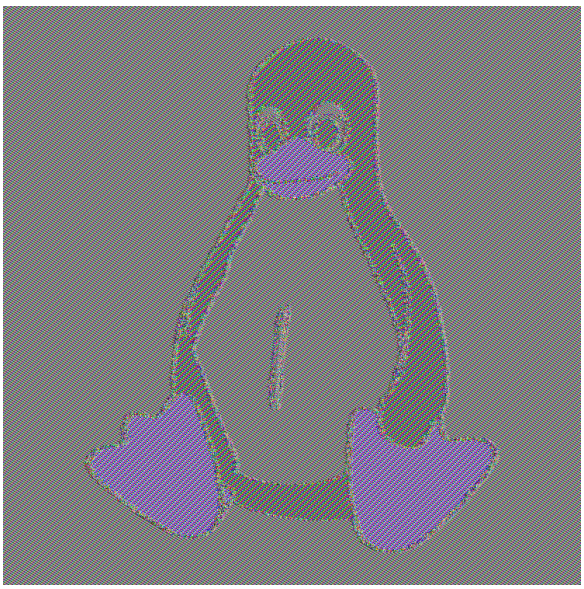
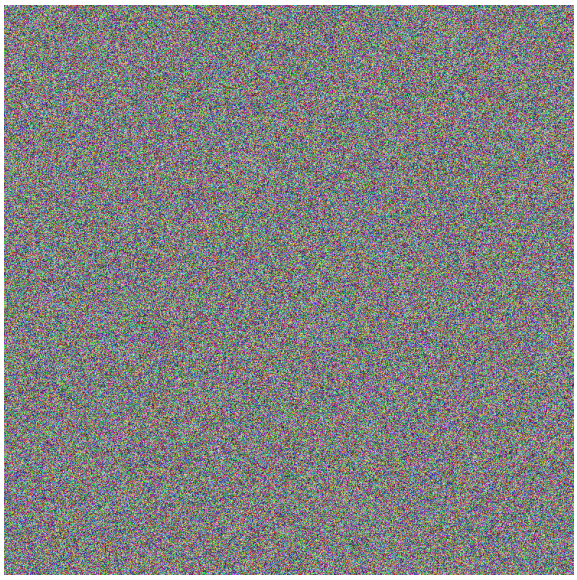
Diana Ramirez 88604827 Assignment 1

3. A. 1.ECB encrypted image



2.CBC Encrypted image



3. As shown in the pictures above, if the attacker has access to the encryption oracle he can more or less figure out the plaintext if ECB is used. If CBC is used, then the data is really scattered around, and the attacker cannot obtain any obvious (if any) info from the resulting encryption.

B.1. Since CBC uses XOR, even by guessing the encryption IVs for m_0 and m_1 the attacker has a chance of 50% of being right. Almost always if the first plaintext block is equal to the IV, the input to the block cipher is equal or "will result in the encryption of 0x0000000". For m_0 I would make sure that the first block is equal to first IV, and the first block of m_1 equal to the second IV. Therefore, if the inputs to the cyber block is 0x0000 ("since first plaintext block is equal to IV"), then we can match the corresponding IV ("defuse.ca").

C.1. OFB modes create keystream bits that are used for encryption of subsequent data blocks. If during this process a bit is lost, most of the plaintext or ciphertext can be recovered. It is possible to recover the block since this same block will be maintained even if the keystream bits passed to the next block are different.

4. A. i. $\boldsymbol{\phi(n)}$ = (p-1)(q-1) = (4)(12) = **48**

ii. **$C = m^e \bmod N$**

$2^3 \bmod 65 =$ **8** $57^3 \bmod 65 =$ **8**

iii.**$de = 1 \bmod \phi(n)$** => $d(3) = 1 \bmod 48$ => $d=(1 \bmod 48)/3$ => $d = 1/3$

$m = C^d \bmod N$ => $= 8^{1/3} \bmod 65$ => = **2**

I only found one of the values of m, even if both give the same ciphertext. I think this is what makes RSA decryption almost impossible to achieve, there could be many combinations.

B. N = 35 => N =(7,5) => $\phi(n)$ = 24 => d = (k$\phi(n)$ +1)/e = (k(24)+1)/5 = 25/5 =5

m = c^d mod N => 10^5 mod 35 => =**5**

5. A.1.

```
def part5A(pkfile,ctfile,possmess):
##convert messages using public key, and compare them

    recovered_public_key = RSA.importKey(open(pkfile, 'r').read())
    public_key = recovered_public_key


    filepossmess = open(possmess)
    possible_message = filepossmess.readlines()


    cyphertxtSent = (open(ctfile)).read()[1:-3]

    for ln in possible_message:
        ln = ln[:-1]
        message_int = tools.text_to_int(ln)
        ciphertext = public_key.encrypt(message_int, None)

        ciphertext = str(ciphertext[0])

        if ciphertext == cyphertxtSent:
            print "Message found: "+ ln
```

2. I could change the key every time I send a new message. The key would have to be random generated, since the textbookRSA implementation gives similar cyphers for certain plaintexts.

B.

```
def part5B(ctfile):
    cipher_mess = int((open(ctfile,'r')).read()[1:-3])
    #print(cipher_mess)
    plaintxt = tools.int_to_text(int(tools.find_root(cipher_mess,3)))
    print(plaintxt)
```

References

- http://www.crypto-it.net/eng/theory/modes-of-block-ciphers.html
- https://defuse.ca/cbcmodeiv.htm
- https://stackoverflow.com/questions/36834580/iv-must-be-16-bytes-long-error-in-aes-encryption