Paxson
Spring 2011

CS 161
Computer Security

Discussion 7

March 16, 2011

Question 1 ***One-Time Pads*** **(10 min)**

Recall how a one-time pad works. Alice shares a stream of random bits with Bob, and she encrypts a message of length n for Bob by XORing the next n bits of this stream with the message. Bob decrypts by XORing the ciphertext with the same n bits from the stream of random bits.

(a) Does this scheme work if we replace XOR with OR? How about with AND?

Solution: No, it doesn't work with either OR or AND. First of all, correctness is broken; it is not true that $(m \vee k) \vee k = m$ for all choices of m and k (where m and k are each a single bit). Similarly, $(m \wedge k) \wedge k \neq m$ for all choices of m and k. This means that you can't actually decrypt an encrypted message.

Security is also broken. For OR, consider what an eavesdropper learns when she sees a 0 bit in the ciphertext. The only way this can happen is if both the key bit k and the message bit m are 0. For AND, when an eavesdropper sees a 1 bit in the ciphertext, she knows that both k and m are 1. Both OR and AND leak information.

(b) Suppose you want to encrypt a message $M \in \{0, 1, 2\}$ using a shared random key $K \in \{0, 1, 2\}$. Suppose you do this by representing K and M using two bits (00, 01, or 10), and then XORing the two representations. Does this scheme have the same security guarantees of the one-time pad? Explain.

Solution: No, this scheme does not have the security guarantees of a one-time pad. The table below lists the resulting encrypted messages using this scheme.

K	M	$E(K, M)$
00	00	00
01	00	01
10	00	10
00	01	01
01	01	00
10	01	11
00	10	10
01	10	11
10	10	00

We can see that some outcomes exclude certain inputs. For example, given $E(K, M) = 11$ an attacker knows that the sent message M is not 0.

(c) Give an alternate encryption algorithm for carrying out the above task that does provide strong security guarantees. NOTE: You must not change the message space $\{0, 1, 2\}$ or the key space $\{0, 1, 2\}$. Instead, we want you to design an encryption algorithm $E(\cdot, \cdot)$ so that $E(K, M)$ is a secure encryption of M, when K and M are distributed as above.

Solution: We wish to design a new encryption algorithm $E^*(\cdot, \cdot)$ that has the security guarantees of the one-time pad, i.e., *if an eavesdropper Eve observers the ciphertext, she learns* ***nothing*** *about the plaintext*. Put more formally, we require that given $E^*(K, M)$, an attacker should not gain any information about M. This property is satisfied for any $E^*(K, M)$ that is *uniform* on $\{0, 1, 2\}$, i.e., generates those values with equal probability, and with a relationship to M that is uniformly spread over the possible values of K.

One such algorithm is as follows:

$$E^*(K, M) = M + K \bmod 3.$$

The table below confirms that each outcome is equally likely.

K	M	$E^*(K, M)$
00	00	00
01	00	01
10	00	10
00	01	01
01	01	10
10	01	00
00	10	10
01	10	00
10	10	01

Question 2 ***PRNGs and Stream Ciphers*** **(5 min)**

(a) Pretend I have given you a pseudo-random number generator R. R is a function that takes a 128-bit seed s, an integer n, and an integer m, and outputs the n^{th} (inclusive) through m^{th} (exclusive) pseudo-random bits produced by the generator when it is seeded with seed s.

Use R to make a secure symmetric-key encryption scheme. That is, define the key generation algorithm, the encryption algorithm, and the decryption algorithm.

Solution:

- **Key generation.** Generate a random 128-bit key $K \in_R \{0,1\}^{128}$.
- **Encryption.** Let j be the latest index we have used from our PRNG. We start with $j := 0$ and maintain the state of j for subsequent encryptions. Let L be the number of bits in message M. Then,

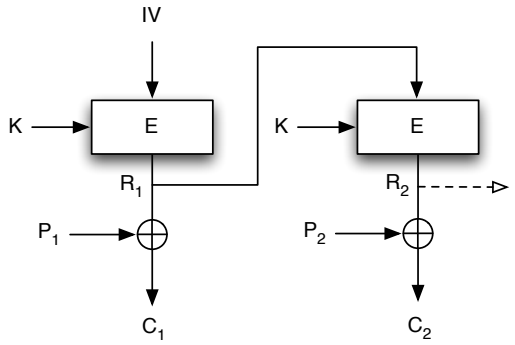
$$E(K, M) = R(K, j, j + L) \oplus M.$$

- **Decryption.** Define j and L as above. We have

$$D(K, C) = R(K, j, j + L) \oplus C.$$

(b) Explain how using a block cipher in counter (CTR) mode or output feedback (OFB) mode is similar to the scenario described above.

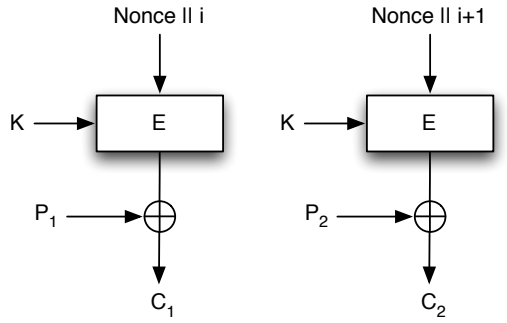
Solution: CTR mode and OFB mode are both stream cipher modes. They use the key to generate a pseudo-random stream of bits. This random stream is then XORed with the message to form the ciphertext. Let's first look at OFB:



OFB begins with encrypting the initialization vector (IV) with K to produce a random stream R_1. We then XOR this stream with the plaintext P_1 to produce the ciphertext C_1. We use the stream R_1 in the next round in the same role as IV played initially, this time to produce R_2. We can concisely summarize the scheme as follows:

$$\begin{aligned} R_0 &:= IV \\ R_i &:= E(K, R_{i-1}) \\ C_i &:= P_i \oplus R_i \\ P_i &:= C_i \oplus R_i \end{aligned}$$

CTR differs in two ways. First, there is no computational dependency between the rounds, which enables an efficient parallel computation. Second, the IV is replaced with a nonce and counter, as visualized below.



Nonce and counter are encrypted with key K to produce the random stream that for a given element of the plaintext P_i is XORed with P_i to produce the ciphertext C_i. In summary, CTR is defined as:

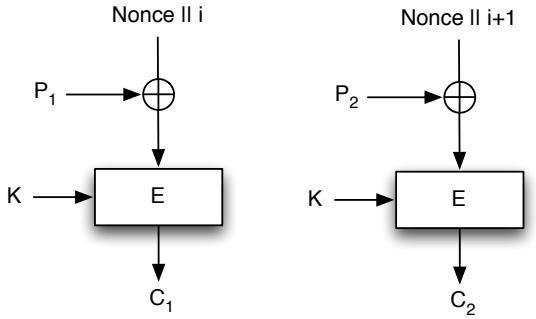
$$\begin{aligned} R_i &:= E(K, \text{Nonce}||i) \\ C_i &:= P_i \oplus R_i \\ P_i &:= C_i \oplus R_i \end{aligned}$$

where $||$ denotes concatenation.

Question 3 ***CTR Mode Variation*** **(5 min)**

In the CTR mode, the nonce and counter are encrypted first and then XORed with the plaintext. Now consider a variation where the plain text is XORed with the nonce and counter prior to encryption. What security properties does that mode have?

Solution: The proposed scheme would perform the XOR operation *before* applying the encryption function E, as shown in this illustration:



Because the nonce is in the clear and the counter can be deduced by Eve, the XOR operation with the plaintext is essentially useless—Eve can perform it as well. The result is a reduction of CTR mode to a scheme with similar security properties as ECB.

In particular, consider the case when two instances of plaintext, P_2 and P_3, differ only in the bottom bit. For the first, the input to the block cipher is $P_2 \oplus \text{Nonce} \,||\, 2$, while for the second it is $P_3 \oplus \text{Nonce} \,||\, 3$.

The only difference in the righthand operands to the $\oplus$ operator is that for the first, the bottom bit is not set, while for the second, it is. But that difference is exactly balanced by the same difference in P_2 vs. P_3. Thus, *in both cases the result of the XOR is the same.* Accordingly, if Eve observes $C_2 = C_3$, she knows that the plaintext differs exactly in just the bottom bit—leaking information about its structure analogous to how ECB would leak information if instead $P_2 = P_3$.