MATH 314 Spring 2018 - Class Notes

04/03/2018

Scribe: Jeff Henrichs

Summary:
Today we finished our discussion on 3DES, talked about the Modes of Operation, and we briefly went over Advanced Encryption Standard (AES) and Simplified AES (SAES).

Notes:

3DES

Encrypt three times with Data Encryption Standard (DES), this is resistant to a meet in the middle attack.

*In theory this how 3DES works:

$$C = E_{K_3}(E_{K_2}(E_{K_1}(P)))$$

$$D_{K_3}(C) = E_{K_2}(E_{K_1}(P))$$

Note that: E_{K_2} has 2^{211} possibilities, this is because each key

(K_1 and K_2) has a possibility of 2^{56}

$$D_{K_2}(D_{K_3}(C)) = E_{K_1}(P)$$

*This is how it is actually performed:

$$\text{3DES(P)} = (D_{K_2}(E_{K_1}(P))) = C$$

$$D_{K_2}(E_{K_1}(P)) = D_{K_1}(C)$$

$$E_{K_1}(P) = E_{K_2}(D_{K_1}(C))$$

$$P = D_{K_1}(E_{K_2}(D_{K_1}(C)))$$

3DES is used in practice today but it is recommended against.

3DES in practice uses only two keys, K_1 and K_2

Modes of Operations

How do we encrypt messages longer than the block size?

Idea 1: Electronic Codebook (ECB)

The idea of ECB is to break a long piece of plaintext into appropriate sized blocks of plaintext and process each block separately with the encryption function E_K.

The `plaintext` P is broken into smaller chunks $P = [P_1, P_2, ..., P_L]$

The `ciphertext` is $C = [C_1, C_2, ..., C_L]$

where $C_i = E_K(P_i)$ is the encryption of P_i using key K

There is a weakness in the ECB mode of operation.

Let's say Eve has been observing communication between Alice and Bob for a long period of time. If Eve has managed to acquire some of plaintext in correspondence to the ciphertext pieces that she has observed, she can build a codebook with which she can decipher future communication between Alice and Bob. Eve will never need the key,she can just look up ciphertext in her codebook and use the corresponding plaintext to decipher the message. This can be a serious problem since many real world messages consist of repeated fragments. Email is a prime example of that.

Another problem that arises in ECB mode occurs when Eve tries to modify the encrypted message being sent to Bob. She might be able to extract key portions of the message and use her codebook to construct a false ciphertext message that she can send to Bob.

Idea 2: Cipher Block Changing (CBC)
One method for overcoming the issues with ECB is to use chaining. Chaining is a feedback mechanism where the encryption of a block depends on the encryption of other blocks.

Alice picks a C_0 (this can be any ciphertext). She sends it to Bob in cleartext.

Encryption Process

$$C_i = E_K(P_i \oplus C_{i-1})$$

Decryption Process

$$P_i = D_K(C_i \oplus C_{i-1})$$

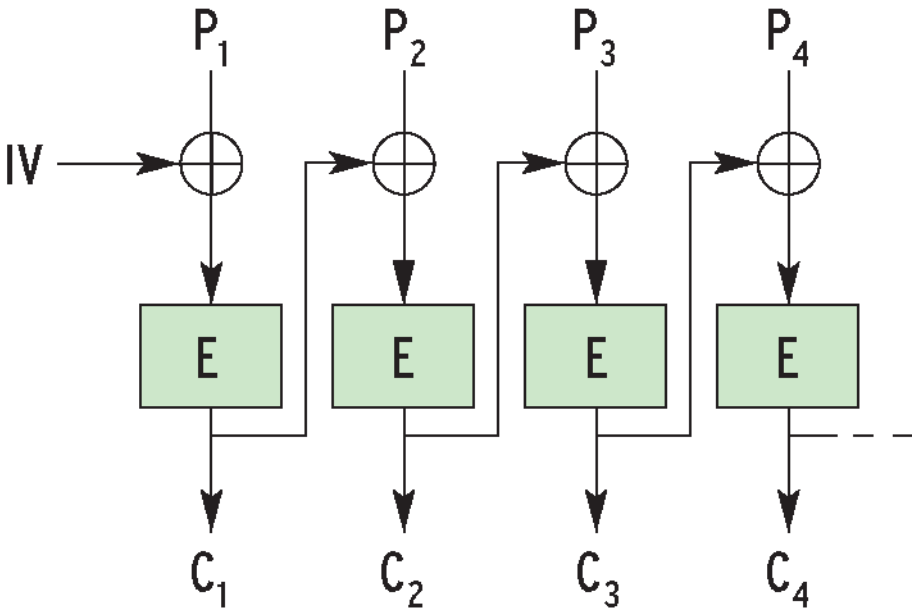

Figure 1: Cipher Block Chaining Mode

*Note that the initial value C_0 is often called the initialization vector (IV).

Idea 3: Cipher Feedback (CFB)
One of the problems with both CBC and ECB methods is that encryption and decryption cannot begin until a complete block of 64 bits of plaintext data is available.

In general, CFB operates in a k-bit mode, where each application produces k random bits for XORing with the plaintext.

We define head(A) as the leftmost k-bits of string A
We define tail(A) as the remaining bits of head(A).

Ex. String A = 1011 0010 1111, k = 8

$$\text{head(A)} = 1011$$

$$\text{tail(A)} = 0010\ 1111$$

First step is to break the plaintext is broken into smaller blocks usually at the size of head(A).

Afterwards the initial X_0 is chosen.

Encryption proceeds as follows:

$$O_i = head(E_K(X_i))$$

$$C_i = P_i \oplus O_i$$

$$X_{i+1} = tail(X_i) \| C_i$$

Decryption is performed like this:

$$P_i = C_i \oplus head(E_K(X_i)$$

$$Xi + 1 = tail(X_i) \| C_i$$

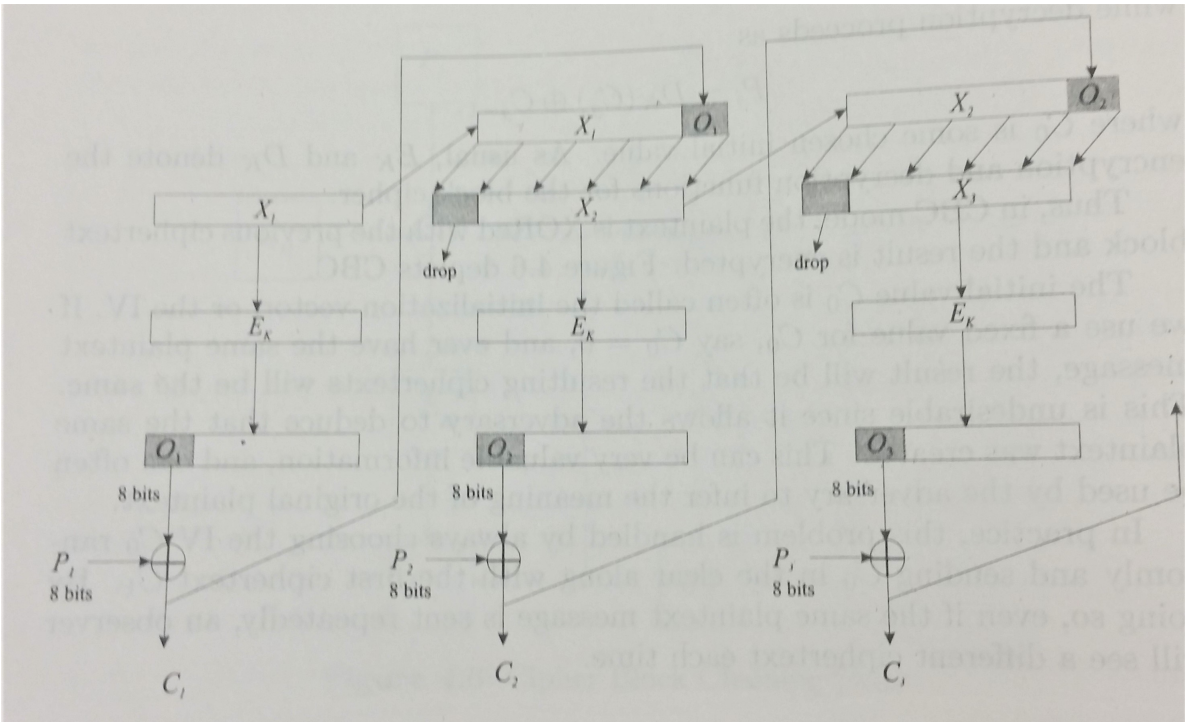

Figure 2: Cipher Feedback Mode

*Note that CFB is an example of a Stream Cipher

Idea 4: Output Feedback (OFB)
OFB is very similar to CFB. Much like CFB, OFB may work on chunks of different sizes and is also stream cipher.

We start with an initial value X_i, which has a length equal to the block length of the cipher.

So far, the operation is a lot like CFB.

Encryption:

$$O_i = head(E_K(X_i)$$

$$X_{i+1} = tail(X_i) \| O_i$$

$$C_i = P_i \oplus O_i$$

Decryption is very simple:

$$P_i = C_i \oplus O_i$$

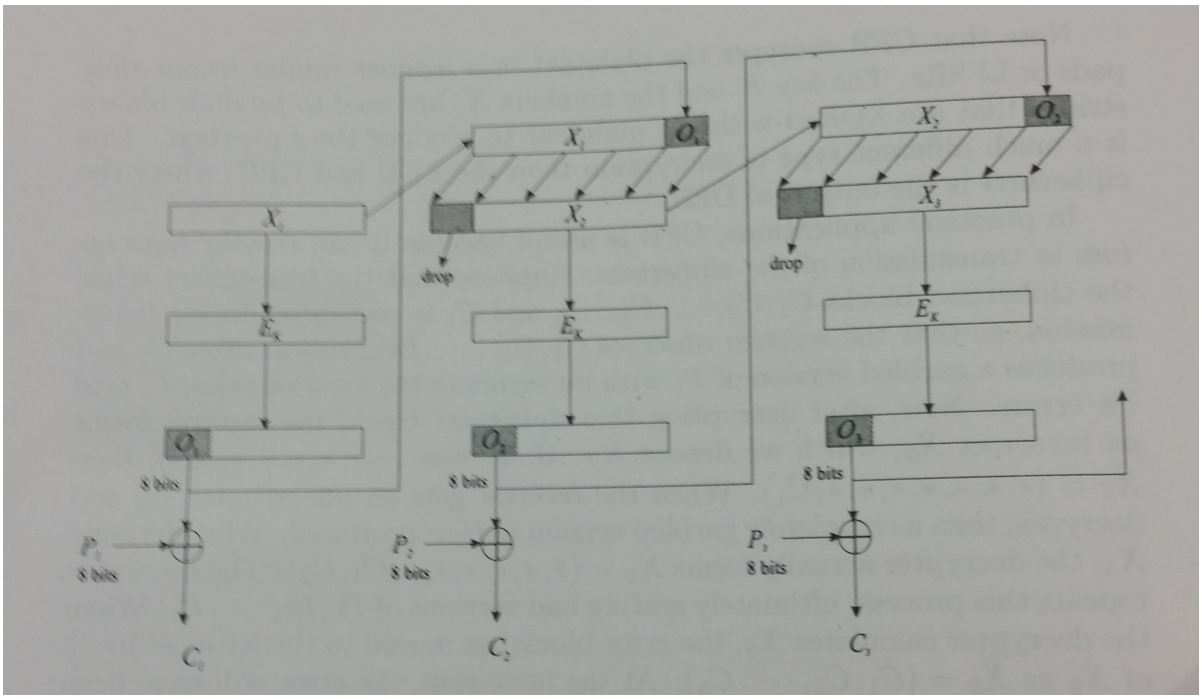

Figure 3: Output Feedback Mode

The main difference between OFC and CFB is that, CFB sends C_i to O_i and appends 0_i to X_i and OFB appends O_i to X_i and xors O_i with P_i. The `ciphertext` C_i in this case is left alone.

One advantage of OFB has over CBC and CFB is that, unlike CBC and CFB, OFB avoids error propagation. In layman's terms, OFB corrects itself.

Idea 5: Counter (CTR)

Counter builds upon the ideas that are used in the OFB. Just like OFB, CTR creates an output key stream that s XORed with chunks of plaintext to produce ciphertext.The main difference between CTR and OFB lies in the fact that the output stream O_i in CTR i not linked to previous output streams.

CTR starts with the plaintext broken into 8-bit pieces, $P = [P_0, P_1, ...]$

We begin with an initial value X_0, which has a length equal to the block length of the ciphertext,for example, 64 bits. X_0 is encrypted using key K to produce 64 bits of output, and the leftmost 8-bits of ciphertext C_0.

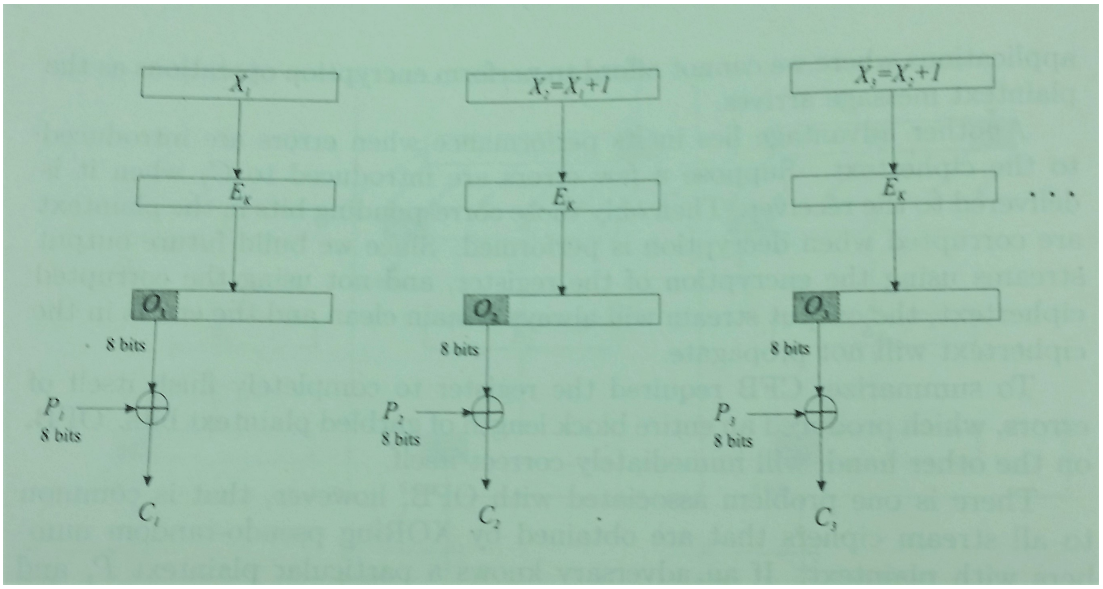

Figure 4: Counter Mode

Rather than to update the register X_i to contain the output of the block cipher, we simply take $X_i = X_{i-1} + 1$

The procedure for CTR is:

$$X_i = X_{i-1} + 1$$

$$O_i = head(E_K(X_i))$$

$$C_i = P_i \oplus O_i$$

The advantage of CTR is that CTR can output many chunks, O_i may be calculate in parallel. This makes CTR mode ideal for parallelizing.

Advanced Encryption Standard (AES)

Interesting fact: When you browse the internet, everything you sends as a user uses AES.

In 1997, the National Institute of Standards and Technology (NIST) put out a call for proposals for a new national cryptosystem.

Out of 5 finalists, Rijndael was chosen as the AES.

*Note that Rijndeal is pronounced as "Rain Doll"

AES is faster, more open, and more secure than DES.

AES is not a Fiestel Cipher.

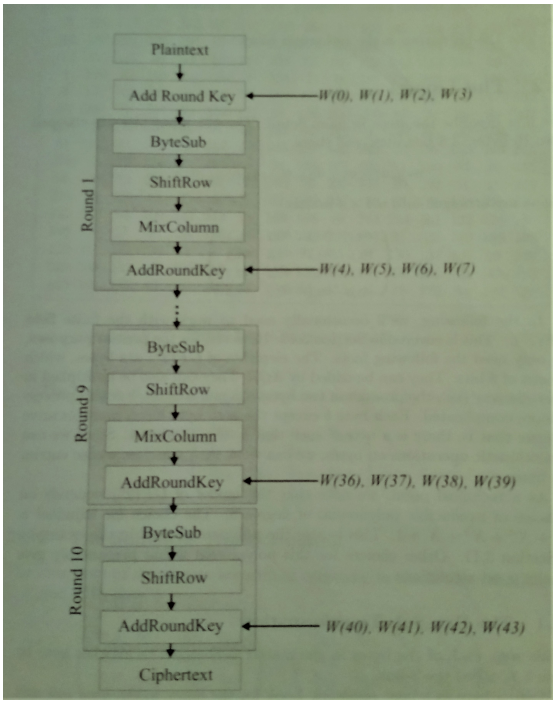

Figure 5: AES-Rijndael Algorithm

Simplified AES (SAES)

- Has 16-bit plaintexts

- 16-bit keys
- 2 rounds

SAES works very similar to AES, however it only as 2 rounds. SEAS Round 2 is works exactly like Round 10 of AES