Lecture 12: 02/28/02

1 The Luby-Rackoff Construction (continued)

In the last lecture, we discussed the Luby-Rackoff construction, which allows us to build pseudorandom permutations from $2n$ bits to $2n$ bits out of pseudorandom functions. Recall that the Luby-Rackoff construction can be denoted $\Psi[f_1, f_2, f_3, f_4]$, where the functions f_i are the pseudorandom functions used in the construction. Below, we'll use H to indicate $\Psi[R_1, R_2, R_3, R_4]$, where the functions R_i are truly random. We'll need the following lemma (proved in the last lecture):

Lemma 1
If A is any adversary (i.e., with unbounded time) making q queries to its oracle, H is the Luby-Rackoff construction based on truly random functions, and π is a truly random permutation, then

$$|\Pr[A^{H,H^{-1}} = 1] - \Pr[A^{\pi,\pi^{-1}} = 1]| \leq \frac{2q^2}{2^n}.$$

That is, we provided a bound on the ability of an adversary to distinguish a truly random permutation from the Luby-Rackoff construction (when based on truly random functions). To complete our discussion of the Luby-Rackoff construction, we simply need to show this result still holds when using *pseudorandom* functions in place of truly random functions.

Theorem 2
If $F : K \times \{0,1\}^n \to \{0,1\}^n$ is a (t, q, ϵ)-secure pseudorandom function, then $\Psi[F_{k_1}, \ldots, F_{k_4}]$ (where the k_i are chosen independently and uniformly at random) is a $(t - O(q), q, \frac{2q^2}{2^n} + 4\epsilon)$-secure pseudorandom permutation.

Proof: First, note that Luby-Rackoff never uses the keys k_i directly; rather, it simply uses F_{k_i} as black boxes.

To prove this theorem, we need to bound the difference in probability between the case where A is given π and the case where A is given $\Psi[F_{k_1}, \ldots, F_{k_1}]$. To this end, we'll use a hybrid argument. First, observe that the difference in probability when using π and $\Psi[R_1, R_2, R_3, R_4]$ (where R_i are truly random) is $\leq \frac{2q^2}{2^n}$, as shown in the lemma above.

Now, consider the difference between $\Psi[R_1, R_2, R_3, R_4]$ and $\Psi[F_{k_1}, R_2, R_3, R_4]$. The difference in probability here is at most ϵ. To see why this claim is true, assume that there exists an algorithm A that can distinguish these permutations with probability $> \epsilon$. Then you can construct an algorithm that distinguishes F_{k_1} from R_1 with probability $> \epsilon$ by simulating R_2, R_3, and R_4, building permutations according to the Luby-Rackoff construction, and then running A on these permutations. This would imply that our pseudorandom function F is not (t, q, ϵ)-secure. Note that the attack on F that we've constructed requires

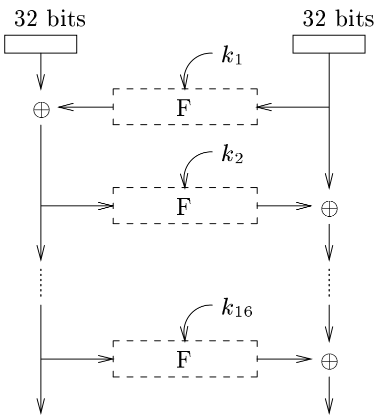

Figure 1: The DES block cipher.

that we simulate $O(q)$ queries to the oracle, which means that A must run in $t - O(q)$ time (the constant here should be reasonably small).

The same line of reasoning can be applied to the next three arguments to Ψ (e.g., compare $\Psi[F_{k_1}, R_2, R_3, R_4]$ and $\Psi[F_{k_1}, F_{k_2}, R_3, R_4]$). Overall, we have four hybrids as follows:

$$\begin{array}{cl}
\pi & \\
 & \leq \frac{2q^2}{2^n} \text{ apart} \\
\Psi[R_1, R_2, R_3, R_4] & \\
 & \leq \epsilon \text{ apart} \\
\Psi[F_{k_1}, R_2, R_3, R_4] & \\
 & \leq \epsilon \text{ apart} \\
\Psi[F_{k_1}, F_{k_2}, R_3, R_4] & \\
 & \leq \epsilon \text{ apart} \\
\Psi[F_{k_1}, F_{k_2}, F_{k_3}, R_4] & \\
 & \leq \epsilon \text{ apart} \\
\Psi[F_{k_1}, F_{k_2}, F_{k_3}, F_{k_4}] &
\end{array}$$

Applying the triangle inequality, we see that the probability difference between using π and $\Psi[F_{k_1}, \dots, F_{k_1}]$ is $\leq \frac{2q^2}{2^n} + 4\epsilon$. $\square$

Note that this proof depends upon the fact that the keys k_i are chosen independently and that the pseudorandom functions are used as black boxes within the Luby-Rackoff construction.

2 DES

DES (illustrated in Figure 1) is a block cipher often used in modern cryptographic applications. It is quite similar to the Luby-Rackoff construction in design. Since DES encrypts 64-bit blocks, it begins by splitting the block into two 32-bit halves. In each of sixteen rounds, one half is passed through the function F and then XOR'ed with the other half as shown in the diagram above. Each application of the function F takes a different 48-bit key, though the sixteen 48-bit keys are actually created by expanding a single 56-bit key.

The function F (Figure 2) takes the place of the pseudorandom functions in the Luby-Rackoff construction. First, the 32 bits of input are expanded to 48 bits by copying some of the inputs. Next, the 48 bits are XOR'ed with the key. Then these values are fed into

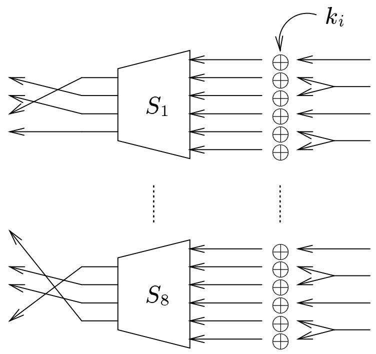

Figure 2: The DES F function.

one of eight S_i functions, each of which takes a 6-bit input to a 4-bit output. Finally, a permutation is applied to the outputs of the eight S_i functions. Overall, $F = P \circ S \circ \oplus_{k_i} \circ E$, where E is expansion, S represents the S_i functions, and P is a permutation.

The S functions are arbitrary functions that are designed to avoid structure and linearity, though of course they are not perfect (e.g. if the outside bits are fixed, the S function is a permutation of the inner 4 bits). The details of the S function aren't really important for the purposes of this class.

So how does DES compare to the theoretical Luby-Rackoff construction? It has four times as many rounds, but the keys k_i are not independent, and F is clearly not a very good pseudorandom function (changing one bit of input affects the inputs to only one or two S_i functions, which affects at most 8 output bits).

If we apply the Luby-Rackoff probability bound to DES, we can observe that the probability that an adversary can break the cipher approaches 1 as q^2 approaches 2^n, where $n = 32$. Thus, the Luby-Rackoff analysis suggests that we wouldn't want to trust DES against any adversary with 2^{16} oracle queries. Nevertheless, DES is conjectured to be more secure than this. An open problem is to improve the analysis of the Luby-Rackoff construction to account for the benefit of additional rounds; some progress has already been made with six rounds instead of four.

In general, the answer to any question about why DES is constructed the way it is (e.g., "Why 16 rounds?" and "Why are the F functions designed the way they are?") is that simpler versions have been broken. Despite being a rather ad-hoc approach to security, though, DES has remained unbroken for the last 20 years.

One common variant of DES is 3-DES ("Triple DES"), in which DES is simply repeated three times with three different keys.

3 Block Cipher Applications

We've discussed the design of the DES block cipher. Assuming this cipher is secure, we'll discuss methods for encrypting a message of arbitrary length. In all of these methods, we'll divide our input message into 64-bit blocks, padding the last block as necessary.

Our first attempt is the ECB ("Electronic Codebook") method, illustrated in Figure 3. In this method, the same key is used to encrypt each block of the message independently.

Our second attempt is the CBC ("Cipher Block Chaining") method, seen in Figure 4. In this method, we XOR the ciphertext of the last block with the plaintext of the current block before encrypting. The message is also transmitted with an initialization vector ("IV") that is XOR'ed against the first block of the message. The IV can simply be a counter that is incremented for each message, but we'll also consider for the sake of argument the case where the IV is always zero (i.e., there is no IV).

A third alternative is the CTR method, illustrated in Figure 5. In this method, we encrypt each number in a sequence starting at the IV and XOR this result with the message blocks. We continue incrementing the counter for the next message. (Note that we might be in trouble if the counter wraps, but we're assuming that that won't be a problem for this application.)

So, how secure are these approaches? And more importantly, what's an appropriate notion of security in the symmetric key case?

First, note that none of these methods gives away the key itself (assuming the block cipher is secure), so simply avoiding key recovery is not an appropriate notion of security.

Next, consider ECB. Clearly, this is a poor encryption scheme, since it doesn't even satisfy our earlier message indistinguishability criterion. Consider two message blocks A and B. If our two messages are AA and AB, then it will be immediately apparent which one has been encrypted; if the ciphertext consists of two identical blocks, it's clearly the encryption of AA, and otherwise it's the encryption of AB.

Now, consider CBC. As far as Prof. Wagner knows, this method is secure in the message indistinguishability sense, even if the IV is always zero. But clearly we have a problem if the IV is always zero, because then messages with identical plaintext prefixes have identical ciphertext prefixes. It turns out that our notion of message indistinguishability is inadequate in the symmetric key case because it doesn't allow the attacker to encrypt *other* messages; we got this for free when doing public key encryption (since the public key is, well, public), but in the symmetric key case, we're going to need to modify our notion of security.

DEFINITION 1 (LEFT-OR-RIGHT SECURITY) *Let $\S_i(x_0, x_1) = x_i$. E is (t, q, μ, ϵ)-secure in the left-or-right sense against CPA if for all A running in time t and making q queries of total length $\leq \mu$,*

$$|\Pr[A^{E_k \circ \S_0} = 1] - \Pr[A^{E_k \circ \S_1} = 1]| \leq \epsilon.$$

Note that the oracle provided to the adversary A chooses its left argument *always* or chooses its right argument *always*. Given (M_i, M_i'), it will return $E_k(M_i)$ always or $E_k(M_i')$ always. The only restriction on the messages that can be passed to the oracle is that they must be of equal length; that is, when asking about (M_i, M_i'), we require that $|M_i| = |M_i'|$.

A few additional notes:

1. The adversary *can* query the oracle on the same messages twice; any encryption scheme worth its salt will need to use randomization or maintain state to avoid such attacks.

2. The adversary *can* query the oracle on (M, M) (i.e., using the same message as both arguments), which accounts for faulty encryption schemes that, for example, reveal information about the key if queried on a specific message.

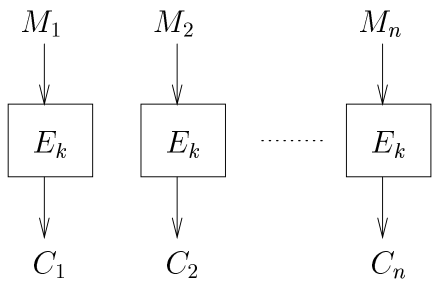

Figure 3: The ECB method.

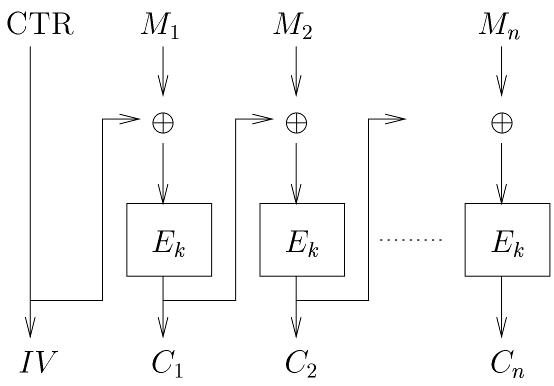

Figure 4: The CBC method.

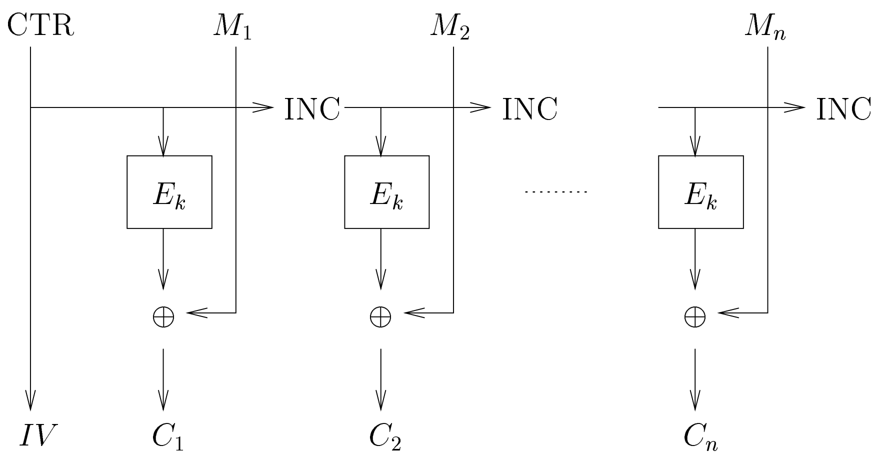

Figure 5: The CTR method.

3. The adversary *can* gain information based on the length of the *encrypted* messages. For example, if an encryption scheme compresses the plaintext before encrypting, then an adversary could distinguish the encryption of zero from the encryption of a random message by examining the length of the resulting ciphertext, assuming that the compression method compresses the zero message to a smaller size than the random message.

The ECB method is clearly insecure in the left-or-right sense, by the same reasoning provided above. CBC without a counter is also insecure by the following argument: consider one query to (R_1, AA) and another query to (R_2, AB) for random two-block messages R_1 and R_2; if the first block of the ciphertext matches, then $\S_i = \S_1$. (Note that this example demonstrates that the left-or-right notion of security compensates for the previously noted shortcomings of the message indistinguishability criterion.) Furthermore, even CBC with a counter is insecure. CTR, however, is secure in the left-or-right sense. These last two results will be discussed in the next lecture.