COMP6453: Week 3 Answers

1 Linear Block Ciphers are not CCA Secure

For any block cipher, the fact that it is a nonlinear function is crucial to its security. To see this, suppose that we have a block cipher that encrypts 256-bit blocks of plaintext into 256-bit blocks of ciphertext. Assume the following property holds:

$$Enc(k, m_1 \oplus m_2) = Enc(k, m_1) \oplus Enc(k, m_2)$$

for all 256 bit messages m_1, m_2. Describe how, with 256 chosen ciphertexts, an adversary can decrypt any ciphertext without knowledge of the secret key k. (A "chosen ciphertext" means that an adversary has the ability to choose a ciphertext and then obtain its decryption. Here, you have 256 plaintext/ciphertext pairs to work with and you have the ability to choose the value of the ciphertexts.)

Answer: For $i = 0, ..., 255$, assume the adversary knows ciphertext c_i corresponds to message m_i where $c_i = (0, ..., 0, 1, 0, ..., 0)$. That is, c_i is the 256 bit string where the ith bit is a 1 and the corresponding m_i is some 256 bit message.
Let c be some ciphertext seen by the adversary. Then $c = \sum_{i \in I} c_i$, where $I \subset \{0, ..., 255\}$. This tells us that the corresponding message is $m = \sum_{i \in I} m_i$.

2 Compare AES vs. DES

For each of the following elements of DES, indicate the comparable element in AES or explain why it is not needed in AES.

1. XOR of sub-key material with the input to the f function
2. XOR of the f function output with the left half of the block
3. f function
4. permutation P
5. swapping of halves of the block

Answer:

- Asymmetric block ciphers include Data Encryption Standards (DES) and Advanced Encryption Standards.
- DES's plaintext is 64 bits long, while AES's plaintext might be 128, 192, or 256 bits long.
- AES relies on the substitution and permutation concept while DES relies on the Feistel cipher structure (16 Feistel rounds make up DES).

a) XOR of subkey material with the input to the f function

- In DES, perform XOR the subkey with the function's input throughout each of the 16 rounds.

- With AES, we follow the same procedure, performing an XOR operation between the subkey and the input in each of the 10 rounds (AES-128) before moving on to the next one.

b) XOR of the f function output with the left half of the block:

- In DES, the 32-bit to 48-bit expansion permutation is followed by an XOR operation on the left side of the block.
- Recall that, in the classic Feistel structure, half of the data block is used to modify the other half of the data block and then the halves are swapped. AES is not Feistel encryption, the XOR operation is not carried out to the left half block in AES. AES processes the entire data block as a single matrix during each round using substitutions and permutations.

c) f function

- In this context, the DES f function corresponds to E extension function, P permutation function, and S-boxes lookup tables (contains 64 entries).
- AES does not have f function but the round function has four different stages: Substitute bytes, ShiftRows, MixColumns, AddRoundKey.

d) permutation P:

- DES permutation function is a mixing permutation that maps 32-bit input to a 32-bit output by rearranging the bits of the input.
- AES has a fixed permutation (one-to-one function) on $\{0,1\}^{128}$ that does not depend on the key. The permutation S is specified in AES standard as a hard-coded table of 256 entries.

e) swapping of halves of the block

- In DES, the left and right blocks are switched after every single round.
- AES avoids this swapping by treating the entire block of plaintext as a single unit and performing operations on the entire block at once.

3 Cryptanalysis of the RC4 PRG

Recall that a PRG is considered secure if and only if given some i bits of PRG output, an "efficient" adversary cannot predict bit $i+1$ with nonnegligible probability. In this problem, we look at the RC4 PRG, which was previously used to encrypt (as a stream cipher) web traffic until its cryptanalysis.

The RC4 PRG works in the following way: we take a short seed s, start with an array S of n bytes. We have $S[0]=0, S[1]=1, ..., S[n-1]=n-1$. Our first step (we do not explain how this works) is to use the seed s to obtain a permutation S' of S. Now to generate our bitstream we apply the following algorithm:

```
$i = 0$
$j = 0$.
repeat until we get a long enough stream:
    $i \leftarrow (i+1) \pmod{n}$
    $j \leftarrow j + S'[i] \pmod{n}$
```

swap $S'[i], S'[j]$.

output $S'[S'[i] + S'[j] \pmod{n}]$.

Show that the second byte of the stream generated is equal to 0 with probability $\approx 2/n$. (*Hint: Consider the event where initially* $S'[2] = 0$ *and* $S'[1] \neq 2$).

Answer: Let P be the event that $S'[2] = 0$ and $S'[1] \neq 2$. Let z_2 be the event that the second byte of output is zero. Then we have

$$Pr[z_2] = Pr[S'[2] = 0|P] \cdot Pr[P] + Pr[S'[2] = 0|\neg P] \cdot Pr[\neg P].$$

A calculation tells us that $Pr[P] = \frac{(n-2)\cdot(n-2)!}{n!} \approx 1/n$. Moreover, if the even P happens, then $z_2 = 0$ with probability 1.

On the other hard, when the event $\neg P$ occurs, z_2 takes on any values in $0, ..., n-1$ at uniform random. Plugging everything into the equation above, we get an approximate probability of

$$1 \cdot \frac{1}{n} + \frac{1}{n} \cdot (1 - \frac{1}{n}) \approx \frac{2}{n}.$$