Andrew Johnson

CS 427

HW4

1) (a)

```
Dec(k,(r,s))
   c = s ⊕ r
   d = F^-1(k,c)
   Return d ⊕ r
```

(b)

```
Adv(m)
   X → {0,1}^λ
   (r,s) = Enc(k,m)
   r* = x ⊕ r
   s* = x ⊕ r
   m* = Dec(k,(r*,s*))
   return m* ⊕ x == m
```

P(real) = 1
P(fake) = 0

2)

```
ADV()
   t = GETMAC(k,0^λ1^λ)    // F(k,0^λ)||F(k,1^λ)
   t' = t_left||t_left     // F(k,0^λ)|| F(k,0^λ)
   Return VER(0^λ0^λ,t')
```

P(real) = 1

P(fake) = 0

3)

```
ADV(m1,m2)   //m1,m2 = λ-bits
   t = MAC(k, m1)
   t' = MAC(k, m2 ⊕ t)
   Return VER(m1||m2, t')
```

P(real) = 1
P(fake) = 0

4) (a) The proof will break down when we try to factor out $L_{mac\text{-}real}$ because it needs to factor out k. If it does then k will become a private variable and ENC and DEC will throw an error since they would be trying to use a private variable.

(b)

```
ENC(k*,m):
   k_e := F(k*,0)
   k_m := F(k*,1)
   c ← E.Enc(k_e,m_L)
   t := M.MAC(k_m,c)
   return (c,t)

DEC(k*,m):
   k_e := F(k*,0)
   k_m := F(k*,1)
   if t ≠ M.MAC(k_m,c)
      return err
   return E.Dec(k_e,c)
```

≡

```
ENC(k*,m):
   k_e := e       // e = {0,1}^λ
   k_m := m'      //m' = {0,1}^λ
   c ← E.Enc(k_e,m_L)
   t := M.MAC(k_m,c)
   return (c,t)

DEC(k*,m):
   k_e := e       //same e as in ENC
   k_m := m'      //same m' as in ENC
   if t ≠ M.MAC(k_m,c)
      return err
   return E.Dec(k_e,c)
```

This is satisfied because F is a secure PRF.

Since k_e and k_m are both in ENC and DEC and are the same in both functions. They can be used as private local variables of the library, and notice k^* isn't used anymore. So we can take that out of the parameters if we want. Either way it brings us back to the original secure Encrypt-then-MAC construction, which we already proved. Thus the Modified Encrypt-then-MAC is CCA-Secure.