YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

CPSC 467a: Cryptography and Computer Security *Notes 5 (rev. 1)*
Professor M. J. Fischer *September 22, 2008*

Lecture Notes 5

25 Group Property

Let (E, D) be a cryptosystem for which $\mathcal{M} = \mathcal{C}$. Each E_k is then a permutation on $\mathcal{M}$.[1] The set of all permutations on $\mathcal{M}$ forms a group.[2] (E, D) is said to have the *group property* if the set of possible encryption functions $\mathcal{E} = \{E_k \mid k \in \mathcal{K}\}$ is closed under functional composition $\circ$. That is, if $k', k'' \in \mathcal{K}$, then there exists $k \in \mathcal{K}$ such that

$$E_k = E_{k''} \circ E_{k'}.$$

If it is closed, then it can be shown that $(\mathcal{E}, \circ)$ is a subgroup of all permutations on $\mathcal{M}$.

When the cryptosystem is a group, double encryption adds no security against a brute force attack. Even though the key length has doubled, the number of distinct encryption functions has not increased. Since every double encryption is same as a single encryption, the double encryption system will fall to a brute force attack on the original cryptosystem. This is what we saw in section 24.1 with Double Caesar.

26 Birthday Attack when Cryptosystem is a Group

If a cryptosystem system is a group, then it is subject to a known plaintext "birthday" attack,[3] which reduces the number of keys that must be tried to roughly the square root of what a brute force attack needs. For example, if the original key length was 56 (as is the case with DES), then only about $\sqrt{2^{56}} = 2^{28}$ keys need be tried.

Briefly, here's how it works: Assume (m, c) is a known plaintext-ciphertext pair, so $E_{k_0}(m) = c$ for Alice's secret key k_0, which we do not know. Choose 2^{28} random keys k_1 and encrypt m using each. Choose another 2^{28} random keys k_2 and decrypt c using each. Look for a match (non-empty intersection) between these two sets. Suppose one is found for k_1 and k_2, meaning that $E_{k_1}(m) = u = D_{k_2}(c)$. It follows that $E_{k_2}(E_{k_1}(m)) = c$, so we have succeeded in finding a key pair (k_1, k_2) that works for the pair (m, c).

Because we are assuming the cryptosystem is a group, it follows that there is a key k such that $E_k = E_{k_2} \circ E_{k_1}$, so $E_k(m) = c$. Alice's key k_0 also has $E_{k_0}(m) = c$. If it happens that $E_k = E_{k_0}$, then we have broken the cryptosystem. Even though we may not be able to quickly find k from k_1 and k_2, we have no need of doing so since we can compute E_k from E_{k_1} and E_{k_2}, and we can compute D_k from D_{k_1} and D_{k_2}.

Assuming the cryptosystem enjoys reasonable randomness properties, it is unlikely that there are very many distinct keys k such that $E_k = E_{k_0}$, so with significant probability we have cracked

[1] A *permutation* on a set S is a function from $S \to S$ that is one-to-one and onto.

[2] A pair $(X, \circ)$, where $\circ$ is a binary operation on X, is a *group* if $\circ$ is associative, X has an identity element, and every $x \in X$ has an inverse.

[3] Wikipedia states, "In probability theory, the *birthday paradox* states that given a group of 23 (or more) randomly chosen people, the probability is more than 50% that at least two of them will have the same birthday." This is far less than the 253 people that are needed for the probability to exceed 1/2 that at least one of them was born on a specific day, say January 1.

the system. (For Caesar, there is only one such k.) Using additional plaintext-ciphertext pairs, we can verify that we have likely found the correct key pair, or repeat this process again if we have not yet succeeded. I've glossed over many assumptions and details, but that's the basic idea.

The drawback to the birthday attack (from the attacker's perspective) is that it requires a lot of storage in order to find a matching element. Nevertheless, if DES were a group, this attack could be carried out in about a gigabyte of storage, easily within the storage capacity of modern workstations.

27 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.)

27.1 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_0R_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.

27.2 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 tables. (See pages 97–99 of the text (Stinson).)

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 on page 100 of 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, where a change in one input bit spreads out through the network and causes many output bits to change.

27.3 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.