Cryptography

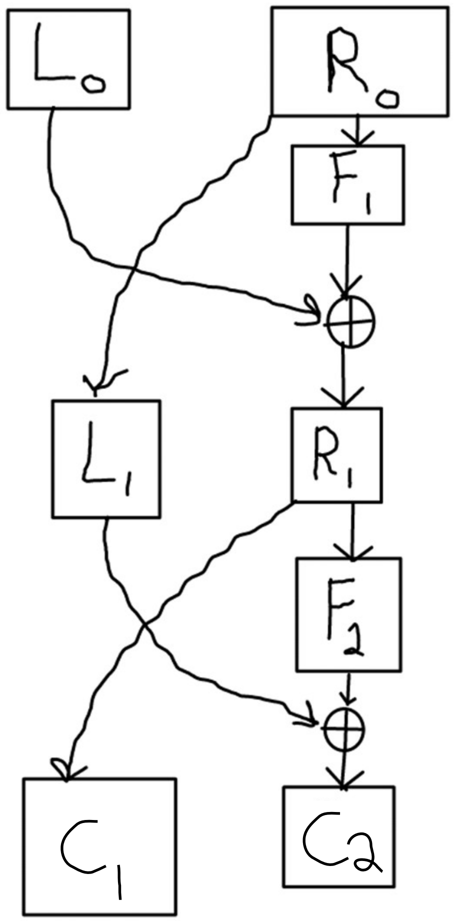
HW #3

Sam Quinn

CS427

02/1/2016

1. Attack the left half of the function since it is the weaker half.



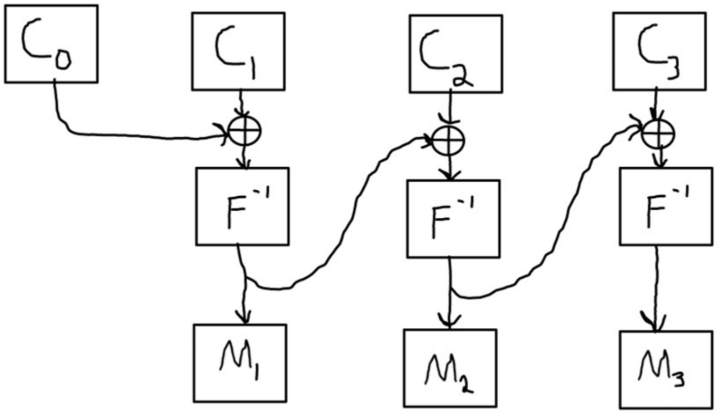
If we call the function on the left with $l_0 = 1111$ and anything you want for $r_0 = ????$. $l_0 \otimes r_0$ will create a value that will go into r_1 which is directly transferred to the left side c_1 without any modifications. Lets store $c_1 \rightarrow x_1$ and call the the same encryption function with $l_0 = 0000$ this time but with the same r_0 . This will produce a c_1 that is a exact complement of the x_1 . Because we are able to get a deterministic output for the cipher text this is **not** a secure PRP.

```
crypto_breaker() {
        L0 = 1111, R0 = 1010
        X1 = f(L0, R0)
        L0 = 0000
        X2 = f(L0, R0)

        Return X1 (xor) X2 ?= L0
}
```

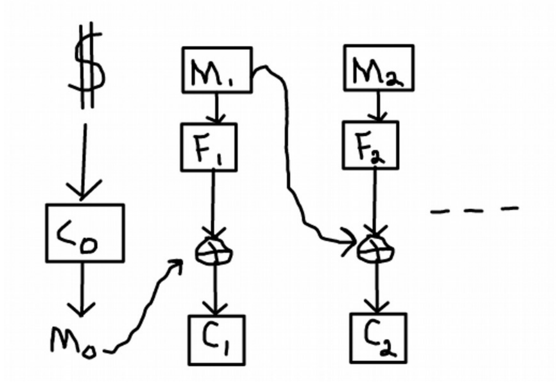
2.

a)



b) I was able to break this encryption with a choose plain text attack as described below:

```
crypto_breaker() {
   M1 ← 0000
   M2 ← 0000
   X||Y ← Enc(k, M1, M2)
   X'||Y' ← Enc(k, Y, M2)
   Return M1 ?= Y'
}
```



If we pass $m_1=0000, m_2=0000$ to the $Enc(k, m_1, m_2)$ function m_1 will xor the random deterministic value that comes out of f_2 and gets stored it into c_2. Now since we can choose our plain text to pass to the $Enc(k, m_1, m_2)$ function if we choose to pass c_2 as m_1 we will end up getting $f_2 \otimes f_2$, and as we know anything that xors itself returns all zeros. Because we can get deterministic cipher text out of this encryption algorithm, it is not CPA secure. The advantages are

$Attack = 1 \quad Rand = 1-(\frac{1}{2^{\lambda}})$

3.

```
crypto_breaker() {
   F0 = False
   M1 ← 1111
   M2 ← 000   // blen – 1
   While ( (F0 & F1) != True) {
         X ← Enc(k, M1, M2)
         If X[-(blen + 1)] == 0
            F0 ← True
            D0 ← x[length(x)]
         If X[-(blen + 1)] == 1
            F1 ← True
            D1 ← x[length(x)]
   }
   C ← Enc(k, M1, M2)
   If last bit in C == D1
         Return the C[-(blen+1)] bit is 1
   Else
         Return the C[-(blen+1)] bit is 0
}
```

This by far my most confusing distinguisher to date but I will try to explain why I believe that this will break CPA security. We start off by passing in a message m that is intentionally (blen -1) bits long to force a pad. We will continue to pass the message in until the last bit covers both cases 0 and 1. When ever we find either one of the values we store the value at -(blen + 1) which is the bit that is used to pad the plain text message. This allows us to create a correlation between the last bit and the -(blen + 1) bit. The reason that we have to continue to run the program for an undetermined amount of time is due to the fact that the random IV could continue to produce the same value for the last bit and we will need to find the other case. This security could have been fixed if they would have added an xor on the last path from m_l to F_k. The advantage for this distinguisher is $Attack=1$

$Rand=1-(\frac{2}{2^{\lambda}})$.

4.

a) If an adversary was granted permission to choose their own IV a new CPA security definition could be to enforce that the user will not use the same IV more than once. If the adversary could reuse the same IV they could continue to use the same IV or even an IV that would always print the plain text out, either way this would break the CPA$ security. The solution could be for the encryption algorithm to store the IV used permanently and be added to a blacklist. When a new IV is passed in the IV will be added to a blacklist then the IV will be used in the encryption algorithm. This would not reveal any other details about the encryption algorithm since it uses OTP security.

b) If the adversary could choose both the plain text and the IV the adversary could always chose the complement. Even for random IVs if the adversary could see the IV before hand could intentionally choose plain text for the encryption algorithm to print out anything they wanted. This would not be secure because if you continue to add the chosen IV as the complement of the message plain text you will continue to get deterministic output. The advantage is $Attack=1 \quad Rand=1-(\frac{1}{2^{\lambda}})$.

```
crypto_breaker() {
        IV ← {0,1}^lambda
        M = complement IV
        C ← Enc(k,IV,M1)
        Return C ?= {0}
}
```