
    BIGNUM *phi = BN_new();
    BIGNUM *one = BN_new();
    BIGNUM *sub_p = BN_new();
    BIGNUM *sub_q = BN_new();

    BN_hex2bn(&phi, "0");
    BN_hex2bn(&one, "1");

    sub_p = BN_sub(p, p, one);
    sub_q = BN_sub(q, q, one);

    phi = BN_mul(phi, sub_p, sub_q, ctx);
    printBN("phi = ", phi);

    // have e, so do mod inverse
    BIGNUM *d = BN_new();
    d = BN_mod_inverse(d, e, n, ctx);
    printBN("d = ", d);


}
```

Task 2

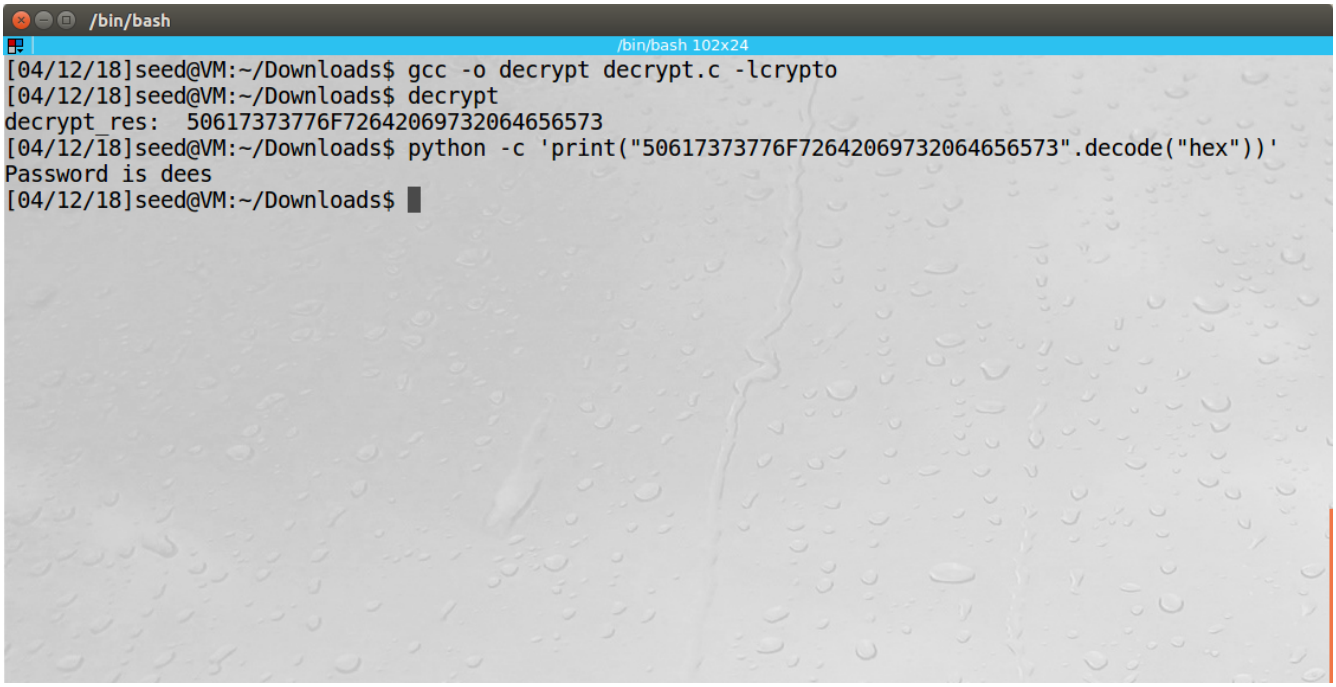


Code:

```
#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 *d = BN_new();
	BIGNUM *encrypt_res = BN_new();
	BIGNUM *decrypt_res = BN_new();
```

```
	BN_hex2bn(&n,
"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
	BN_hex2bn(&e, "010001");
	BN_hex2bn(&M, "4120746f702073656372657421");
	BN_hex2bn(&d,
"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

	// encrypt M^e mod n
	BN_mod_exp(encrypt_res, M, e, n, ctx);
	printBN("encrypted message: ", encrypt_res);

	// decrypt y^d mod n
	BN_mod_exp(decrypt_res, encrypt_res, d, n, ctx);
	printBN("decrypt_res: ", decrypt_res);

	return 0;
}
```

Task 3



Code:

```
#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 *C = BN_new();
	BIGNUM *d = BN_new();
	BIGNUM *decrypt_res = BN_new();



	BN_hex2bn(&n,
"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
	BN_hex2bn(&e, "010001");
	BN_hex2bn(&C,
"8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F");
	BN_hex2bn(&d,
"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

	// decrypt y^d mod n
	BN_mod_exp(decrypt_res, C, d, n, ctx);
	printBN("decrypt_res: ", decrypt_res);

	return 0;
}
```

Task 4

The signatures are very different despite being nearly the same.

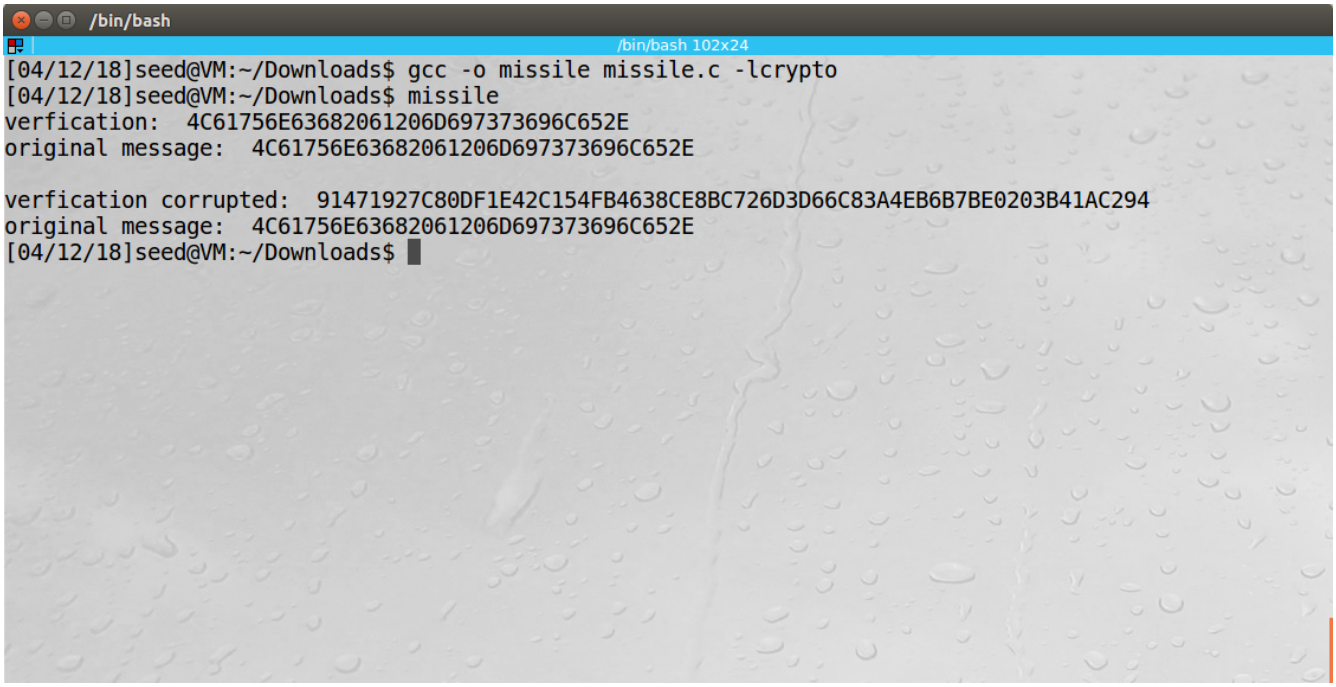
Code:

```
#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(); // python -c 'print("I owe you $2000".encode("hex"))'
        BIGNUM *Mprime = BN_new(); // python -c 'print("I owe you $3000".encode("hex"))'
        BIGNUM *d = BN_new();
        BIGNUM *sig = BN_new();
```

```
	BN_hex2bn(&n,
"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
	BN_hex2bn(&e, "010001");
	BN_hex2bn(&M, "49206f776520796f75202432303030");
	BN_hex2bn(&Mprime, "49206f776520796f75202433303030");
	BN_hex2bn(&d,
"74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

	// compute sig M^d mod n
	BN_mod_exp(sig, M, d, n, ctx);
	printBN("signature: ", sig);

	// compute sig Mprime^d mod n
	BN_mod_exp(sig, Mprime, d, n, ctx);
	printBN("other signature: ", sig);


	return 0;
}
```

Task 5

The verification code will change significantly from the original message.



Code:

```
#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(); // python -c 'print("Launch a missile.".encode("hex"))'
	BIGNUM *S = BN_new();
	BIGNUM *S_corrupted = BN_new();
	BIGNUM *verify = BN_new();




	BN_hex2bn(&n,
"AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");
	BN_hex2bn(&e, "010001");
	BN_hex2bn(&M, "4c61756e63682061206d697373696c652e");
	BN_hex2bn(&S,
"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F");
	BN_hex2bn(&S_corrupted,
"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F");

	// verify S^e mod n
	BN_mod_exp(verify, S, e, n, ctx);
	printBN("verfication: ", verify);
	printBN("original message: ", M);

	printf("\n");

	// verify S_corrupted^e mod n
	BN_mod_exp(verify, S_corrupted, e, n, ctx);
	printBN("verfication corrupted: ", verify);
	printBN("original message: ", M);
```

```
        return 0;
}
```

Task 6

Step 1:

```
/bin/bash
/bin/bash 102x24
[04/12/18]seed@VM:~/Downloads$ clear
[3;J
[04/12/18]seed@VM:~/Downloads$ openssl s_client -connect www.google.com:443 -showcerts
CONNECTED(00000003)
depth=2 C = US, O = GeoTrust Inc., CN = GeoTrust Global CA
verify return:1
depth=1 C = US, O = Google Inc, CN = Google Internet Authority G2
verify return:1
depth=0 C = US, ST = California, L = Mountain View, O = Google Inc, CN = www.google.com
verify return:1
---
Certificate chain
 0 s:/C=US/ST=California/L=Mountain View/O=Google Inc/CN=www.google.com
   i:/C=US/O=Google Inc/CN=Google Internet Authority G2
-----BEGIN CERTIFICATE-----
MIIEdjCCA16gAwIBAgIIX+fxYwVg068wDQYJKoZIhvcNAQELBQAwSTELMAkGA1UE
BhMCVVMxEzARBgNVBAoTCkdvb2dsZSBJbmMxJTAjBgNVBAMTHEdvb2dsZSBJbnRl
cm5ldCBBdXRob3JpdHkgRzIwHhcNMTgwMzI4MTM0NzMwWhcNMTgwNjIwMTMyMzAw
WjBoMQswCQYDVQQGEwJVUzETMBEGA1UECAwKQ2FsaWZvcm5pYTEWMBQGA1UEBwwN
TW91bnRhaW4gVmlldzETMBEGA1UECgwKR29vZ2xlIEluYzEXMBUGA1UEAwwOd3d3
Lmdvb2dsZS5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDMNkI2
/8x7MN9EJ8LK+iwGuXp7EITLBJUOM1xY6miH8VH3DB+6RURsiRL4rD9vFwCfa/FR
kzD2qdU/oiUXtOCPcdQ86W3ePqjcVFcbmp8jsnaIPF8SWSNs6XtMKE1CdhYsY/fb
1qW5W1xOPXrWaTG2yxbNRc8iO8797QQV3djxwrw28KPjZh5qCEQ+LXJCJWD0bRnW
```

Step 2:

Modulus=9C2A04775CD850913A06A382E0D85048BC893FF119701A88467EE08FC5F189CE21EE
5AFE610DB7324489A0740B534F55A4CE826295EEEB595FC6E1058012C45E943FBC5B4838F453
F724E6FB91E915C4CFF4530DF44AFC9F54DE7DBEA06B6F87C0D0501F28300340DA0873516C
7FFF3A3CA737068EBD4B1104EB7D24DEE6F9FC3171FB94D560F32E4AAF42D2CBEAC46A1A
B2CC53DD154B8B1FC819611FCD9DA83E632B8435696584C819C54622F85395BEE3804A10C62
AECBA972011C739991004A0F0617A95258C4E5275E2B6ED08CA14FCCE226AB34ECF46039797
037EC0B1DE7BAF4533CFBA3E71B7DEF42525C20D35899D9DFB0E1179891E37C5AF8E7269

Exponent: 65537 (0x10001)

```
/bin/bash
/bin/bash 102x24
3;J
[04/12/18]seed@VM:~/Downloads$ openssl x509 -in c1.pem -noout -modulus
Modulus=9C2A04775CD850913A06A382E0D85048BC893FF119701A88467EE08FC5F189CE21EE5AFE610DB7324489A0740B534F
55A4CE826295EEEB595FC6E1058012C45E943FBC5B4838F453F724E6FB91E915C4CFF4530DF44AFC9F54DE7DBEA06B6F87C0D0
501F28300340DA0873516C7FFF3A3CA737068EBD4B1104EB7D24DEE6F9FC3171FB94D560F32E4AAF42D2CBEAC46A1AB2CC53DD
154B8B1FC819611FCD9DA83E632B8435696584C819C54622F85395BEE3804A10C62AECBA972011C739991004A0F0617A95258C
4E5275E2B6ED08CA14FCCE226AB34ECF46039797037EC0B1DE7BAF4533CFBA3E71B7DEF42525C20D35899D9DFB0E1179891E37
C5AF8E7269
[04/12/18]seed@VM:~/Downloads$ openssl x509 -in c1.pem -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            01:00:21:25:88:b0:fa:59:a7:77:ef:05:7b:66:27:df
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=GeoTrust Inc., CN=GeoTrust Global CA
        Validity
            Not Before: May 22 11:32:37 2017 GMT
            Not After : Dec 31 23:59:59 2018 GMT
        Subject: C=US, O=Google Inc, CN=Google Internet Authority G2
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
```

Step 3:

Signature Algorithm: sha256WithRSAEncryption

5c:10:7b:26:7d:e7:e8:71:5d:d3:70:c8:27:3d:b5:9d:1f:df:
53:3e:9c:f6:7a:dc:cd:bb:05:a6:6c:fa:d6:8a:8e:e3:bb:3e:
42:f3:b3:0d:7b:82:b9:96:51:29:95:70:f5:fd:4d:e0:75:bd:
6c:02:6c:92:8c:02:28:22:10:bf:50:25:de:5a:5b:32:8d:7e:
e4:10:52:f2:41:ab:a2:2c:90:bf:05:39:8a:2e:2b:87:fe:5f:
fc:2a:4a:cc:54:70:e8:91:2c:a3:a8:3d:95:5e:5c:02:dc:20:
9b:c9:f1:5b:21:c5:ba:f6:df:9a:30:03:10:33:bf:c0:f6:b8:
4c:00:de:4a:47:ed:bf:ca:df:5c:71:0c:db:f1:2b:4a:83:d3:
f7:1d:33:11:05:ef:2c:dd:7f:7d:ae:8b:39:cd:34:34:fb:9c:
61:74:a3:17:a7:0d:a6:29:25:70:62:a6:8c:dc:f4:28:2c:95:
36:14:87:f5:2e:96:f6:1e:8e:32:22:1b:91:ac:e8:85:eb:ba:
c5:03:c5:29:17:54:46:1a:9d:1f:33:65:61:34:db:9d:82:62:
6d:52:67:c6:6e:67:08:a3:6e:82:93:70:48:b8:af:15:af:d5:
06:43:68:e1:63:63:19:7e:97:c1:56:6b:ae:98:a1:61:aa:d1:
95:e3:85:03

Step 4:

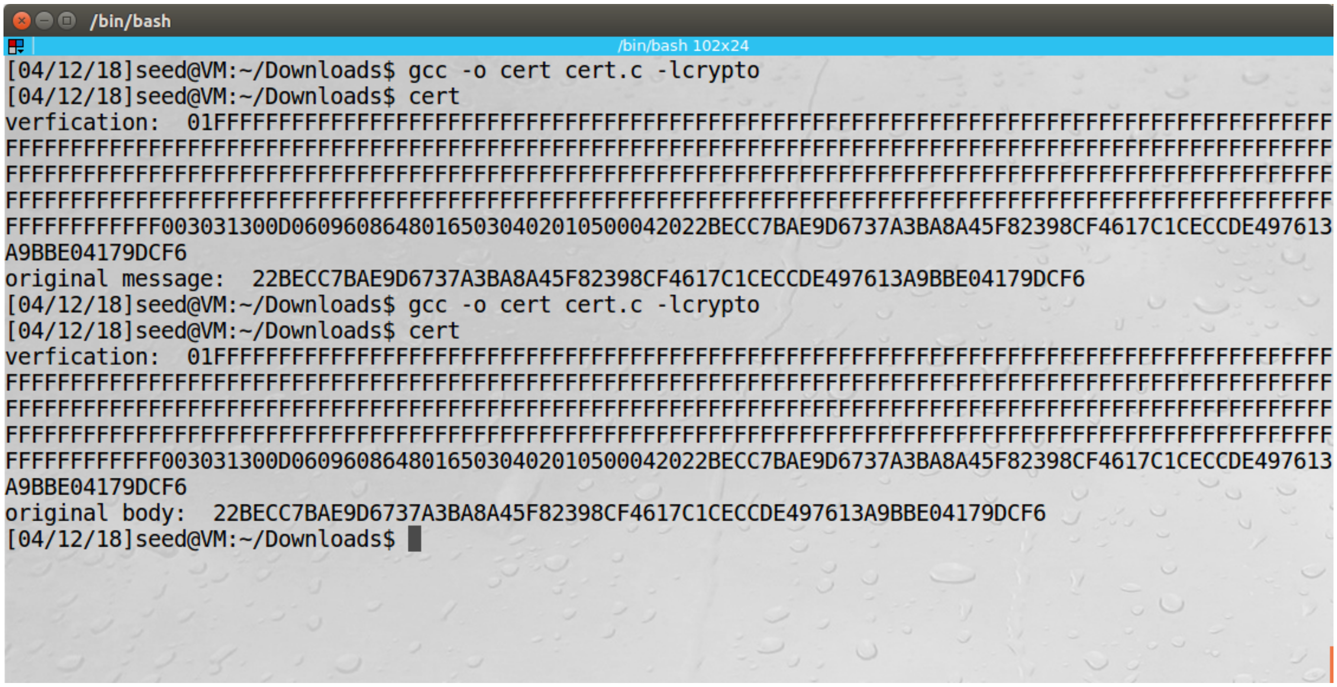
5c107b267de7e8715dd370c8273db59d1fdf533e9cf67adccdbb05a66cfad68a8ee3bb3e42f3b30d7b82b9
9651299570f5fd4de075bd6c026c928c02282210bf5025de5a5b328d7ee41052f241aba22c90bf05398a2e
2b87fe5ffc2a4acc5470e8912ca3a83d955e5c02dc209bc9f15b21c5baf6df9a30031033bfc0f6b84c00de4a
47edbfcadf5c710cdbf12b4a83d3f71d331105ef2cdd7f7dae8b39cd3434fb9c6174a317a70da629257062a
68cdcf4282c95361487f52e96f61e8e32221b91ace885ebbac503c5291754461a9d1f33656134db9d82626
d5267c66e6708a36e82937048b8af15afd5064368e16363197e97c1566bae98a161aad195e38503

```
/bin/bash
/bin/bash 102x24
                Policy: 2.23.140.1.2.2

            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://pki.google.com/GIAG2.crl

    Signature Algorithm: sha256WithRSAEncryption
         5c:10:7b:26:7d:e7:e8:71:5d:d3:70:c8:27:3d:b5:9d:1f:df:
         53:3e:9c:f6:7a:dc:cd:bb:05:a6:6c:fa:d6:8a:8e:e3:bb:3e:
         42:f3:b3:0d:7b:82:b9:96:51:29:95:70:f5:fd:4d:e0:75:bd:
         6c:02:6c:92:8c:02:28:22:10:bf:50:25:de:5a:5b:32:8d:7e:
         e4:10:52:f2:41:ab:a2:2c:90:bf:05:39:8a:2e:2b:87:fe:5f:
         fc:2a:4a:cc:54:70:e8:91:2c:a3:a8:3d:95:5e:5c:02:dc:20:
         9b:c9:f1:5b:21:c5:ba:f6:df:9a:30:03:10:33:bf:c0:f6:b8:
         4c:00:de:4a:47:ed:bf:ca:df:5c:71:0c:db:f1:2b:4a:83:d3:
         f7:1d:33:11:05:ef:2c:dd:7f:7d:ae:8b:39:cd:34:34:fb:9c:
         61:74:a3:17:a7:0d:a6:29:25:70:62:a6:8c:dc:f4:28:2c:95:
         36:14:87:f5:2e:96:f6:1e:8e:32:22:1b:91:ac:e8:85:eb:ba:
         c5:03:c5:29:17:54:46:1a:9d:1f:33:65:61:34:db:9d:82:62:
         6d:52:67:c6:6e:67:08:a3:6e:82:93:70:48:b8:af:15:af:d5:
         06:43:68:e1:63:63:19:7e:97:c1:56:6b:ae:98:a1:61:aa:d1:
         95:e3:85:03
[04/12/18]seed@VM:~/Downloads$
```

Hash: 22becc7bae9d6737a3ba8a45f82398cf4617c1ceccde497613a9bbe04179dcf6

```
/bin/bash
/bin/bash 102x24
[04/12/18]seed@VM:~/Downloads$ openssl asn1parse -i -in c0.pem -strparse 4 -out c0_body.bin -noout
[04/12/18]seed@VM:~/Downloads$ sha256sum c0_body.bin
22becc7bae9d6737a3ba8a45f82398cf4617c1ceccde497613a9bbe04179dcf6  c0_body.bin
[04/12/18]seed@VM:~/Downloads$
```

Step 5:



```
[04/12/18]seed@VM:~/Downloads$ gcc -o cert cert.c -lcrypto
[04/12/18]seed@VM:~/Downloads$ cert
verfication:  01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFF003031300D06096086480165030402010500042022BECC7BAE9D6737A3BA8A45F82398CF4617C1CECCDE497613
A9BBE04179DCF6
original message:  22BECC7BAE9D6737A3BA8A45F82398CF4617C1CECCDE497613A9BBE04179DCF6
[04/12/18]seed@VM:~/Downloads$ gcc -o cert cert.c -lcrypto
[04/12/18]seed@VM:~/Downloads$ cert
verfication:  01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFF003031300D06096086480165030402010500042022BECC7BAE9D6737A3BA8A45F82398CF4617C1CECCDE497613
A9BBE04179DCF6
original body:  22BECC7BAE9D6737A3BA8A45F82398CF4617C1CECCDE497613A9BBE04179DCF6
[04/12/18]seed@VM:~/Downloads$ █
```

Code:

```
#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 *body = BN_new();
	BIGNUM *signature = BN_new();
	BIGNUM *verify = BN_new();

	BN_hex2bn(&n,
"9C2A04775CD850913A06A382E0D85048BC893FF119701A88467EE08FC5F189CE21EE5AFE610
DB7324489A0740B534F55A4CE826295EEEB595FC6E1058012C45E943FBC5B4838F453F724E6F
B91E915C4CFF4530DF44AFC9F54DE7DBEA06B6F87C0D0501F28300340DA0873516C7FFF3A3
CA737068EBD4B1104EB7D24DEE6F9FC3171FB94D560F32E4AAF42D2CBEAC46A1AB2CC53
```

```
DD154B8B1FC819611FCD9DA83E632B8435696584C819C54622F85395BEE3804A10C62AECBA
972011C739991004A0F0617A95258C4E5275E2B6ED08CA14FCCE226AB34ECF46039797037EC0
B1DE7BAF4533CFBA3E71B7DEF42525C20D35899D9DFB0E1179891E37C5AF8E7269");
	BN_hex2bn(&e, "10001");
	BN_hex2bn(&body,
"22becc7bae9d6737a3ba8a45f82398cf4617c1ceccde497613a9bbe04179dcf6");
	BN_hex2bn(&signature,
"5c107b267de7e8715dd370c8273db59d1fdf533e9cf67adccdbb05a66cfad68a8ee3bb3e42f3b30d7b82b
99651299570f5fd4de075bd6c026c928c02282210bf5025de5a5b328d7ee41052f241aba22c90bf05398a2
e2b87fe5ffc2a4acc5470e8912ca3a83d955e5c02dc209bc9f15b21c5baf6df9a30031033bfc0f6b84c00de4
a47edbfcadf5c710cdbf12b4a83d3f71d331105ef2cdd7f7dae8b39cd3434fb9c6174a317a70da629257062
a68cdcf4282c95361487f52e96f61e8e32221b91ace885ebbac503c5291754461a9d1f33656134db9d8262
6d5267c66e6708a36e82937048b8af15afd5064368e16363197e97c1566bae98a161aad195e38503");

	// verify S^e mod n
	BN_mod_exp(verify, signature, e, n, ctx);
	printBN("verfication: ", verify);
	printBN("original body: ", body);

	return 0;
}
```