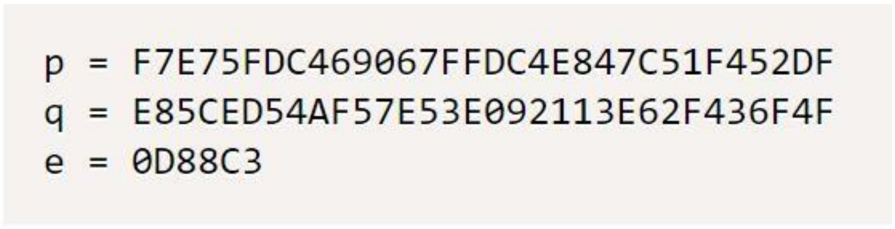
Justin Guerrero

Lab 11

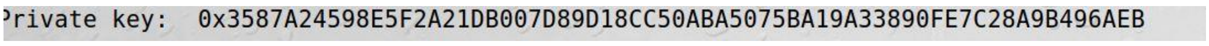
Task 1: Deriving a private key

In task one we are asked to derive a public key given p, q, and e provided by Travis. The values will then be put into a C script we are given and that is slightly modified.



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

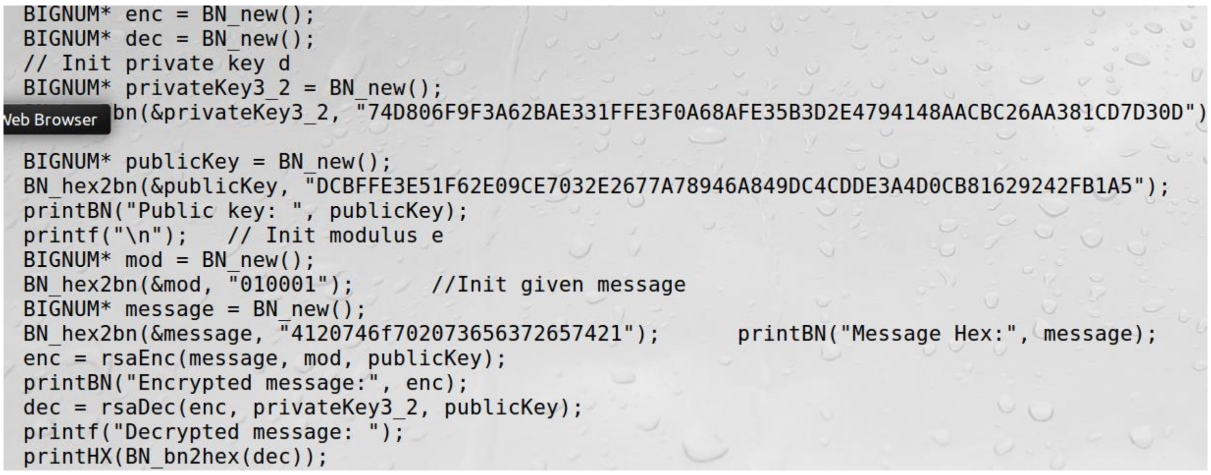
We are then given our private key



```
?rivate key:  0x3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
```

Task 2: Encrypting a message

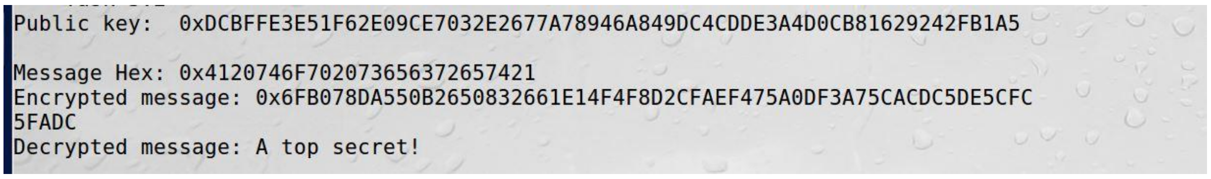
In this task we are given a secret message we want to decrypt. We start by converting the ASCII string to a hex string, and then the hex string to a BIGNUM using the BN_hex2bn() function. We are given the public key values from the lab and a private key to help verify the result.



```
 BIGNUM* enc = BN_new();
 BIGNUM* dec = BN_new();
 // Init private key d
 BIGNUM* privateKey3_2 = BN_new();
        bn(&privateKey3_2, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D")

 BIGNUM* publicKey = BN_new();
 BN_hex2bn(&publicKey, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
 printBN("Public key: ", publicKey);
 printf("\n");   // Init modulus e
 BIGNUM* mod = BN_new();
 BN_hex2bn(&mod, "010001");      //Init given message
 BIGNUM* message = BN_new();
 BN_hex2bn(&message, "4120746f702073656372657421");      printBN("Message Hex:", message);
 enc = rsaEnc(message, mod, publicKey);
 printBN("Encrypted message:", enc);
 dec = rsaDec(enc, privateKey3_2, publicKey);
 printf("Decrypted message: ");
 printHX(BN_bn2hex(dec));
```

We then can see our public key, the message hex, encrypted message, and the decrypted message also.

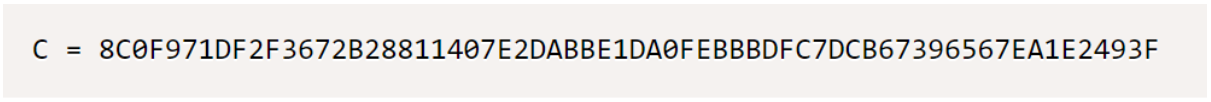


```
Public key:  0xDCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5

Message Hex: 0x4120746F702073656372657421
Encrypted message: 0x6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5CFC
5FADC
Decrypted message: A top secret!
```

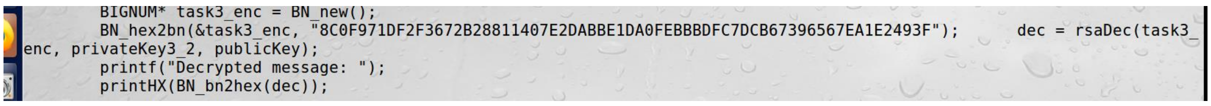
Task 3: Decrypting a message

This task we want to decrypt a ciphertext and convert is back to an ASCII string, the cipher text we are given is.



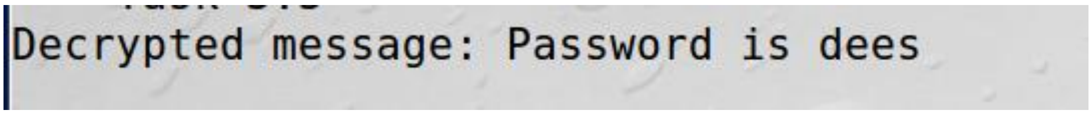
```
C = 8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F
```

Since we already have the cipher text all we need to do is convert is with BN_hex2bn() and then run the same rsn decoder we used in task 2.



```
        BIGNUM* task3_enc = BN_new();
        BN_hex2bn(&task3_enc, "8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F");       dec = rsaDec(task3_
enc, privateKey3_2, publicKey);
        printf("Decrypted message: ");
        printHX(BN_bn2hex(dec));
```

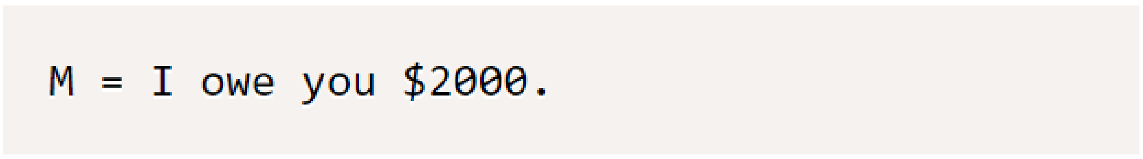
We run this code bit and receive the message:



```
Decrypted message: Password is dees
```

Task 4: Signing a message

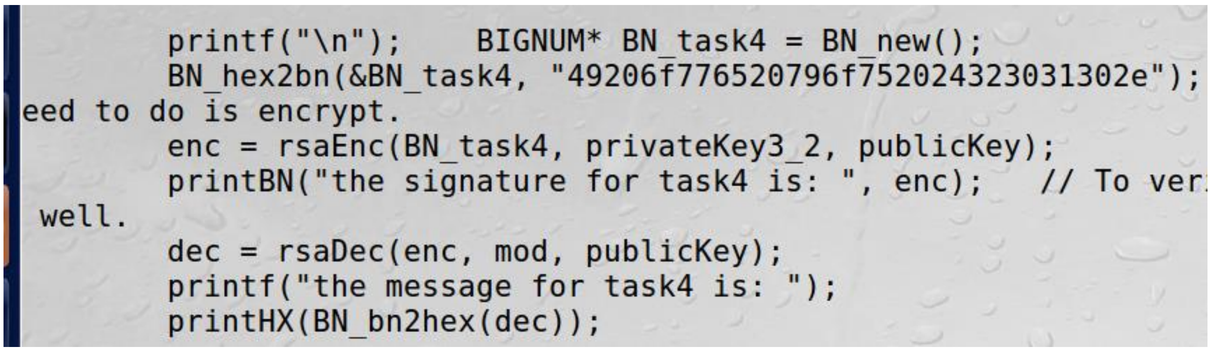
Here we want to generate a signature to the message given to us by Travis. The message is:



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

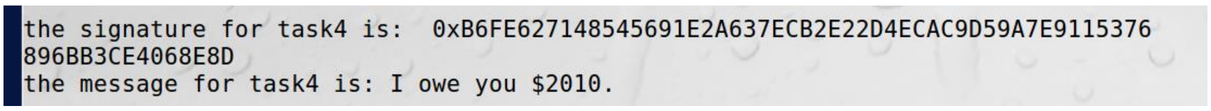
Please make a slight change to the message M, such a

We change the message into hex and run out bn function again. Since we already have the private key, all we need to do is encrypt the message. After we encrypt we want to decrypt the new value we created.



```
        printf("\n");    BIGNUM* BN_task4 = BN_new();
        BN_hex2bn(&BN_task4, "49206f776520796f752024323031302e");
eed to do is encrypt.
        enc = rsaEnc(BN_task4, privateKey3_2, publicKey);
        printBN("the signature for task4 is: ", enc);   // To ver
 well.
        dec = rsaDec(enc, mod, publicKey);
        printf("the message for task4 is: ");
        printHX(BN_bn2hex(dec));
```

After we run this code snip we receive the following:



```
the signature for task4 is:  0xB6FE627148545691E2A637ECB2E22D4ECAC9D59A7E9115376
896BB3CE4068E8D
the message for task4 is: I owe you $2010.
```

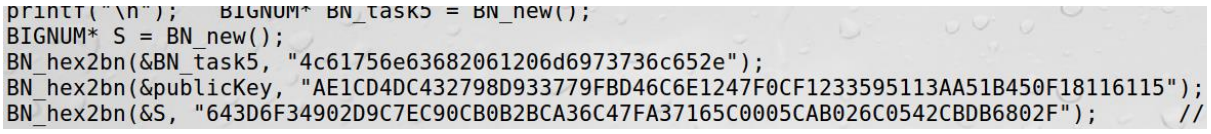
Task 5: verifying a signature

In this task we want to make sure the signatures are correct. If they are not correct we will not see the correct output. Lets take a look what we're working with.

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

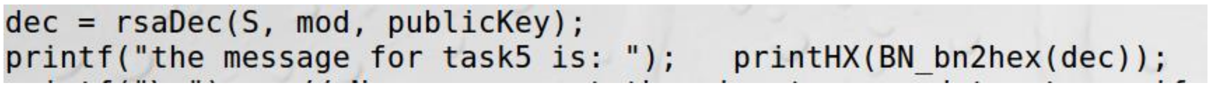
This should be handled, if not how do we know we need to drop a missile!?

First, like the rest of the tasks we run out new favorite command bn_hex2bn() with our values.



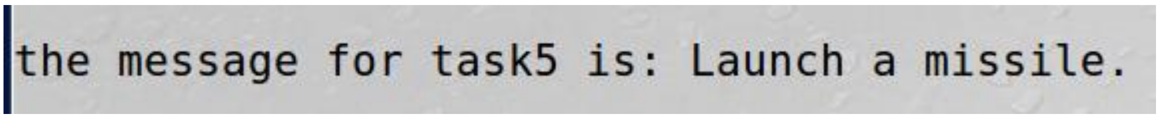
```
printf("\n");   BIGNUM* BN_task5 = BN_new();
BIGNUM* S = BN_new();
BN_hex2bn(&BN_task5, "4c61756e63682061206d6973736c652e");
BN_hex2bn(&publicKey, "AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");
BN_hex2bn(&S, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F");      //
```

Next we want to decrypt the message using our public key and print it out.



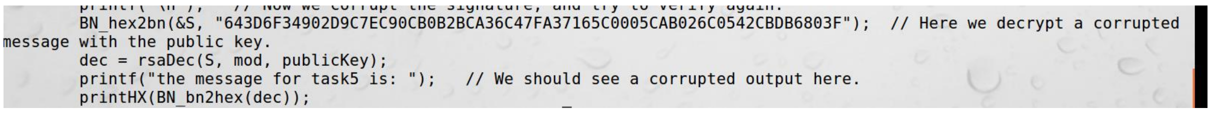
```
dec = rsaDec(S, mod, publicKey);
printf("the message for task5 is: ");   printHX(BN_bn2hex(dec));
```

And we receive the output:



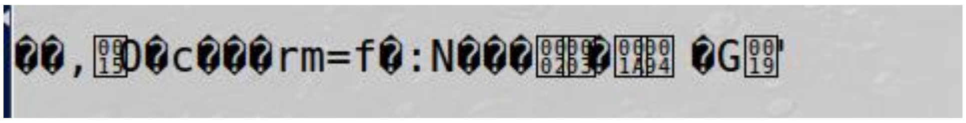
```
the message for task5 is: Launch a missile.
```

Great, lets send it. But wait, what happens if the signature is corrupted even just a little bit? Say if only one byte is changed, what happens?



```
        printf("\n");   // Now we corrupt the signature, and try to verify again.
        BN_hex2bn(&S, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F");  // Here we decrypt a corrupted
message with the public key.
        dec = rsaDec(S, mod, publicKey);
        printf("the message for task5 is: ");   // We should see a corrupted output here.
        printHX(BN_bn2hex(dec));
```

Note we've changed the signature just one byte from 2F to 3F at the end, this is the result:



```
��,D�c���rm=f�:N���� � �G'
```

Completely unreadable. Thus, we have verified the signature we had was indeed the right one and we should send the missile!