YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

CPSC 467b: Cryptography and Computer Security

Week 3 (rev. 2)

Professor M. J. Fischer *January 25 & 27, 2005*

Lecture Notes, Week 3

1 Stream ciphers from CFB and OFB modes

CFB and OFB block chaining modes (see Lecture Notes, Week 2) can be naturally extended to stream ciphers on units smaller than full blocks. The idea is to use a shift register R to accumulate the feedback bits from previous stages of encryption so that the full-sized blocks needed by the block chaining method are available. R is initialized to some public initialization vector.

Assume for sake of discussion a 64-bit block size for the underlying block cipher and a character size of s-bits. (Think of $s = 8$.) Let $B = \{0, 1\}$. We define two operations: $\sigma : B^{64} \rightarrow B^s$ and $\mu : B^{64} \times B^s \rightarrow B^{64}$, where $\sigma(x)$ is the leftmost s bits of x, and $\mu(x, c)$ is the rightmost 64 bits of the bit string xc.

The extended version of CFB and OFB are very similar. Both compute a *byte key* k_i and use it to encrypt message byte m_i with a simple XOR cipher. That is, $c_i = m_i \oplus k_i$. In both modes, k_i can be computed knowing only the ciphertext and master key, so Bob computes k_i and then decrypts by computing $m_i = c_i \oplus k_i$. Finally, both modes compute $k_i = \sigma(E_k(R_i))$ where R_i is the new contents of the shift register at stage i. The two modes differ in how they update the shift register. In extended CFB mode, $R_i = \mu(R_{i-1}, c_{i-1})$. In extended OFB mode, $R_i = \mu(R_{i-1}, k_{i-1})$. Thus, CFB updates R using the previous ciphertext byte, whereas OFB updates it using the previous byte key.

The differences between the two modes seem minor, but they have profound implications on the resulting cryptosystem. In CFB mode, the loss of ciphertext byte c_i will cause m_i and several succeeding message bytes to become undecipherable. At first sight it might seem that all future message bytes would be lost, but if one looks carefully at the shift register updating algorithm, one sees that $R_j = c_{j-8}c_{j-7} \ldots c_{j-2}c_{j-1}$ (in our special case of $s = 8$), so it depends on only the last eight ciphertext bytes. Hence, Bob will be able to recover plaintext bytes beginning with m_{i+8} after the loss of c_i. In OFB mode, R_i depends only on i and the master key k (and the initialization vector IV), so loss of a ciphertext byte causes loss of only the corresponding plaintext byte.

The downside of OFB is the same as for the one-time pad and other simple XOR ciphers, namely, if two message streams are encrypted using the same master key, then the XOR of their encryptions is the same as the XOR of the plaintexts. This allows Eve to recover potentially useful information about the plaintexts and renders the method vulnerable to a known plaintext attack. CFB does not suffer from this problem since different messages lead to different ciphertexts and hence different key streams. However, even CFB mode has the undesirable property that the key streams will be the same up to and including the first byte in which the two message streams differ. This will enable Eve to determine the length of the common prefix of the two message streams and also to determine the XOR of the first bytes at which they differ.

One way around this problem in both ciphers is to use a different initialization vector for each message. The IV is sent to B in the clear, along with the ciphertext. $R = R_0$ is initialized to IV, then $k_0 = \sigma(E_k(R_0))$ is computed, and then normal encryption proceeds.

2 Entropy

I have been talking loosely about the "information content" of a message and the fact that normal English text is quite "redundant". Information theory, pioneered by Claude Shannon half a century ago, provides an elegant and rigorous framework for studying these notions. I present just a bit of it here to give you a flavor of how one might proceed.

Redundancy in language refers to the fact that most combinations of letters do not form a meaningful English-language message. Meaning of course exists at many different levels. "QXUUUOV FMMZ" is not meaningful because the individual "words" are non-sense. "FUZZY WILL ARE" contains three valid words, but they are combined in an ungrammatical and non-sensible order. "PURPLE COWS SURF MODEMS" is grammatically correct but lacks semantic meaning.

In our abstract theory, we assume some classification of the entire message space $\mathcal{M}$ into a meaningful subset $\mathcal{M}'$ and a meaningless complement $\mathcal{M} - \mathcal{M}'$, and we assume Alice sends only meaningful messages.

Consider the usual case where Σ is a finite alphabet, and $\mathcal{M} = \Sigma^*$ consists of all strings over Σ. For a set of strings S, let S_n be the subset of strings in S of length n. We define the "number of bits" of information in an arbitrary member of S_n to be $B(S_n) = \log_2(|S_n|)$. This makes sense, for if one imagines listing the elements of S_n in a table, the length of the table is $|S_n|$, and integers of length $\lceil \log_2(|S_n|) \rceil$ are adequate to index the table. Let $b(S, n) = B(S_n)/n$, the average bits per character for a string in S_n. Clearly, $b(\Sigma^*, n) = \log_2 s$, where $s = |\Sigma|$. The ratio $b(S, n)/b(\Sigma^*, n) = b(S, n)/(\log_2 s)$ compares the amount of information per character for strings in S_n to the amount of information per character for arbitrary strings. This ratio is 1 when $S_n = \Sigma^n$ and is 0 when $|S_n| = 1$.[1]

The above argues that there is a coding scheme for S_n such that the average length representation is $\approx \log_2(|S_n|)$. However, one does not always want to assume that all words are equally likely. When different words of S_n have different probabilities of occurring, the expected length representation is minimized by a Huffman code and can be much less than the average encoding length in the unweighted case.

The notion of *entropy* extends the informal definition of $B(S_n)$ to this case. We imagine an idealized encoding scheme in which each word x is represented by $-\log_2(p_x)$ bits, where p_x is its probability of occurring.[2] Then the expected length of an encoding of a random word in S_n is the weighted average, where we weight the length of each representation x by its probability p_x. Formally, if X is a random variable ranging over S_n, we define its entropy $H(X)$ to be

$$H(X) = \sum_{x \in S_n} -p_x \log_2(p_x)$$

Note that $H(X) = B(S_n)$ in the special case that all elements are equally likely. That is, if $p_x = 1/|S_n|$ for all $x \in S_n$, we have

$$\begin{aligned} H(X) &= \sum_{x \in S_n} -\frac{1}{|S_n|} \log_2 \left(\frac{1}{|S_n|} \right) \\ &= -\log_2 \left(\frac{1}{|S_n|} \right) \\ &= \log_2(|S_n|) \\ &= B(S_n) \end{aligned}$$

[1] It is undefined when $S_n = \emptyset$.

[2] This assumption is approximately true for Huffman encodings and becomes exactly true in a "limiting" sense that is beyond the scope of these notes.

In a similar way, we extend the definition of $b(S,n)$, the "average" bits per character, by letting $h(X,n) = H(X)/n$. We now define the *redundancy* of X to be $\rho_n = 1 - h(X)/(\log_2 s)$. Thus, the redundancy is 0 when all strings of S_n are equally likely and 1 when one string x occurs with probability $p_x = 1$ and all other strings have probability 0.

Now, suppose $\mathcal{M}'$ is the set of meaningful English language messages, however one wants to define that, and to keep this discussion simple, we assume that all meaningful messages are equally likely. It is reasonable to assume in our model that $b(\mathcal{M}', n)$ approaches a limit b as $n \to \infty$. The reason for this is that $|\mathcal{M}'_{n+m}| \approx |\mathcal{M}'_n| \cdot |\mathcal{M}'_m|$ when n and m are both large. Hence, $b(\mathcal{M}', n) = \log_2(|\mathcal{M}'|)/n \approx b$. Thus, we will assume that $|\mathcal{M}'_n| = 2^{bn}$ and the redundancy is $\rho_n = 1 - b/(\log_2 s)$.

Suppose Eve intercepts the ciphertext $c = E_k(m)$ for a message m of length n, where $k \in \mathcal{K}$ is a randomly selected key. The preimage C of c in $\mathcal{M}$ under the various choices of k has size at most $|\mathcal{K}|$, and possibly less, since we in general make no assumption that c always decrypts to different messages under different keys. Obviously, $m \in C$. If it happens that $C \cap \mathcal{M}'$ is the singleton set $\{m\}$, then Eve knows that m is the correct plaintext decryption of c. The probability of this occurring depends on both the length of the plaintext and also on the particular encryption function being used.

Two extreme cases are worth mentioning. Suppose the encryption function is "well matched" to the meaningful messages in the sense that $E_k(m)$ is always in some subset $\mathcal{C}'$ of $\mathcal{C}$ whenever $m \in \mathcal{M}'$, and $E_k(m)$ is in $\mathcal{C} - \mathcal{C}'$ whenever $m \in \mathcal{M} - \mathcal{M}'$. Then the inverse image of every $c \in \mathcal{C}'$ is contained in $\mathcal{M}'$, so Eve is never able to uniquely identify m from c, no matter how much time she spends (unless the inverse image happens to be a singleton set, which is obviously very insecure).

To the contrary, one could imagine an encryption function such that the inverse image of every $c \in \mathcal{C}$ contains either zero or one meaningful message. In such a case, Eve can always find m from c, given enough time.

In practice, it is often assumed that the encryption function is not at all correlated with the notion of meaningful messages. This means that the family of encryption functions $\{E_k\}_{k \in \mathcal{K}}$ behaves like a randomly-chosen such family. In that case, Eve has a certain probability of unique decryption, which goes to 1 as n gets large.

We can approximate this probability by a simple calculation. Let $m_0 \in \mathcal{M}'_n$, $k_0 \in \mathcal{K}$, and let $c_0 = E_{k_0}(m_0)$. We wish to estimate

$$p_n = \text{prob}[\ \{m \in \mathcal{M}'_n \mid E_k(m) = c_0 \text{ for some } k \in \mathcal{K}\} = \{m_0\}\ ]\ .$$

This is the probability that the only meaningful message in the preimage of c_0 is the original message m_0. For this to be the case, it must be that every other key $k \neq k_0$ decrypts c_0 to a message in $\mathcal{M}_n - \mathcal{M}'_n$. Assuming $|\mathcal{M}_n| = |\mathcal{C}_n|$ and each decryption function is a random permutation, the probability that one key causes c_0 to decrypt to a meaningless message is

$$q_n = \frac{|\mathcal{M}_n - \mathcal{M}'_n|}{|\mathcal{M}_n|} = 1 - \frac{|\mathcal{M}'_n|}{|\mathcal{M}_n|}\ .$$

But under our assumptions, $|\mathcal{M}'_n| = 2^{bn}$ and $|\mathcal{M}_n| = s^n$, so $|\mathcal{M}'_n|/|\mathcal{M}_n| = 2^{bn}/s^n$. Since $b = (1-\rho_n)\log_2 s$, it follows that

$$q_n = 1 - \frac{2^{bn}}{s^n} = 1 - \frac{s^{(1-\rho_n)n}}{s^n} = 1 - (s^{-\rho_n})^n\ .$$

Now, the probability that all $t = |\mathcal{K}| - 1$ keys other than k_0 cause c_0 to decrypt to a meaningless message is

$$p_n = (q_n)^t = (1 - s^{-n \cdot \rho_n})^t$$

By the binomial theorem, p_n is approximately $1 - ts^{-n \cdot \rho_n}$ when $s^{-n \cdot \rho_n}$ is small. Since t, s, and ρ_n are all constants independent of n, we see that $s^{-n \cdot \rho_n} \to 0$ as $n \to \infty$. Hence, $p_n \to 1$ as $n \to \infty$, so large messages can be uniquely decrypted with probability approaching 1.

3 Data Encryption Standard (DES)

The Data Encryption Standard is a block cipher that operates on 64-bit blocks and uses a 56-bit key. It was the standard algorithm for data encryption for over 20 years until it became widely acknowledged that the key length was too short and it was subject to brute force attack. (The new standard used the Rijndael algorithm and is called AES.)

3.1 Feistel Networks

DES is based on a *Feistel network*. This is a general method for building an invertible function from any function f that scrambles bits. It consists of some number of stages. Each stage i maps a pair of 32-bit words (L_i, R_i) to a new pair (L_{i+1}, R_{i+1}). By applying the stages in sequence, a t-stage network maps (L_0, R_0) to (L_t, R_t). The (L_0, R_0) is the plaintext, and (L_t, R_t) is the corresponding ciphertext.

Each stage works as follows:

$$L_{i+1} = R_i \tag{1}$$

$$R_{i+1} = L_i \oplus f(R_i, K_i) \tag{2}$$

Here, K_i is a *subkey*, which is generally derived in some systematic way from the master key k.

The security of a Feistel-based code lies in the construction of the function f and in the method for producing the subkeys K_i, However, the invertibility follows just from properties of $\oplus$.

The inversion problem is to find (L_i, R_i) given (L_{i+1}, R_{i+1}). Equation 1 gives us R_i. Knowing R_i and K_i, we can compute $f(R_i, K_i)$. We can then solve equation 2 to get

$$L_i = R_{i+1} \oplus f(R_i, K_i)$$

DES uses a 16 stage Feistel network. The pair $L_0 R_0$ is constructed from a 64-bit message by a fixed initial permutation IP. The ciphertext output is obtained by applying IP^{-1} to $R_{16} L_{16}$.

The scrambling function $f(R_i, K_i)$ operates on a 32-bit data block and a 48-bit key block. Thus, a total of $48 \times 16 = 768$ key bits are used. They are all derived in a systematic way from the 56-bit primary key and are far from independent of each other.

3.2 The Scrambling Function

The scrambling function $f(R_i, K_i)$ is the heart of DES. It operates on a 32-bit data block and a 48-bit key block. Thus, a total of $48 \times 16 = 768$ key bits are used. They are all derived in a systematic way from the 56-bit master key k and are far from independent of each other. In a little more detail, k is split into two 28-bit pieces C and D. At each stage, C and D are rotated by one or two bit positions. Subkey K_i is then obtained by applying a fixed permutation (transposition) to CD. (See Table 3.4c of the text.)

The scrambling function itself is rather involved. However, at its heart are eight "S-boxes". These are boxes with 6 binary inputs $c_0, x_1, x_2, x_3, x_4, c_1$ and 4 binary outputs y_1, y_2, y_3, y_4. Each computes some fixed function in $\{0,1\}^6 \to \{0,1\}^4$. Moreover, each S-box has the very special property that for each of the four possible ways of fixing the values of (c_0, c_1) to Boolean constants, the resulting function on the remaining four inputs $x_1, \ldots, x_4$ is a permutation from $\{0,1\}^4 \to \{0,1\}^4$. Therefore, we can regard an S-box as performing a substitution on four-bit "characters", where the substitution performed depends both on the structure of the particular S-box and on the values of its "control inputs" c_0 and c_1. The eight S-boxes are all different and are specified by their truth tables.

The S-boxes together have a total of 48 input lines. Each of these lines is the output of a corresponding $\oplus$-gate. One input of each of these $\oplus$-gates is connected to a corresponding bit of the 48-bit subkey K_i. (This is the only place that the key enters into DES.) The other input of each $\oplus$-gate is connected to one of the 32 bits of the first argument of f. Since there are 48 $\oplus$-gates and only 32 bits in the first argument to f, some of those bits get used more than once. The mapping of input bits to $\oplus$-gates is called the *expansion permutation* E and is given by Table 3.2(c) in the text. By looking at the table, one sees that the $\oplus$-gates connected to the six inputs $c_0, x_1, x_2, x_3, x_4, c_1$ for S-box 1 are in turn connected to the input bits $32, 1, 2, 3, 4, 5$, respectively. For S-box 2, they go to bits $4, 5, 6, 7, 8, 9$, etc. Thus, inputs bits $1, 4, 5, 8, 9, \ldots 28, 29, 32$ are each used twice, and the remaining input bits are each used once.

Finally, the 32 bits of output from the S-boxes are passed through a fixed permutation P (transposition) that spreads out the output bits. The outputs of a single S-box at one stage of DES become inputs to several different S-boxes at the next stage. This helps provide the desirable "avalanche" effect described in the text.

3.3 Security considerations

We have mentioned previously that DES is vulnerable to a brute force attack because of its small key size of only 56 bits. However, it has turned out to be remarkably resistant to two recently discovered cryptanalysis attacks, differential cryptanalysis and linear cryptanalysis. The former can break DES using "only" 2^{47} chosen ciphertext pairs. The latter works with 2^{43} chosen plaintext pairs. Neither attack is feasible in practice.

DES has now been replaced as a national standard by the new AES (Advanced Encryption Standard), based on the Rijndael algorithm, developed by two Dutch computer scientists. AES supports key sizes of 128, 192, and 256 bits and works on 128-bit blocks. We will say more about it later in the course.