# CS 427, Assignment 5

Cody Malick
malickc@oregonstate.edu

February 23, 2017

## 1

The weakness in this altered CBC mode is that the IV is encrypted without being XORed with something else. This leaves us with an encrypted IV that can be used to tell whether or not another block of the encryption is empty. For example, we can look at $c_0$, and compare it to $c_1$. If $c_1$ is all zeroes, it'll be equal to the encrypted IV. This would reduce the complexity of the encryption by the size of the block. For example:

```
k := {0,1}^ℓ
ATTACK():
  // A string that begins with 0's in the first block size
  m_1 := 00..0 + {0,1}^(ℓ−λ)
  c_0, c_1, ..., c_ℓ := Enc(k, m_1)
  // Use the encrypted IV to gain information about the first block
  if c_0 == c_1:
    return true
  return false
```

The attacking function will return true with a probability 1. This shows that the function is not CPA secure, as we were able to glean information about the ciphertext.

## 2

### a

We can show that this scheme is secure through a simple hybrid proof. If $\Sigma$ is secure, then it returns a uniformly random distribution result, and is not susceptible to CPAs. Building of these assumptions, we can show a relatively simple hybrid proof. Here is the starting state:

```
Σ'.Enc(k, m):
  c_1 := Σ.Enc(k, m)
  c_2 := Σ.Enc(k, m)
  return(c_1, c_2)
```

The first step in our hybrid proof is to pull out both of our encryption calls into a function call $\mathcal{F}$:

```
Σ'.Enc(k, m):
  c_1 := F(k, m)
  c_2 := F(k, m)
  return (c_1, c_2)
```

$\diamond$

```
F(k, m):
  return Σ.Enc(k, m)
```

This doesn't change the outcome of the calling function, as the behavior inside of $\mathcal{F}$ is the same as it was before the abstraction. We can now safely replace the return value inside of $\mathcal{F}$ with a uniformly random distribution because it has been assumed that it is a CPA secure function:

$$\boxed{\begin{array}{l}\underline{\Sigma'.\mathsf{Enc}(k,m):}\\ \quad c_1 := \mathcal{F}(k,m)\\ \quad c_2 := \mathcal{F}(k,m)\\ \quad \text{return } (c_1,c_2)\end{array}} \diamond \boxed{\begin{array}{l}\underline{\mathcal{F}(k,m):}\\ \quad \text{return } x \leftarrow \{0,1\}^\lambda\end{array}}$$

Now, we can simply substitute the random value from $\mathcal{F}$ into its parent function without changing any behavior:

$$\boxed{\begin{array}{l}\underline{\Sigma'.\mathsf{Enc}(k,m):}\\ \quad c_1 \leftarrow \{0,1\}^\lambda\\ \quad c_2 \leftarrow \{0,1\}^\lambda\\ \quad \text{return } (c_1,c_2)\end{array}}$$

This function is CPA secure as there is no way for a chosen plaintext attack to glean any information. The resulting values are uniformly random, and inputting the same plaintext twice will not yield the same result. $\Sigma'$ is CPA secure.

### b

This function can be attacked with a chosen ciphertext attack, because it does not check a resulting ciphertext against a list of ciphertexts it has generated before decrypting. We can then take advantage of this fairly easily:

$$\boxed{\begin{array}{l}\underline{Attack():}\\ \quad m_1 := 000...00\\ \quad m_2 := 111...11\\ \quad c_1, c_2 := CHALLENGE(m_1||m_2)\\ \quad d_1, d_2 := CHALLENGE(m_2||m_1)\\ \quad \text{if } Dec(c_2, d_1)\text{: return true}\\ \quad \text{else: return false}\end{array}}$$

Because of the scheme, the decryption function will only return true when both halves of the ciphertext are equal. In this case, the probability of that will be 1. Which means we can find a relationship between the two ciphertexts. That means that it is not CCA secure.

## 3

The decryption algorithm would be:

$$\boxed{\begin{array}{l}\underline{Dec(r,k,x):}\\ \quad c := x \oplus r\\ \quad m := F^{-1}(k,c) \oplus r\\ \quad \text{return } m\end{array}}$$

This scheme is not CCA secure as we can glean some information about the decrypt function by simply xoring our input ciphertext with r before inputing it into the decryption function. The returned value, instead of m, will be $m \oplus r$, which gives us information about how the decryption function works.