



TASK 1:



TASK 2:

```

Task 2: Encrypting a Message 4 4120746f702073656372657421
qasim@ubuntu: ~/Downloads/Labsetup
qasim@ubuntu:~/Downloads/Labsetup$ gcc encrypt.c -o encrypt -lcrypto
qasim@ubuntu:~/Downloads/Labsetup$ ./encrypt
encryption of message = 6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75ACADC5DE5CFC5FADC
decryption of message = 4120746f702073656372657421
qasim@ubuntu:~/Downloads/Labsetup$ python3 -c 'print("A top secret!".encode("utf-8").hex())'
4120746f702073656372657421
qasim@ubuntu:~/Downloads/Labsetup$

```

```

1 #include <stdio.h>
2 #include <openssl/bn.h>
3 #define NBITS 256
4
5 void printBNhex(char *msg, BIGNUM * a)
6 {
7     /* Use BN_bn2hex(a) for hex string*/
8     char * number_str = BN_bn2hex(a);
9     printf("%s %s\n", msg, number_str);
10    OPENSSL_free(number_str);
11 }
12 int main ()
13 {
14     BN_CTX *ctx = BN_CTX_new();
15     BIGNUM *e = BN_new();
16     BIGNUM *n = BN_new();
17     BIGNUM *M = BN_new();
18     BIGNUM *c = BN_new();
19     BIGNUM *d = BN_new();
20     BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDD3A4D0CB81629242FB1A5");
21     BN_hex2bn(&e, "010001");
22     BN_hex2bn(&M, "4120746f702073656372657421");
23     BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
24     BN_mod_exp(c, M, e, n, ctx);
25     printBNhex("encryption of message = ", c);
26     BN_mod_exp(M, c, d, n, ctx);
27     printBNhex("decryption of message = ", M);
28
29 }

```

```

]

```

TASK 3:

```
Task 3: Decrypting a Message 5 C = 8C0F971DF2F3672B28811407E2DABBE1DA0FE8BBD9FC7DCB67396567EA1E2493F

qasim@ubuntu: ~/Downloads/Labsetup
qasim@ubuntu:~/Downloads/Labsetup$ ./decrypt
Decrypted message (M) in hex: 50617373776F72642069732064656573
qasim@ubuntu:~/Downloads/Labsetup$ python3 -c 'print(bytes.fromhex("50617373776F72642069732064656573").decode("utf-8"))'
Password is dees
qasim@ubuntu:~/Downloads/Labsetup$
```

```
Open ▼ decrypt.c
~/Downloads/Labsetup

1 #include <stdio.h>
2 #include <openssl/bn.h>
3 void printBN(char *msg, BIGNUM *a) {
4     char *number_str = BN_bn2hex(a);
5     printf("%s %s\n", msg, number_str);
6     OPENSSL_free(number_str);
7 }
8 int main() {
9     BN_CTX *ctx = BN_CTX_new();
10    BIGNUM *n = BN_new();
11    BIGNUM *d = BN_new();
12    BIGNUM *C = BN_new();
13    BIGNUM *M = BN_new();
14    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
15    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
16    BN_hex2bn(&C, "8C0F971DF2F3672B28811407E2DABBE1DA0FE8BBD9FC7DCB67396567EA1E2493F");
17    BN_mod_exp(M, C, d, n, ctx);
18    printBN("Decrypted message (M) in hex: ", M);
19 }
```

TASK 4:

Task 4: Signing a Message

5

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:

```

qasim@ubuntu: ~/Downloads/Labsetup
qasim@ubuntu:~/Downloads/Labsetup$ gcc sign.c -o sign -lcrypto
qasim@ubuntu:~/Downloads/Labsetup$ ./sign
Encryption of message "$2000." = 55A4E7F17F04CCFE2766E1EB32ADDBA890BBE92A6FBE2D785ED6E73CCB35E4CB
Encryption of message "$3000." = BCC20FB7568E5D48E434C387C06A6025E90D29D848AF9C3EBAC0135D99305822
qasim@ubuntu:~/Downloads/Labsetup$

```

```

1 #include <stdio.h>
2 #include <openssl/bn.h>
3 #define NBITS 256
4
5 void printBNhex(char *msg, BIGNUM * a){
6     char *number_str = BN_bn2hex(a);
7     printf("%s %s\n", msg, number_str);
8     OPENSSL_free(number_str);
9 }
10
11 int main () {
12     BN_CTX *ctx = BN_CTX_new();
13     BIGNUM *e = BN_new();
14     BIGNUM *n = BN_new();
15     BIGNUM *M1 = BN_new();
16     BIGNUM *M2 = BN_new();
17     BIGNUM *c1 = BN_new();
18     BIGNUM *c2 = BN_new();
19     BIGNUM *d = BN_new();
20
21     BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
22     BN_hex2bn(&e, "010001");
23     BN_hex2bn(&M1, "49206F776520796F752024323030302E");
24     BN_hex2bn(&M2, "49206F776520796F752024333030302E");
25     BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
26
27     // Encrypting $2000
28     BN_mod_exp(c1, M1, d, n, ctx);
29     printBNhex("Encryption of message \"$2000.\" = ", c1);
30
31     // Encrypting $3000
32     BN_mod_exp(c2, M2, d, n, ctx);
33     printBNhex("Encryption of message \"$3000.\" = ", c2);
34
35

```

TASK 5:

Task 5: Verifying a Signature

5

```

qasim@ubuntu: ~/Downloads/Labsetup
qasim@ubuntu:~/Downloads/Labsetup$ gcc verify.c -o verify -lcrypto
qasim@ubuntu:~/Downloads/Labsetup$ ./verify
Decrypted message hex = 4C61756E63682061206D697373696C652E
Expected message hex = 4C61756E63682061206D697373696C652E
Signature is valid.
qasim@ubuntu:~/Downloads/Labsetup$

```

```

1 #include <stdio.h>
2 #include <openssl/bn.h>
3 #include <string.h>
4 #define NBITS 256
5 void printBNhex(char *msg, BIGNUM *a) {
6     char *number_str = BN_bn2hex(a);
7     printf("%s %s\n", msg, number_str);
8     OPENSSL_free(number_str);
9 }
10 void convertMessageToHex(const char *msg, BIGNUM *msg_bn) {
11
12     size_t len = strlen(msg);
13     char hex_str[len * 2 + 1];
14     for (size_t i = 0; i < len; i++) {
15         sprintf(&hex_str[i * 2], "%02x", (unsigned char)msg[i]);
16     }
17     hex_str[len * 2] = '\0';
18     BN_hex2bn(&msg_bn, hex_str);
19 }
20 int main() {
21
22     BN_CTX *ctx = BN_CTX_new();
23     BIGNUM *e = BN_new();
24     BIGNUM *n = BN_new();
25     BIGNUM *S = BN_new();
26     BIGNUM *m = BN_new();
27     BIGNUM *expected_msg = BN_new();
28     BN_hex2bn(&n, "AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");
29     BN_hex2bn(&e, "010001");
30     BN_hex2bn(&S, "643D6F34902D9C7EC90CB082BCA36C47FA37165C0005CAB026C0542CBD86802F");
31     BN_mod_exp(m, S, e, n, ctx);
32     printBNhex("Decrypted message hex = ", m);
33     convertMessageToHex("Launch a missile.", expected_msg);
34     printBNhex("Expected message hex = ", expected_msg);
35     if (BN_cmp(m, expected_msg) == 0) {
36         printf("Signature is valid.\n");
37     } else {
38         printf("Signature is invalid.\n");
39     }
40 }

```


3. **Signature Extraction:** Extracted and formatted the signature from the server's certificate using `openssl x509`.

```
gasim@ubuntu:~/Downloads/Labsetup$ openssl x509 -in c0.pem -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            0d:25:d1:ef:e1:6c:ba:1c:42:2c:c8:d3:ac:56:82:38
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 High Assurance
Server CA
    Validity
        Not Before: Jul 22 00:00:00 2024 GMT
        Not After : Oct 20 23:59:59 2024 GMT
    Subject: C = US, ST = California, L = Menlo Park, O = "Meta Platforms, Inc.", CN = *.facebook.com
```

4. Body Extraction: Used openssl asn1parse to extract the certificate body and saved it as c0_body.bin.

```
qasim@ubuntu:~/Downloads/Labsetup$ cat c0.pem | grep "Signature" | awk '{print $2}' | tr -d ':' > signature.hex
qasim@ubuntu:~/Downloads/Labsetup$ openssl asn1parse -in c0.pem -out c0_body.bin
 0:d=0  hl=4  l=1688  cons: SEQUENCE
 4:d=1  hl=4  l=1408  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           :0D25D1EFE16CBA1C422CC8D3AC568238
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= 112  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
```

5. Hashing: Computed the SHA-256 hash of c0 body.bin to prepare for signature verification.

```
gasim@ubuntu:~/Downloads/Labsetup$ sha256sum c0_body.bin > hash.txt
gasim@ubuntu:~/Downloads/Labsetup$
```

6. **Signature Verification:** Verified the signature using the extracted public key and noted the differences in hashes during verification.

```
i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert SHA2 High Assurance Server CA
```

```
Task 6: Manually Verifying an X... 6
```

```
qasim@ubuntu: ~/Downloads/Labsetup
```

```
qasim@ubuntu:~/Downloads/Labsetup$ gcc fb.c -o fb -lcrypto
```

```
qasim@ubuntu:~/Downloads/Labsetup$ ./fb
```

```
message hex = 01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF003031300D06096  
086480165030402010500042012CA8F2F03B99047253CDB5677343C45007BF5894912FF907654F7AC6F0E5350
```

```
qasim@ubuntu:~/Downloads/Labsetup$
```