Cryptography Part 1 **Fall 2017**

Lecture 2: Block Cipher

Lecturer: Dan Boneh *Scribe: Lily Li*

2.1 What is a Block Cipher?

As an overview: block ciphers take an n bit plane text, run it through an encryption and a decryption algorithm with a specified key which is k bits long. The output is an n bit cipher text. The actual cipher is built through iteration. The key is first expanded into n keys $k_1, ..., k_n$ then repleted apply a *round* function on the message with each k_i.

Definition 2.1 *A **pseudo random function** (PRF) defined over (K, X, Y), which is the key, input, and output spaces respectively is a function $F : K \times X \to Y$ such that there exists an efficient algorithm to evaluate $F(k, x)$.*

*A specific type of PRF is the **pseudo random permutation** (PRP), defined over (K, X), $E : K \times X \to X$ such that*

1. *There exists and efficient* deterministic *algorithm to evaluate $E(k, x)$.*
2. *$E(k, \cdot)$ is injective.*
3. *There exists an efficient inversion algorithm $E(k, y)$.*

Next we want to define what it means for a PRF to be secure. Let $F : K \times X \to Y$ be a PRF. Let $\mathsf{Funs}[X, Y]$ be the set of all functions from X to Y and $S_F = \{F(k, \cdot) : k \in K\}$ be a subset of $\mathsf{Funs}[X, Y]$.

Intuitively, a PRF is **secure** if a random function in $\mathsf{Funs}[X, Y]$ is indistinguishable from a pseudo-random function in S_F.

2.2 Data Encryption System (DES)

The key component of DES is the Feistel Network, described in the following.

2.2.1 Feistel Network

We are given functions $f_1, ..., f_d : \{0,1\}^n \to \{0,1\}^n$ and we wish to build an invertible function $F : \{0,1\}^{2n} \to \{0,1\}^{2n}$. With two blocks of n bits as input, denoted R_0 and L_0, the output, R_d and L_d are defined recursively: $L_i = R_{i-1}$ and $R_i = L_{i-1} \oplus f_i(R_{i-1})$.

Notices that this process is completely invertible since $R_{i-1} = L_i$ and $L_{i-1} = R_i \oplus f_i(L_i)$.

Theorem 2.2 *(Luby-Rackoff '85) If $f : K \times \{0,1\}^n \to \{0,1\}^n$ is a secure PRF then the three-round Feistel network $F : K^3 \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ is a secure PRP. (Note here the key applied to each of the three functions is independently generated.)*

2.2.2 How DES Works

The length of a DES key is 56 bits and the length of the message and cipher text is 64 bits long. A the heart of DES is a 16 round Feistel network. Functions $f_1, ..., f_{16} : \{0,1\}^{32} \to \{0,1\}^{32}$ are all of the form $f_i(x) = F(k_i, x)$ where the round keys k_i are derived from the initial input key k.

The steps of DES are these:

1. Input 64 bit.
2. Perform an initial permutation (for standards only, not needed for security).
3. Run the 16 round Feistel network.
4. Perform an inverse permutation (also only needed for standard definition purposes).
5. Output 64 bits.

The function F is defined as follows: on inputs x (of 32-bits) and k_i (of 48-bits), F expands x to 48-bits by replicating and shuffling some bits around. Then F takes the xor of the modified x and k_i. After breaking the result into eight blocks of six bits, F passes each block of six bits into an S_i lookup table (for $1 \leq i \leq 8$) to transform it into four bit. Finally F collects up the resulting 32 bits and, after a final permutation, returns the output.

A key property of these lookup tables is that S_i is not linear! Otherwise DES is linear and vulnerable.

2.2.3 Brute Force Attacks on DES

The goal now is to consider the attacker's perspective and find the key k given a view input, output pairs.

Lemma 2.3 *Suppose that DES is an **ideal cipher** — that is each key in $\{0,1\}^{56}$ produces a random invertible function $\pi : \{0,1\}^{64} \to \{0,1\}^{64}$. Then for all inputs-outputs m, c respectively there is at most one key k such that $c = \mathsf{DES}(k, m)$ with* very *high probability.*

Proof: Consider

$$\Pr[\exists k' \neq k : c = \mathsf{DES}(k, m) = \mathsf{DES}(k', m)]$$

which is the probability that two functions, produced from two different keys, will map the same message to the same cipher text. By union bound we have that

$$\Pr[\exists k' \neq k : c = \mathsf{DES}(k, m) = \mathsf{DES}(k', m)] \leq \sum_{k \in \{0,1\}^{64}} \Pr[\mathsf{DES}(k, m) = \mathsf{DES}(k', m)]$$

The right-hand-side works out to $2^{56}/2^{64} = 2^8 = 1/256$ which is approximately 99.5%. ■

2.2.4 Triple DES

In light of these vulnerabilities, people tried to salvage DES by increasing the size of the key space. In particular, taking the encryption algorithm $E : K \times M \to M$ (or any other block cipher), $3D : K^3 \times M \to M$ is defined as:

$$3E((k_1, k_2, k_3), m) = E(k_1, D(k_2, E(k_3, m))).$$

Triple DES has keys three times as long as those of DES but is also three times slower.

Note: if you only double the length of the key, then you are at risk of a *meet in the middle attack*.

2.2.5 DESX

Another way to make DES more secure is as follows. Again, take the block cipher $E : K \times \{0,1\}^n \rightarrow \{0,1\}^n$. Define $EX((k_1, k_2, k_3), m) = k_1 \oplus E(k_2, m \oplus k_3)))$. Note that the length of the key for $DESX$ is $64 + 56 + 64$ bits long and this modification does not suffer from a longer encryption and decryption time.

Even though it can be shown that there are no exhaustive search attacks against this construction, this method is vulnerable to more subtle attacks and is not entirely safe.

2.3 Attacks on Block Ciphers

2.3.1 Attacks on Implementation

Those who wish to obtain the secret key can apply advanced techniques if they have their hands on the encryption-decryption hardware. Suppose the data is stored on some smart card or computer. By measuring the time to encrypt/decrypt or the amount of power used for encryption and decryption the attacker can learn quite a bit about the key itself. Further, over clocking the hardware and causing a data fault can leak data about the secret key.

2.4 Advanced Encryption Standard (AES)

Unlike DES and 3DES (above), AES uses a substitution-permutation network. The encoding happens in several rounds each of which involves an XOR with the round key, an application of a set of substitution block then a permutation of the results. Each of these steps must be invertible, in particular the substitution blocks must be invertible, in-order for decryption to be possible.

Specifically, AES takes keys of three possible sizes (128, 192, and 265 bits) and input blocks of 128 bits.

1. Write the 128-bit block (16 bytes) into a four by four matrix of bytes.
2. XOR the input matrix with round key k_0.
3. Perform the three functions: ByteSub, ShiftRow, and MixCol. Note that these three functions must all be invertible.
4. Repeat steps two and three a total of ten times. In the last step do NOT apply MixCol.
5. XOR the resulting matrix with the last round key k_{10} and output the result.

The eleven round keys are obtained by key expanding the original 16-byte key and are also arranged into the four by four byte matrices.

The following is a brief description of the three functions.

ByteSub: suppose the current byte in cell i, j is $a_{i,j}$. The output of ByteSub is the substitution box S applied to $a_{i,j}$. S is essentially a lookup table — encoding the multiplicative of each number over $GF(2^8)$. This function together with a invertible affine transformation is known to have desirable properties such as: very non-linear, no fix points.

ShiftRow: rows are cyclically shifted by an offset. For row i, all elements are shifted i spaces to the left (rows are zero indexed). This steps mixes the columns so that they are not encrypted independently.

MixCol: all elements of the same column are passed through a invertible linear transformation. This step also helps with diffusion.

2.5 Building Block Ciphers From PRGs

2.5.1 PRF from a PRG

Suppose we have a secure PRG $G : K \to K^2$. Then it is possible to define a one bit PRF by using the function $F : K \times \{0,1\} \to K$ such that $F(k, x) = G(k)[x]$ where $G(k) = G(k)[0] \parallel G(k)[1]$. The function mapping the key to the output is

Theorem 2.4 *If G is a secure PRG then F is a secure PRF.*

Proof: Since G is a secure PRG, it is indistinguishable from random. Further both the left and right halves of the output of G are indistinguishable from random. Thus all functions $F(k, \cdot)$ "look" random. ■

Note that we can iterate this idea to build a larger PRG, and thus a larger PRF. First apply G to the input k to produce a pseudo-random key of twice the length. Denote this as $G(k) = G(k)[0] \parallel G(k)[1]$. Next apply G to each part separately. By combining everything we get a key which is four times the size of the original and looks indistinguishable from random. We can apply this as many times as we want to build a PRG $G' : K \to K^{2^{128}}$ then use G' to build $F' : K \times \mathbf{x} \in \{0,1\}^{128} \to K$ which uses each bit of $\mathbf{x}$ to indicate the left or right block to take at each step.

Unfortunately this is construction is not used in practice because it is quite slow. With the above PRF, we can uses the Luby-Rackoff construction to build a secure PRP.

2.5.2 Secure PRFs

We will define a Secure PRF in the following. First let us recall these next definitions. If $F : K \times X \to Y$ is a PRF then $\mathsf{Funs}[X, Y]$: is the set of all functions form X to Y and $S_F = \{F(k, \cdot) : k \in K\}$ is a subset of $\mathsf{Funs}[X, Y]$.

Our definition will again use the challenger-adversary model similar to what we saw when discussing security of stream ciphers.

There are two experiments depending on $b \in \{0,1\}$ defined as $EXP(b)$. The challenger takes as input b. If $b = 0$ then the challenger choses a key $k \in K$ then choses a PRF $f \leftarrow F(k, \cdot)$. Otherwise, if $b = 1$ then the challenger take f to be a truly random function from $\mathsf{Funs}[X, Y]$.

Then, for q rounds, the adversary can choose q inputs $x_1, ..., x_q$ and obtain the outputs $f(x_1), ..., f(x_q)$. The adversary then outputs $b' \in \{0,1\}$ which is its guess for the value of b.

Security for PRPs is almost exactly the same except that in experiment $b = 1$, the challenger choses f to be a random element in $\mathsf{Perms}[X]$, which is the set of random permutations.

3DES and AES are two PRPs that are believed to be secure.

It also turns out that any secure PRP is also a secure PRF, if $|X|$ is sufficiently large.

Lemma 2.5 *Let E be a PRP over (K, X). Then for any q-query adversary A:*

$$\mathsf{Adv}_{PRF}[A, E] - \mathsf{Adv}_{PRP}[A, E] < \frac{q^2}{2|X|}$$

2.5.3 One Time Key (CBC)

(Here the adversary can only see one cipher text.)

Before we learn how to securely use PRPs to encode data, let us see what is the wrong way to do this.

Definition 2.6 *Then **Electronic Code Book** (ECB) encoding works as follows. Break the plain text into blocks of the same size as the code blocks. Encrypt the blocks separately.*

This strategy is horribly broken and leaks data since blocks of plain text with the same data get map to the same cipher text. Do not use this method!

One secure construction is the deterministic counter mode from a PRF F. Here we again break the message m into blocks $m[0], ..., m[L]$ and this time we XOR each block $m[i]$ with PRP output $F(k, i)$.

To see that this construction is secure, consider our experiments. In $b = 0$ the adversary gets the encryption of $m_0 \oplus F(k, 0) \parallel \cdots \parallel F(k, L)$. This is indistinguishable from $m_0 \oplus f(0) \parallel \cdots \parallel f(L)$ where f is a truly random function since F is a secure PRP. This is futher indistinguishable from $m_1 \oplus f(0) \parallel \cdots \parallel f(L)$ for any message m_1 since f is random, so the adversary cannot tell apart experiment $b = 0$ and $b = 1$.

2.5.4 Many Time Key (CTR)

Since the same key is used to encrypt several message, we must update our definition of security. Notably the adversary can launch chosen-plaintext attacks (CPA).

For the security definition we will again use the two experiment construction. The challenger takes as input $b \in \{0, 1\}$. The adversary sends the challenger q rounds of message pairs $(m_{1,0}, m_{1,1})$,$(m_{2,0}, m_{2,1})$, ..., $(m_{q,0}, m_{q,1})$ where for each i, $|m_{i,0}| = |m_{i,1}|$. For round i the challenger return the cipher text $c_i = E(k, m_{i,b})$. Then the adversary outputs the guess for the index of the encrypted message, b'.

Can you see how the chosen plain text attack can be carried out under this paradigm?

Again, our definition of security is as follows.

Definition 2.7 *A cipher $\mathbb{E} = (E, D)$ defined over (K, M, C) is semantically secure under CPA if for all* efficient A

$$\mathsf{Adv}_{CPA}[A, \mathbb{E}] = |\Pr[EXP(0) = 1] - \Pr[EXP(1) = 1]|$$

is negligible.

Consider why our previous encryption schemes (one-time-pad, deterministic counter mode) are not secure under CPA. Note in particular that the same message always gets encrypted to the same cipher text under the same key. Use this to create an experiment which breaks security.

There are two ways to remedy this problem. Randomized encryption and ...

2.5.4.1 Randomized Encryption

The encryption algorithm $E(k, m)$ is changed to a randomized algorithm! For any message m_0 there could be a set of cipher texts that it gets mapped to. If this set is sufficiently large, the risk that multiple instances of the same message gets mapped to the same cipher text is small. Note that the set of cipher texts associated with different messages must be disjoint or the decoder will have issues decoding the message.

The problem here is that the ciphertext needs to be longer than the plaintext to encode the randomness. This is a problem when the messages themselves are quite short.

2.5.4.2 Nonce-based Encryption

The encryption algorithm is now $E(k, m, n) = c$ where n is a nonce (similarly the decryption algorithm $D(k, c, n) = m$ also takes the nonce as an input). The nonce is some value that changes from message to message such that the pair (k, n) is never repeated for the lifetime of the key. Notably, the nonce need *NOT* be random or private.

Popular choices include: counters and random nonce. If the nonce is implicit, the size of the ciphertext need not be longer than the plaintext.

To define CPA security properly we need to allow the adversary to be able to choose the nonce — in practice the adversary may choose to attack only when the nonce is at a certain value. But the adversary still needs to ensure that all nonces are distinct.

Finally we look at two encryption schemes which are chosen plaintext secure.

2.5.5 Cipher Block Chaining (CBC) with Random IV

CBC works as follows: let (E, D) be a PRP. $E_{CPB}(k, m)$ choses a random IV (initialization vector) in the input space. For the first block of the message $m[0]$, input $IV \oplus m[0]$ into the encryption algorithm to get the encryption of the first block. Suppose $c[0]$ is the output of $E(k, IV \oplus m[0])$. Input $c[0] \oplus m[1]$ into the encryption algorithm to obtain the encryption of the second block. Continue XOR-ing the cipher-text of the previous block with with current plain-text block until everything is encoded. The entire cipher-text is the IV concatenated with all the cipher-text blocks. Note that the decryption is quite similar to encryption.

Theorem 2.8 *CBC Theorem: if E is a secure PRP over (K, X) then E_{CBC} is semantically secure under a chosen plaintext attack (CPA) over $(K, X^L, XL + 1)$ where $L > 0$ is the length of the message.*

For a q-query adversary A attacking E_{CBC} there exists a PRP adversary B such that

$$\mathsf{Adv}_{CPA}[A, E_{CBC}] \leq 2 \cdot \mathsf{Adv}_{PRP}[B, E] + \frac{2q^2L^2}{|X|}.$$

Note here that the first part of the RHS is negligible by definition, but in order for the RHS to be entirely negligible we need $|X| \gg q^2L^2$.

Observe that it is crucial that the adversary cannot predict the IV. Otherwise there exists a sequence of two message pairs that allows the adversary to distinguish between $m_{i,0}$ and $m_{i,1}$.

Alternative to the random CBC is the nonce-based CBC where instead of an IV we chain with a unique nonce. Here the key is (k, k_1) where the nonce is encrypted with k_1 and the actual message is encrypted with k.

If the last message block is shorter than the cipher block then we need to add padding. Suppose i more bytes were needed to complete the block. Add i bytes each written with the number i. After decryption, simply read the last byte and remove that many bytes.

2.5.6 Random Counter Mode

Let F be a secure PRF where $F : K \times \{0,1\}^n \to \{0,1\}^n$. Chose a random IV and XOR message block $m[i]$ with $F(k, IV + i)$. Send the IV along with the cipher-text. Observe that this process works in parallel.

Similar to the above we can define security for counter mode as follows:

Theorem 2.9 *Let F be a secure PRF over (K, X, X). Then E_{CTR} is semantically secure under CPA over (K, X^L, X^{L+1}).*

In particular, for a q-query adversary A attacking E_{CTR} there exists a PRF adversary B such that

$$\mathsf{Adv}_{CPA}[A, E_{CTR}] \leq 2 \cdot \mathsf{Adv}_{PRF}[B, F] + \frac{2q^2 L}{|X|}.$$

Note, the bound is better than the one for cipher block chaining.