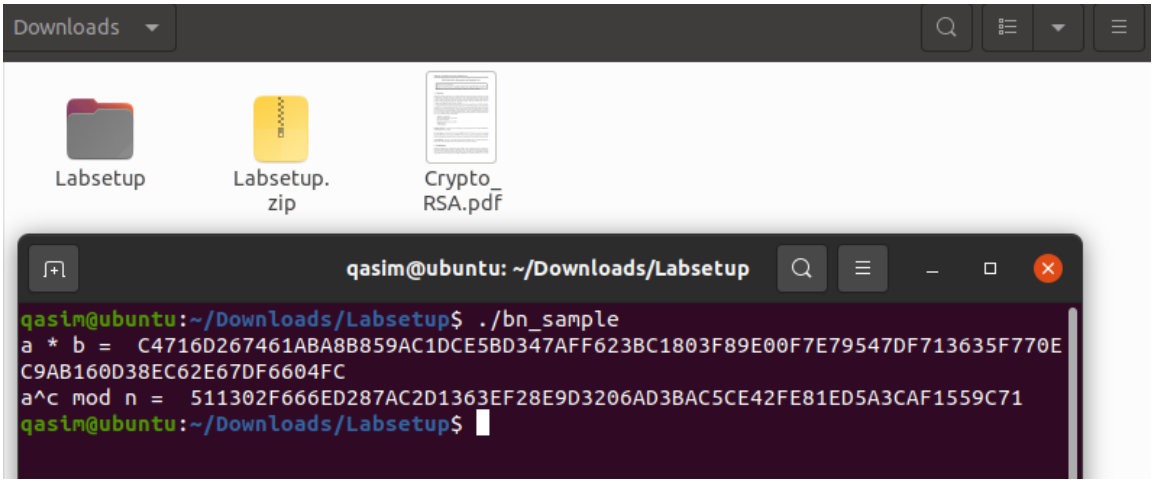
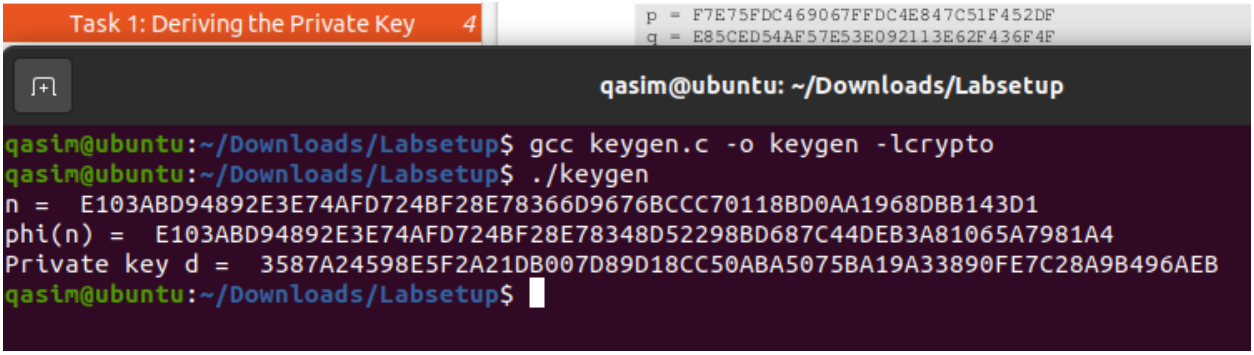
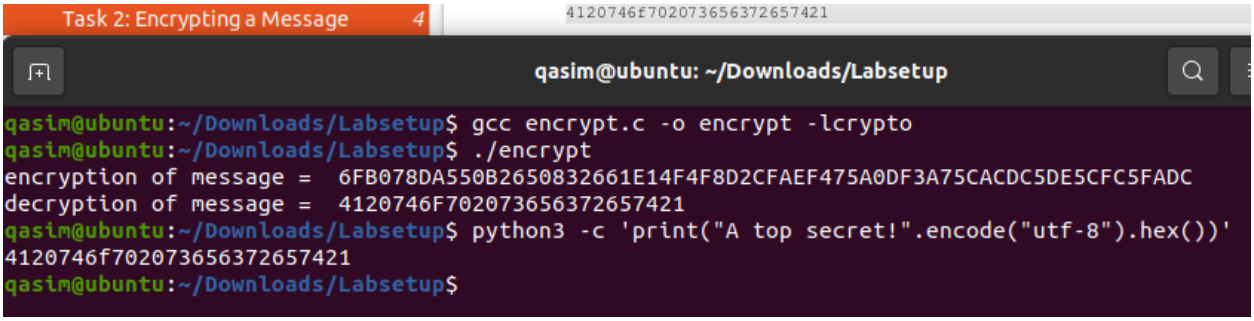


TASK 1:




```c
#include <stdio.h>
#include <openssl/bn.h>
void printBN(char *msg, BIGNUM *a) {
    char *num_str = BN_bn2hex(a);
    printf("%s %s\n", msg, num_str);
    OPENSSL_free(num_str);
}
int main() {
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *p = BN_new(), *q = BN_new(), *n = BN_new(), *e = BN_new(), *phi = BN_new();
    BIGNUM *p_1 = BN_new(), *q_1 = BN_new(), *d = BN_new()
    BN_hex2bn(&p, "F7E75FDC469067FFDC4E847C51F452DF");
    BN_hex2bn(&q, "E85CED54AF57E53E092113E62F436F4F");
    BN_hex2bn(&e, "0D88C3");
    BN_mul(n, p, q, ctx); printBN("n = ", n);
    BN_sub(p_1, p, BN_value_one());
    BN_sub(q_1, q, BN_value_one());
    BN_mul(phi, p_1, q_1, ctx); printBN("phi(n) = ", phi);
    BN_mod_inverse(d, e, phi, ctx); printBN("Private key d = ", d);
}
```

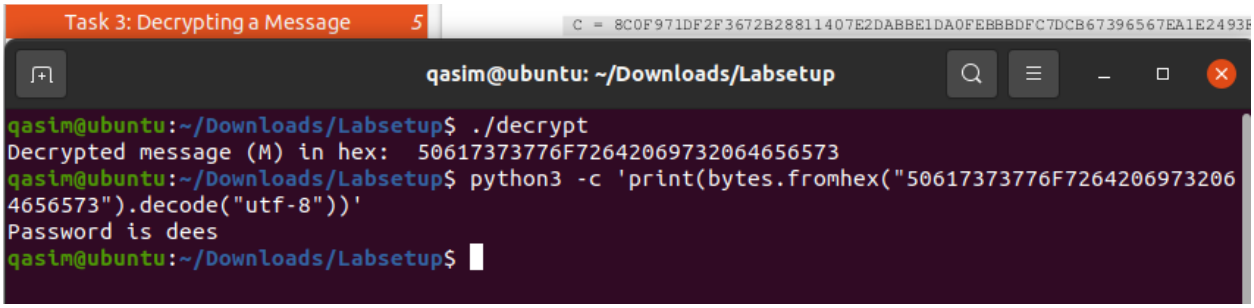
TASK 2:




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

void printBNhex(char *msg, BIGNUM * a)
{
        /* Use BN_bn2hex(a) for hex 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 *e = BN_new();
        BIGNUM *n = 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");
        BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
        BN_mod_exp(c, M, e, n, ctx);
        printBNhex("encryption of message = ", c);
        BN_mod_exp(M, c, d, n, ctx);
        printBNhex("decryption of message = ", M);

}
```

]

TASK 3:

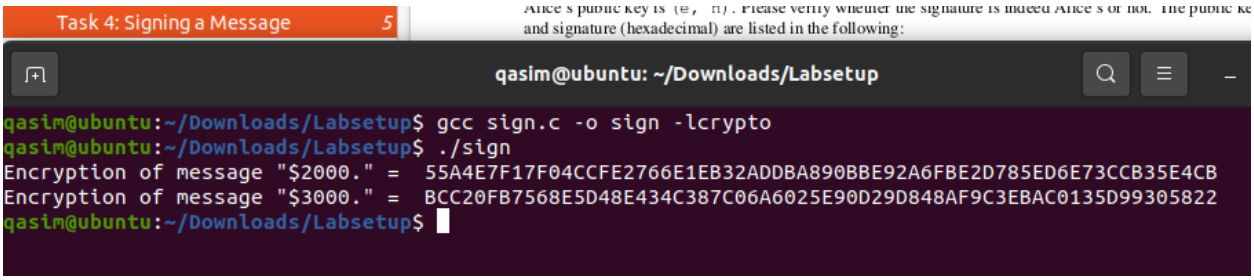


Open

decrypt.c
~/Downloads/Labsetup

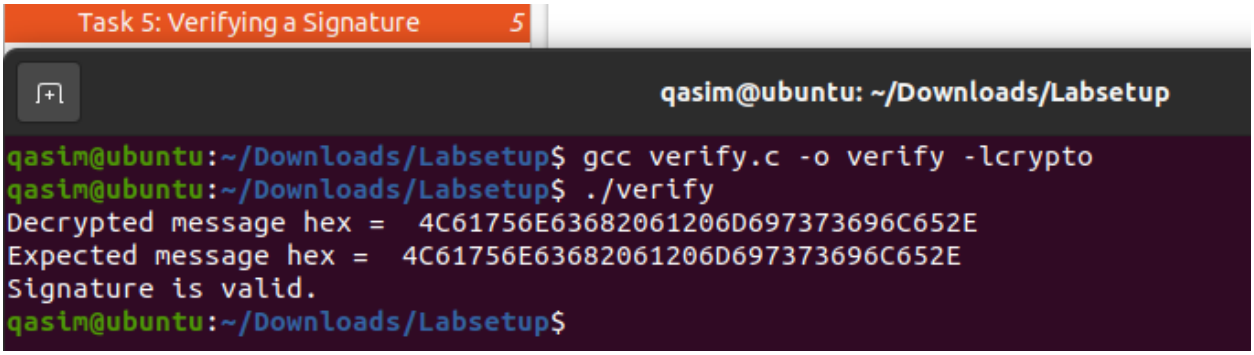
```c
#include <stdio.h>
#include <openssl/bn.h>
void printBN(char *msg, BIGNUM *a) {
    char *number_str = BN_bn2hex(a);
    printf("%s %s\n", msg, number_str);
    OPENSSL_free(number_str);
}
int main() {
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *n = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *C = BN_new();
    BIGNUM *M = BN_new();
    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
    BN_hex2bn(&C, "8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F");
    BN_mod_exp(M, C, d, n, ctx);
    printBN("Decrypted message (M) in hex: ", M);
}
```

TASK 4:




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

void printBNhex(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 *e = BN_new();
    BIGNUM *n = 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(&e, "010001");
    BN_hex2bn(&M1, "49206F776520796F752024323030302E");
    BN_hex2bn(&M2, "49206F776520796F752024333030302E");
    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

    // Encrypting $2000
    BN_mod_exp(c1, M1, d, n, ctx);
    printBNhex("Encryption of message \"$2000.\" = ", c1);

    // Encrypting $3000
    BN_mod_exp(c2, M2, d, n, ctx);
    printBNhex("Encryption of message \"$3000.\" = ", c2);

}
```

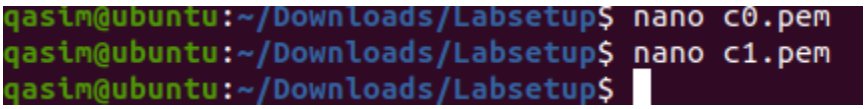
TASK 5:



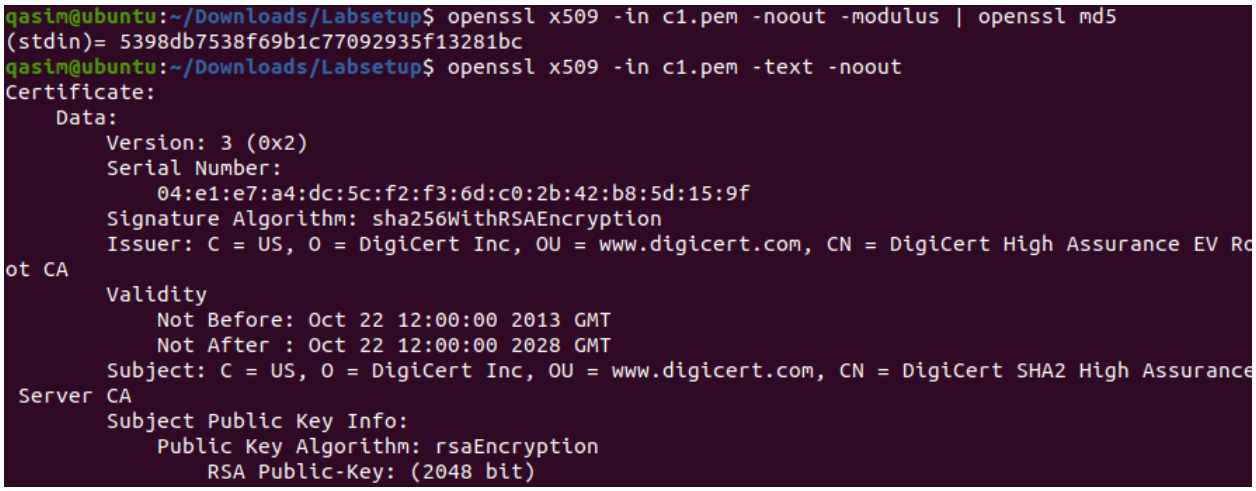

```c
#include <stdio.h>
#include <openssl/bn.h>
#include <string.h>
#define NBITS 256
void printBNhex(char *msg, BIGNUM *a) {
    char *number_str = BN_bn2hex(a);
    printf("%s %s\n", msg, number_str);
    OPENSSL_free(number_str);
}
void convertMessageToHex(const char *msg, BIGNUM *msg_bn) {

    size_t len = strlen(msg);
    char hex_str[len * 2 + 1];
    for (size_t i = 0; i < len; i++) {
        sprintf(&hex_str[i * 2], "%02x", (unsigned char)msg[i]);
    }
    hex_str[len * 2] = '\0';
    BN_hex2bn(&msg_bn, hex_str);
}
int main() {

    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *e = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *S = BN_new();
    BIGNUM *m = BN_new();
    BIGNUM *expected_msg = BN_new();
    BN_hex2bn(&n, "AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&S, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F");
    BN_mod_exp(m, S, e, n, ctx);
    printBNhex("Decrypted message hex = ", m);
    convertMessageToHex("Launch a missile.", expected_msg);
    printBNhex("Expected message hex = ", expected_msg);
    if (BN_cmp(m, expected_msg) == 0) {
        printf("Signature is valid.\n");
    } else {
        printf("Signature is invalid.\n");
    }
}
```

TASK 6:

1. Certificate Retrieval: Retrieved Facebook's certificates using openssl s_client and saved them as c0.pem and c1.pem.



2. Public Key Extraction: Extracted the public key (n,e) from the issuer's certificate.



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

```
qasim@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 = *.facebo
ok.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.

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

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

```
qasim@ubuntu:~/Downloads/Labsetup$ gcc fb.c -o fb -lcrypto
qasim@ubuntu:~/Downloads/Labsetup$ ./fb
message hex =  01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF003031300D06096
086480165030402010500042012CA8F2F03B99047253CDB5677343C45007BF5894912FF907654F7AC6F0E5350
qasim@ubuntu:~/Downloads/Labsetup$
```