ACIT 4630 – Lab 5 – Cryptography

Markus Afonso, Devin Kandola
February 12, 2024
Group 41, Set C

Q1. Note the name of the algorithm. What does each part refer to (aes, 128, cbc)?

- AES: Advanced Encryption Standard, a symmetric encryption algorithm.
- 128: Refers to the key size (128-bit key in this case).
- CBC: Cipher Block Chaining, a mode of operation for block ciphers like AES.

Q2. What is the plaintext of the message?

In cryptography, the one-time pad (OTP) is an encryption technique that cannot be cracked, but requires the use of a single-use pre-shared key that is no smaller than the message being sent. In this technique, a plaintext is paired with a random secret key (also referred to as a one-time pad). Then, each bit or character of the plaintext is encrypted by combining it with the corresponding bit or character from the pad using modular addition.

https://en.wikipedia.org/wiki/One-time_pad

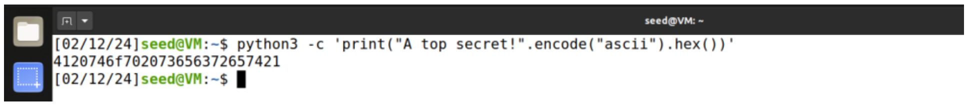
Q3. Is the provided key part of a key pair?

No, the provided key looks to be a symmetric encryption key used for AES

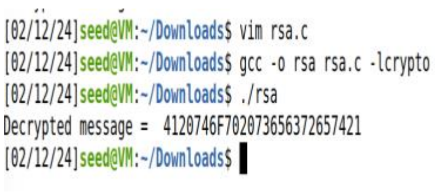
Q4. Should this key be shared publicly?

No, the key should not be shared publicly. Since it is a symmetrical key, anyone with the key can access sensitive information.

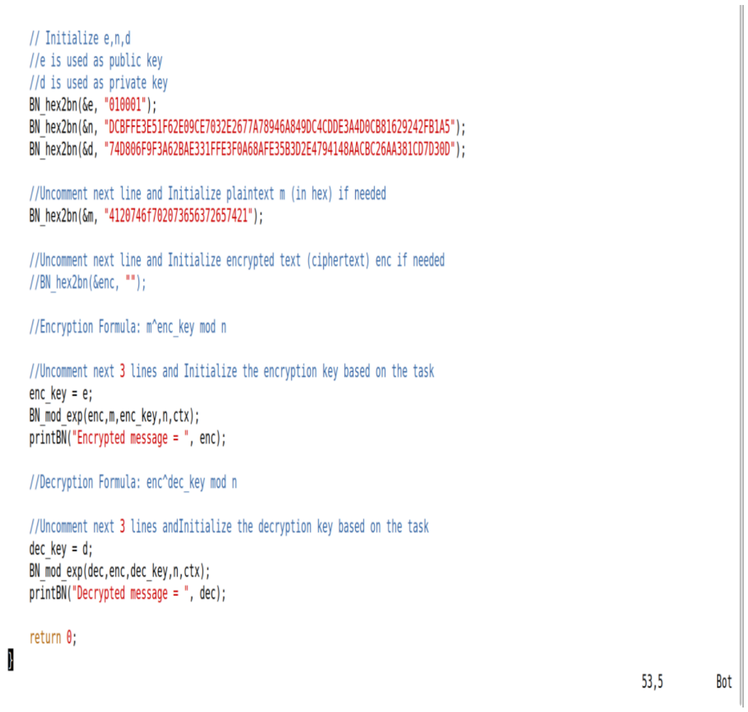
Q5. What's the ciphertext?



```
[02/12/24]seed@VM:~$ python3 -c 'print("A top secret!".encode("ascii").hex())'
4120746f702073656372657421
[02/12/24]seed@VM:~$
```

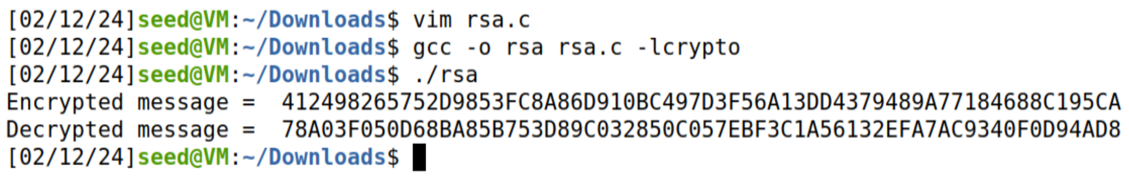


```
[02/12/24]seed@VM:~/Downloads$ vim rsa.c
[02/12/24]seed@VM:~/Downloads$ gcc -o rsa rsa.c -lcrypto
[02/12/24]seed@VM:~/Downloads$ ./rsa
Decrypted message =  4120746F702073656372657421
[02/12/24]seed@VM:~/Downloads$
```

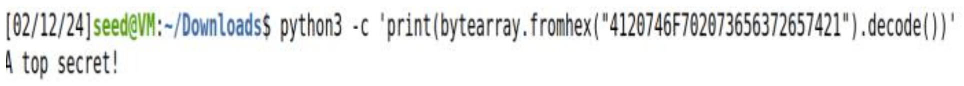


```
    // Initialize e,n,d
    //e is used as public key
    //d is used as private key
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

    //Uncomment next line and Initialize plaintext m (in hex) if needed
    BN_hex2bn(&m, "4120746f702073656372657421");

    //Uncomment next line and Initialize encrypted text (ciphertext) enc if needed
    //BN_hex2bn(&enc, "");

    //Encryption Formula: m^enc_key mod n

    //Uncomment next 3 lines and Initialize the encryption key based on the task
    enc_key = e;
    BN_mod_exp(enc,m,enc_key,n,ctx);
    printBN("Encrypted message = ", enc);

    //Decryption Formula: enc^dec_key mod n

    //Uncomment next 3 lines andInitialize the decryption key based on the task
    dec_key = d;
    BN_mod_exp(dec,enc,dec_key,n,ctx);
    printBN("Decrypted message = ", dec);

    return 0;
}
                                                        53,5          Bot
```

Q6. Verify that it can be decrypted back to the original message



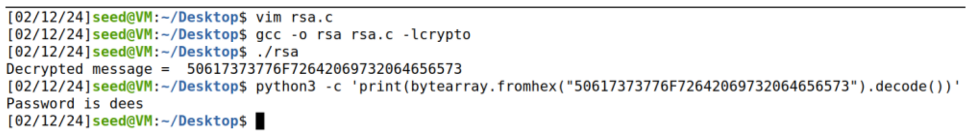
```
[02/12/24]seed@VM:~/Downloads$ vim rsa.c
[02/12/24]seed@VM:~/Downloads$ gcc -o rsa rsa.c -lcrypto
[02/12/24]seed@VM:~/Downloads$ ./rsa
Encrypted message =  412498265752D9853FC8A86D910BC497D3F56A13DD4379489A77184688C195CA
Decrypted message =  78A03F050D68BA85B753D89C032850C057EBF3C1A56132EFA7AC9340F0D94AD8
[02/12/24]seed@VM:~/Downloads$ 
```



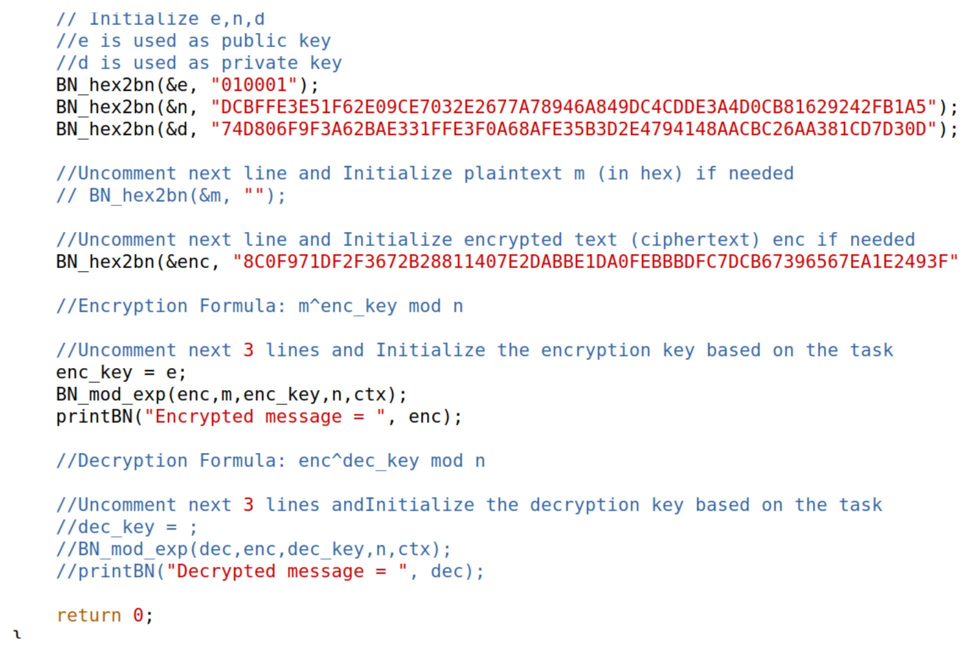
```
[02/12/24]seed@VM:~/Downloads$ python3 -c 'print(bytearray.fromhex("4120746F702073656372657421").decode())'
A top secret!
```

Q7. What's the message in plain text?

‘Password is dees’



```
[02/12/24]seed@VM:~/Desktop$ vim rsa.c
[02/12/24]seed@VM:~/Desktop$ gcc -o rsa rsa.c -lcrypto
[02/12/24]seed@VM:~/Desktop$ ./rsa
Decrypted message =  50617373776F72642069732064656573
[02/12/24]seed@VM:~/Desktop$ python3 -c 'print(bytearray.fromhex("50617373776F72642069732064656573").decode())'
Password is dees
[02/12/24]seed@VM:~/Desktop$ 
```

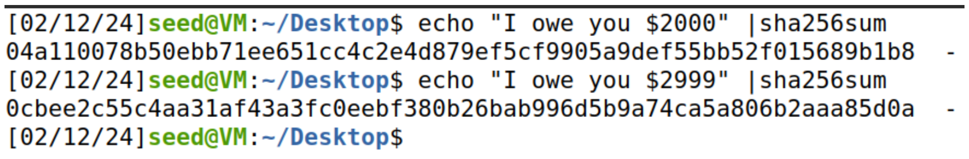


```
    // Initialize e,n,d
    //e is used as public key
    //d is used as private key
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

    //Uncomment next line and Initialize plaintext m (in hex) if needed
    // BN_hex2bn(&m, "");

    //Uncomment next line and Initialize encrypted text (ciphertext) enc if needed
    BN_hex2bn(&enc, "8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F"

    //Encryption Formula: m^enc_key mod n

    //Uncomment next 3 lines and Initialize the encryption key based on the task
    enc_key = e;
    BN_mod_exp(enc,m,enc_key,n,ctx);
    printBN("Encrypted message = ", enc);

    //Decryption Formula: enc^dec_key mod n

    //Uncomment next 3 lines andInitialize the decryption key based on the task
    //dec_key = ;
    //BN_mod_exp(dec,enc,dec_key,n,ctx);
    //printBN("Decrypted message = ", dec);

    return 0;
}
```

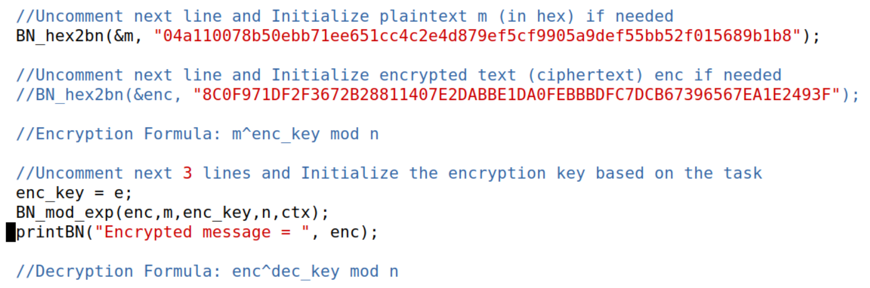
Q8. Compare both signatures and describe what you observe.

They are entirely different:

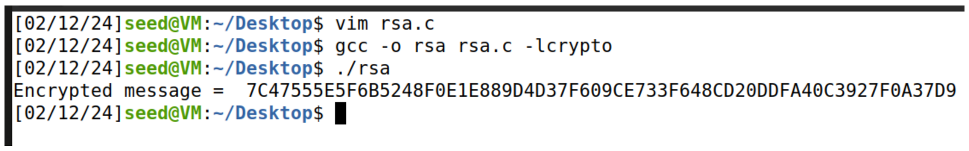
7C47555E5F6B5248F0E1E889D4D37F609CE733F648CD20DDFA40C3927F0A37D9
BC44B25DD4DE6802B521D62A7276E6B99F020EC94A3EE83A45D8C92A9C2D654D



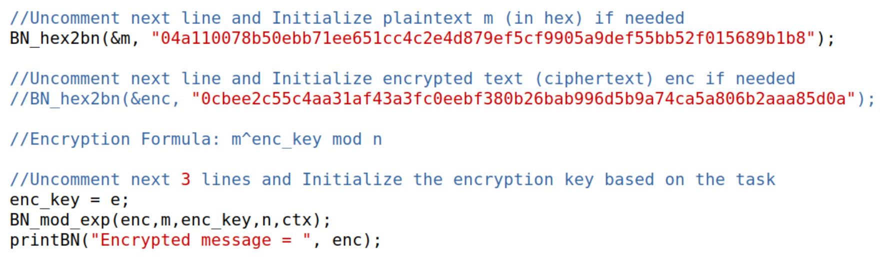
```
[02/12/24]seed@VM:~/Desktop$ echo "I owe you $2000" |sha256sum
04a110078b50ebb71ee651cc4c2e4d879ef5cf9905a9def55bb52f015689b1b8  -
[02/12/24]seed@VM:~/Desktop$ echo "I owe you $2999" |sha256sum
0cbee2c55c4aa31af43a3fc0eebf380b26bab996d5b9a74ca5a806b2aaa85d0a  -
[02/12/24]seed@VM:~/Desktop$
```



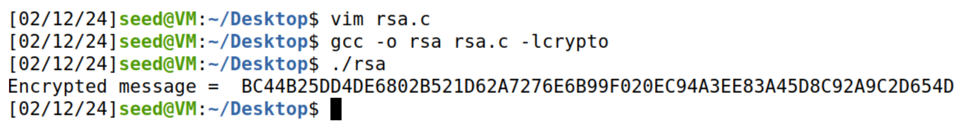
```
  //Uncomment next line and Initialize plaintext m (in hex) if needed
  BN_hex2bn(&m, "04a110078b50ebb71ee651cc4c2e4d879ef5cf9905a9def55bb52f015689b1b8");

  //Uncomment next line and Initialize encrypted text (ciphertext) enc if needed
  //BN_hex2bn(&enc, "8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F");

  //Encryption Formula: m^enc_key mod n

  //Uncomment next 3 lines and Initialize the encryption key based on the task
  enc_key = e;
  BN_mod_exp(enc,m,enc_key,n,ctx);
  printBN("Encrypted message = ", enc);

  //Decryption Formula: enc^dec_key mod n
```



```
[02/12/24]seed@VM:~/Desktop$ vim rsa.c
[02/12/24]seed@VM:~/Desktop$ gcc -o rsa rsa.c -lcrypto
[02/12/24]seed@VM:~/Desktop$ ./rsa
Encrypted message =  7C47555E5F6B5248F0E1E889D4D37F609CE733F648CD20DDFA40C3927F0A37D9
[02/12/24]seed@VM:~/Desktop$ 
```



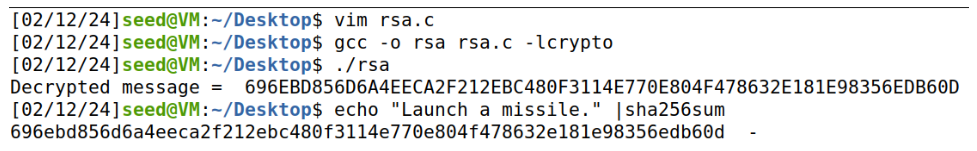
```
//Uncomment next line and Initialize plaintext m (in hex) if needed
BN_hex2bn(&m, "04a110078b50ebb71ee651cc4c2e4d879ef5cf9905a9def55bb52f015689b1b8");

//Uncomment next line and Initialize encrypted text (ciphertext) enc if needed
//BN_hex2bn(&enc, "0cbee2c55c4aa31af43a3fc0eebf380b26bab996d5b9a74ca5a806b2aaa85d0a");

//Encryption Formula: m^enc_key mod n

//Uncomment next 3 lines and Initialize the encryption key based on the task
enc_key = e;
BN_mod_exp(enc,m,enc_key,n,ctx);
printBN("Encrypted message = ", enc);
```



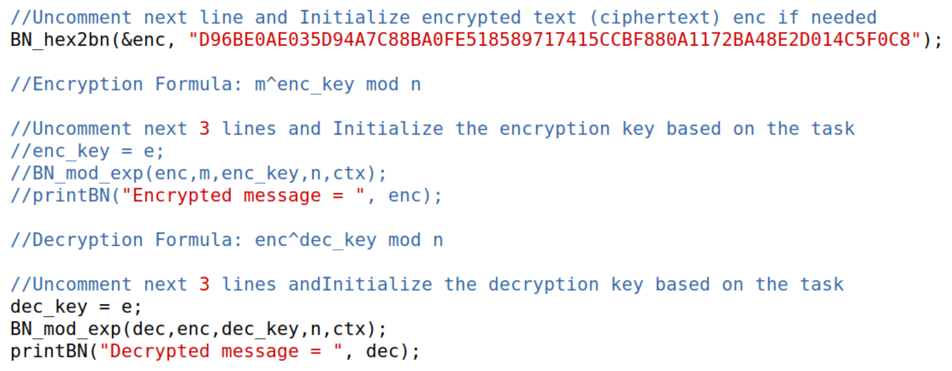
```
[02/12/24]seed@VM:~/Desktop$ vim rsa.c
[02/12/24]seed@VM:~/Desktop$ gcc -o rsa rsa.c -lcrypto
[02/12/24]seed@VM:~/Desktop$ ./rsa
Encrypted message =  BC44B25DD4DE6802B521D62A7276E6B99F020EC94A3EE83A45D8C92A9C2D654D
[02/12/24]seed@VM:~/Desktop$ 
```

Q9. Suppose that the signature is corrupted, such that the last byte of the signature changes from C8 to C9, i.e., there is only one bit of change. Please repeat this task and describe what will happen to the verification process.

With C8 they match:

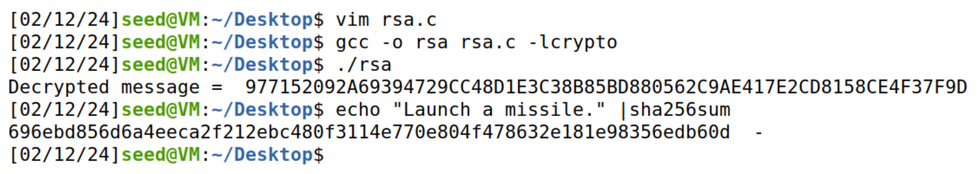


```
[02/12/24]seed@VM:~/Desktop$ vim rsa.c
[02/12/24]seed@VM:~/Desktop$ gcc -o rsa rsa.c -lcrypto
[02/12/24]seed@VM:~/Desktop$ ./rsa
Decrypted message =  696EBD856D6A4EECA2F212EBC480F3114E770E804F478632E181E98356EDB60D
[02/12/24]seed@VM:~/Desktop$ echo "Launch a missile." |sha256sum
696ebd856d6a4eeca2f212ebc480f3114e770e804f478632e181e98356edb60d  -
```

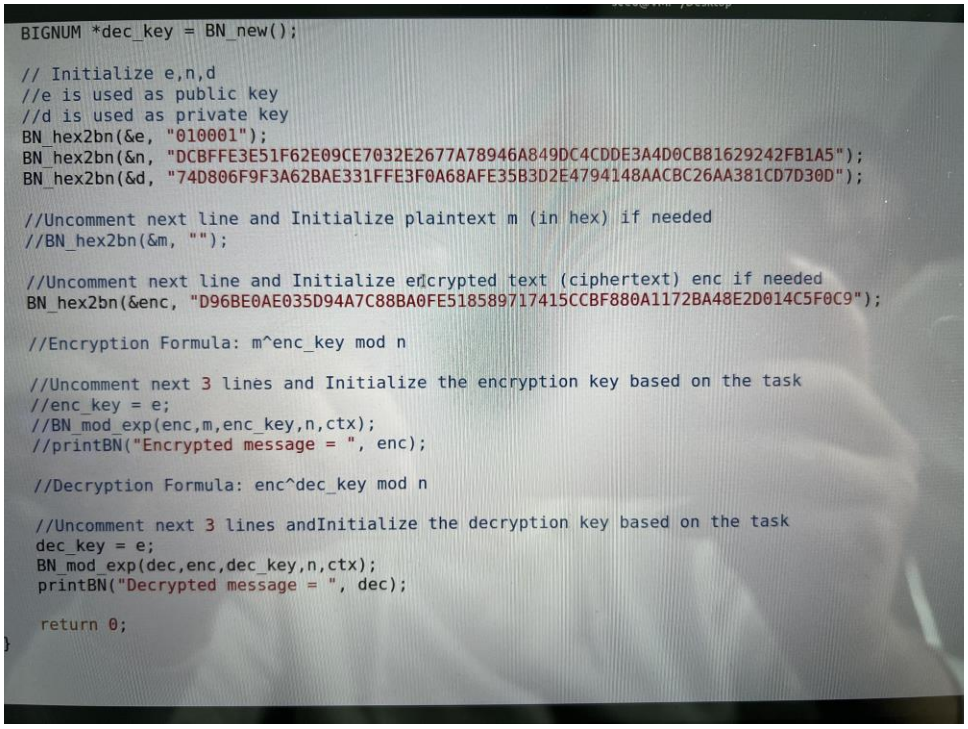


```
//Uncomment next line and Initialize encrypted text (ciphertext) enc if needed
BN_hex2bn(&enc, "D96BE0AE035D94A7C88BA0FE518589717415CCBF880A1172BA48E2D014C5F0C8");

//Encryption Formula: m^enc_key mod n

//Uncomment next 3 lines and Initialize the encryption key based on the task
//enc_key = e;
//BN_mod_exp(enc,m,enc_key,n,ctx);
//printBN("Encrypted message = ", enc);

//Decryption Formula: enc^dec_key mod n

//Uncomment next 3 lines andInitialize the decryption key based on the task
dec_key = e;
BN_mod_exp(dec,enc,dec_key,n,ctx);
printBN("Decrypted message = ", dec);
```

With C9 they're drastically different:



```
[02/12/24]seed@VM:~/Desktop$ vim rsa.c
[02/12/24]seed@VM:~/Desktop$ gcc -o rsa rsa.c -lcrypto
[02/12/24]seed@VM:~/Desktop$ ./rsa
Decrypted message =  977152092A69394729CC48D1E3C38B85BD880562C9AE417E2CD8158CE4F37F9D
[02/12/24]seed@VM:~/Desktop$ echo "Launch a missile." |sha256sum
696ebd856d6a4eeca2f212ebc480f3114e770e804f478632e181e98356edb60d  -
[02/12/24]seed@VM:~/Desktop$
```



```
BIGNUM *dec_key = BN_new();

// Initialize e,n,d
//e is used as public key
//d is used as private key
BN_hex2bn(&e, "010001");
BN_hex2bn(&n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
BN_hex2bn(&d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

//Uncomment next line and Initialize plaintext m (in hex) if needed
//BN_hex2bn(&m, "");

//Uncomment next line and Initialize encrypted text (ciphertext) enc if needed
BN_hex2bn(&enc, "D96BE0AE035D94A7C88BA0FE518589717415CCBF880A1172BA48E2D014C5F0C9");

//Encryption Formula: m^enc_key mod n

//Uncomment next 3 lines and Initialize the encryption key based on the task
//enc_key = e;
//BN_mod_exp(enc,m,enc_key,n,ctx);
//printBN("Encrypted message = ", enc);

//Decryption Formula: enc^dec_key mod n

//Uncomment next 3 lines andInitialize the decryption key based on the task
dec_key = e;
BN_mod_exp(dec,enc,dec_key,n,ctx);
printBN("Decrypted message = ", dec);

return 0;
}
```