Chapter 2

CONVENTIONAL CRYPTOGRAPHY

Exercises

Exercise 1 Weak Keys of DES

We say that a DES key k is *weak* if DES_k is an involution. Exhibit four weak keys for DES.
Reminder: Let $\mathcal{S}$ be a finite set and let f be a bijection from $\mathcal{S}$ to $\mathcal{S}$. The function f is an *involution* if $f(f(x)) = x$ for all $x \in \mathcal{S}$.

▷ Solution on page 34

Exercise 2 Semi-Weak Keys of DES

We say that a DES key k is *semi-weak* if it is not weak and if there exists a key k' such that

$$\mathsf{DES}_k^{-1} = \mathsf{DES}_{k'}.$$

Exhibit four semi-weak keys for DES.

▷ Solution on page 34

Exercise 3 Complementation Property of DES

Given a bitstring x we let $\overline{x}$ denote the bitwise complement, i.e., the bitstring obtained by flipping all bits of x.

1 Prove that

$$\mathsf{DES}_{\overline{K}}(\overline{x}) = \overline{\mathsf{DES}_K(x)}$$

for any x and K.

2 Deduce a brute force attack against DES with average complexity of 2^{54} DES encryptions.
Hint: Assume that the adversary who is looking for K is given a plaintext block x and the two values corresponding to $\mathsf{DES}_K(x)$ and $\mathsf{DES}_K(\overline{x})$.

▷ Solution on page 35

Exercise 4 3DES Exhaustive Search

1 What is the average complexity of an exhaustive search against the two-key 3DES?

2 How can an adversary take advantage of the complementation property $\mathsf{DES}_{\overline{K}}(\overline{x}) = \overline{\mathsf{DES}_K(x)}$? What is the complexity now?

▷ Solution on page 36

Exercise 5 2DES and Two-Key 3DES

1 2DES encrypts a 64-bit message M in the following manner.

$$C = \mathsf{DES}_{K_1}(\mathsf{DES}_{K_2}(M)).$$

Here, K_1 and K_2 are bitstrings of 56 bits each.

(a) Give the average complexity of a "naive" exhaustive key search?

(b) We perform now a meet-in-the-middle attack. Give an approximate of the time and memory complexities.

2 Two-Key 3DES encrypts a 64-bit message M in the following manner.

$$C = \mathsf{DES}_{K_1}(\mathsf{DES}^{-1}_{K_2}(\mathsf{DES}_{K_1}(M))). \tag{2.1}$$

Here, K_1 and K_2 are strings of 56 bits each.

(a) What is the average complexity of a "naive" exhaustive search?

(b) We are given a box that encrypts a message M according to (2.1). We may use the box to encrypt plaintexts of our choice. Denoting 0 the all-zero message, we first build a table containing

the standard DES decryption of the message 0 under all 2^{56} keys. Then we use a chosen-plaintext attack to build a second table containing the 2^{56} ciphertexts resulting from box encryptions of the elements of the first table. Given these two tables, one can find both K_1 and K_2 used by the encryption box. Explain how one may proceed. The whole attack should take no more than 2^{60} DES encryptions (or decryptions) and no more than 2^{61} bytes of memory.

▷ Solution on page 37

Exercise 6 ⋆Exhaustive Search on 3DES

We consider 3DES with three independent keys. Let $P, C \in \{0,1\}^{64}$ be

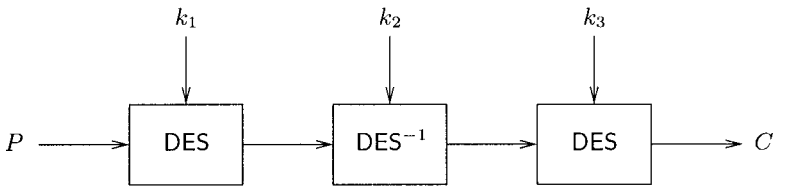


Figure 2.1. 3DES with three independent keys

a plaintext/ciphertext pair, where $C = \text{3DES}_k(P)$ for some unknown key $k = (k_1, k_2, k_3)$ (see Figure 2.1). We want to recover k by an exhaustive search.

1 What is the number of DES encryptions/decryptions of Algorithm 1?

Algorithm 1 Exhaustive key search algorithm on 3DES

Input: a plaintext/ciphertext couple (P, C)
Output: key candidate(s) for $k = (k_1, k_2, k_3)$
Processing:

```
for each possible key K = (K1, K2, K3) do
    if C = 3DES_K(P) then
        display K = (K1, K2, K3)
    end if
end for
```

2 Let $\text{C}^* : \{0,1\}^{64} \to \{0,1\}^{64}$ denote a uniformly distributed random permutation. What is the probability that $\text{C}^*(P) = C$.

3 Assuming that $\mathsf{3DES}_K$ roughly behaves like C^* when K is a uniformly distributed random key, estimate the number of wrong keys (i.e., different from k) displayed by Algorithm 1.

4 Assume that an adversary has t distinct plaintext/ciphertext pairs denoted (P_i, C_i) for $i = 1, \ldots, t$, all encrypted under the same (still unknown) key k (so that $C_i = \mathsf{3DES}_k(P_i)$). Write an algorithm similar to Algorithm 1 that reduces the number of wrong keys that are displayed (but which does at least display k). What is the total number of DES encryptions/decryptions of this algorithm?

5 Express the average number of wrong keys that are displayed by your algorithm in function of t (which is the number of available plaintext/ciphertext couples). Evaluate the necessary number of couples in order to be almost sure that *only* the good key $k = (k_1, k_2, k_3)$ is displayed.

▷ Solution on page 37

Exercise 7 An Extension of DES to 128-bit Blocks

DES is a 64-bit plaintext block cipher which uses a 56 bit key.

1 What is the complexity of exhaustive search against DES?

We can increase the security against exhaustive search in a triple mode by using two-key 3DES.

2 What is the complexity of exhaustive search against 3DES?

3 We now consider the CBC mode of operation. We want to mount a "collision attack". Show how a collision on encrypted blocks in CBC mode can leak some information on the plaintexts. What is the complexity of this attack when the block cipher used is DES? What is the complexity if we replace DES by 3DES? How can we protect ourselves against this attack?

We now try to transform DES into a block cipher with 128-bit plaintext blocks, that we denote ExtDES. We use a 112-bit key which is split into two DES keys K_1 and K_2. For this, we define the encryption of a 128-bit block x as follows:

- we split x into two 64-bit halves x_L and x_R such that $x = x_L \| x_R$
- we let $u_L = \mathsf{DES}_{K_1}(x_L)$ and $u_R = \mathsf{DES}_{K_1}(x_R)$

- we split $u_L \| u_R$ into four 32-bit quarters u_1, u_2, u_3, u_4 such that $u_L = u_1 \| u_2$ and $u_R = u_3 \| u_4$
- we let $v_L = \mathsf{DES}_{K_2}^{-1}(u_1 \| u_4)$ and $v_R = \mathsf{DES}_{K_2}^{-1}(u_3 \| u_2)$
- we split $v_L \| v_R$ into four 32-bit quarters v_1, v_2, v_3, v_4 such that $v_L = v_1 \| v_2$ and $v_R = v_3 \| v_4$
- we let $y_L = \mathsf{DES}_{K_1}(v_1 \| v_4)$ and $y_R = \mathsf{DES}_{K_1}(v_3 \| v_2)$
- we define $y = y_L \| y_R$ as the encryption $\mathsf{ExtDES}_{K_1 \| K_2}(x)$ of x

4 Draw a diagram of ExtDES.

5 Explain how this special mode is retro-compatible with 3DES: if an embedded system implements it, how can it simulate a 3DES device? Same question with DES: how is this special mode retro-compatible with DES?

6 Do you think that the new scheme is more secure than 3DES? Do you think that it is more secure than DES?

7 Let x and x' be two plaintexts, and let $y = \mathsf{ExtDES}_{K_1 \| K_2}(x)$ and $y' = \mathsf{ExtDES}_{K_1 \| K_2}(x')$ be the corresponding known ciphertexts. Explain how a smart choice of x and x' allows us to detect that we have $u_4 = u'_4$ and $v_4 = v'_4$ simultaneously (here u'_4 and v'_4 are the internal intermediate values for computing y').

8 Use the previous question to mount a chosen plaintext attack whose goal is to find a (x, x') pair with $u_4 = u'_4$ and $v_4 = v'_4$ simultaneously. What is the complexity of this attack?

9 Explain how to use this attack in order to reduce the security of ExtDES to the security of DES against exhaustive search? What can you say about the security of ExtDES now?

▷ Solution on page 40

Exercise 8 Attack Against the OFB Mode

Assume that someone sends encrypted messages by using DES in the OFB mode of operation with a secret (but fixed) IV value.

1 Show how to perform a known plaintext attack in order to decrypt transmitted messages.

2 Is it better with the CFB mode?

3 What about the CBC mode?

▷ Solution on page 42

Exercise 9 ⋆Linear Feedback Shift Registers

We consider the ring $\mathbf{Z}_2[X]$ of polynomials with coefficients in $\mathbf{Z}_2$ with the usual addition and multiplication. In the whole exercise, we consider an *irreducible* polynomial $P(X) \in \mathbf{Z}_2[X]$ of degree d. We define the finite field $\mathbf{K} = \mathbf{Z}_2[X]/(P(X))$ of the polynomials with a degree at most $(d-1)$ with coefficients in $\mathbf{Z}_2$, with the usual addition and with the multiplication between $a(X), b(X) \in \mathbf{Z}_2[X]$ defined by

$$a(X) * b(X) = a(X) \times b(X) \bmod P(X).$$

We build a sequence $s_0(X), s_1(X), \ldots$ in $\mathbf{K}$ defined by $s_0(X) = 1$ and $s_{t+1}(X) = X * s_t(X)$ for all $t \geq 0$. We have

$$s_t(X) = X^t \bmod P(X) \quad \text{for all } t \geq 0.$$

1 Compute the first eight elements of the sequence when $P(X) = X^3 + X + 1$. What is the period of the sequence?

2 To each element $q(X) = q_0 + \cdots + q_{d-1}X^{d-1}$ of $\mathbf{K}$ we assign an integer $\widetilde{q}$ defined by

$$\widetilde{q} = q_0 + q_1 \cdot 2 \cdots + q_{d-1} \cdot 2^{d-1}.$$

How is it possible to implement the computation of $\widetilde{s}_{t+1}$ from $\widetilde{s}_t$ with the usual instructions available in a microprocessor?

3 We define $c_{t,j}$ as being the coefficient of X^j in $s_t(X)$ and the $d \times d$ matrix M_t with elements in $\mathbf{Z}_2$ as

$$(M_t)_{i,j} = c_{i+t-1,j-1}$$

for $1 \leq i, j \leq d$ and $t \geq 0$.

- Show that there exists a relation $M_{t+1} = B \times M_t$ and compute the matrix B.
- Show that for a given $0 \leq j \leq d-1$, there exists an order-d linear recurrence relation for the sequence $c_{t+d,j}$ for all $t \geq 0$, i.e., from $c_{t,j}, c_{t+1,j}, \ldots, c_{t+d-1,j}$ one can linearly compute $c_{t+d,j}$.
- How is it possible to build an electronic circuit which computes the sequence defined in the first question with 1-bit registers and 1-bit adders?

4 What are the possible values of the period of the sequence $s_i(X)$ for $i \geq 0$? When is it maximal?

▷ Solution on page 42

Exercise 10 ⋆Attacks on Cascade Ciphers

In this exercise, we consider a block cipher of block length n and of key length ℓ. The encryption function of the block cipher is denoted E. If $P \in \{0,1\}^n$ denotes a plaintext and $k \in \{0,1\}^\ell$ is an encryption key, then $\mathsf{E}_k(P) = C \in \{0,1\}^n$ is the ciphertext obtained by encrypting P under the key k. We denote D the corresponding decryption function, such that $\mathsf{D}_k(\mathsf{E}_k(P)) = P$ for any plaintext $P \in \{0,1\}^n$ and any key $k \in \{0,1\}^\ell$. A *cascade cipher* is the concatenation of $L > 1$ identical block ciphers with *independent* keys, denoted $k_1, \ldots, k_L$. In this configuration, the output of block cipher i is the input of block cipher $i+1$. The plaintext is the input of the first block cipher and the ciphertext is the output of the last block cipher. For simplicity, we denote E_{k_i} and D_{k_i} by E_i and D_i respectively (see Figure 2.2).

1 What is the complexity (in terms of number of encryptions) of the exhaustive key search of Algorithm 2 on the block cipher? What is the complexity of a similar exhaustive key search on a cascade of L block ciphers? Give the name of an attack which reduces this complexity for the specific case where $L = 2$. Recall its complexity.

Algorithm 2 Exhaustive key search algorithm

Input: a plaintext/ciphertext pair (P, C) such that $C = \mathsf{E}_k(P)$
Output: key candidate(s) for k
Processing:
1: **for** each possible key K **do**
2: **if** $C = \mathsf{E}_K(P)$ **then**
3: display K
4: **end if**
5: **end for**

We now wonder how many (wrong) keys are displayed by Algorithm 2.

2 Let $\mathsf{C}^* : \{0,1\}^n \to \{0,1\}^n$ denote a uniformly distributed random permutation. Let x and y be some fixed elements of $\{0,1\}^n$. What is the probability that $\mathsf{C}^*(x) = y$? Let $K \in \{0,1\}^\ell$ be a random

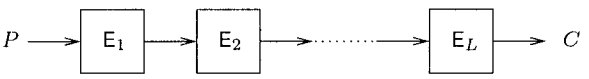


Figure 2.2. A cascade of L block ciphers

variable. Assuming that E_K roughly behaves like C^*, compute an estimation of the amount of wrong keys displayed by Algorithm 2. How many wrong keys are displayed for a similar algorithm on a cascade of L ciphers?

Assume that the adversary knows t plaintext/ciphertext pairs, all corresponding to the same key k.

3 Write an optimized algorithm, similar to Algorithm 2, which exploits these t pairs to reduce the number of wrong guesses. Estimate the number of wrong keys that are displayed.

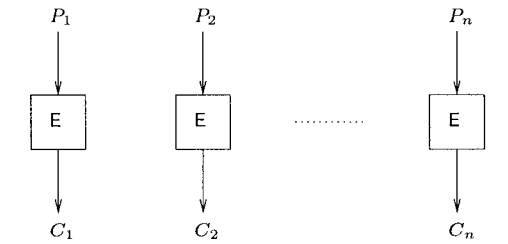
4 If you replace the block cipher by a cascade of L block ciphers in your algorithm, what would be an estimation of the number of wrong keys which are displayed? Using your approximation, how should t be selected in order to be almost sure to have only one good key candidate after an exhaustive search on 3DES (with 3 independent keys)?

▷ Solution on page 44

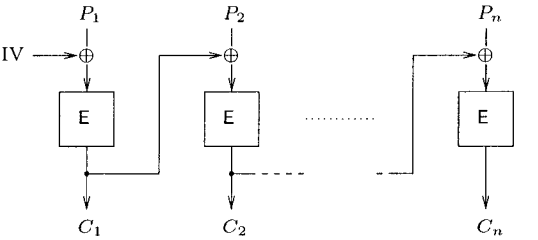
Exercise 11 Attacks on Encryption Modes I

In this exercise, we consider a block cipher of block length n and of key length ℓ. The encryption function of the block cipher is denoted E. If $P \in \{0,1\}^n$ denotes a plaintext, and $k \in \{0,1\}^\ell$ is an encryption key, then $\mathsf{E}_k(P) = C \in \{0,1\}^n$ is the ciphertext obtained by encrypting P under the key k. We denote by D the corresponding decryption function, such that $\mathsf{D}_k(\mathsf{E}_k(P)) = P$ for any plaintext $P \in \{0,1\}^n$ and any key $k \in \{0,1\}^\ell$. Instead of using a simple cascade of block ciphers, we consider so called *multiple modes of operation.* The four modes of operation we will consider are ECB, CBC, OFB, and CFB (represented on Figure 2.3). Just as cascade of block ciphers consists in concatenating block ciphers, multiple modes of operation consist in concatenating modes of operations. For example, the notation CBC|CFB refers to the mode where the output of the CBC mode is the input of the CFB mode (see Figure 2.4).

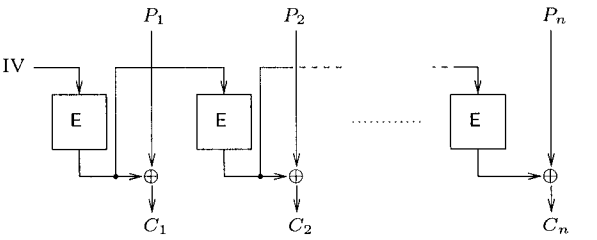
Note that two independent keys are used here, one in the CBC mode, the other in the CFB mode. In this exercise, we assume that $n > \ell$ (i.e., that the block length is larger than the key length) and that *all the* IV*'s are known to the adversary.* For simplicity, we denote E_{k_i} and D_{k_i} by E_i and D_i respectively.



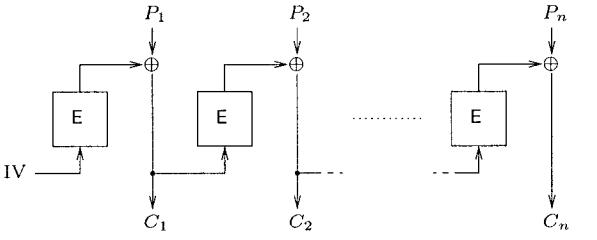
(a) ECB mode



(b) CBC mode



(c) OFB mode



(d) CFB mode

Figure 2.3. Basic modes of operation

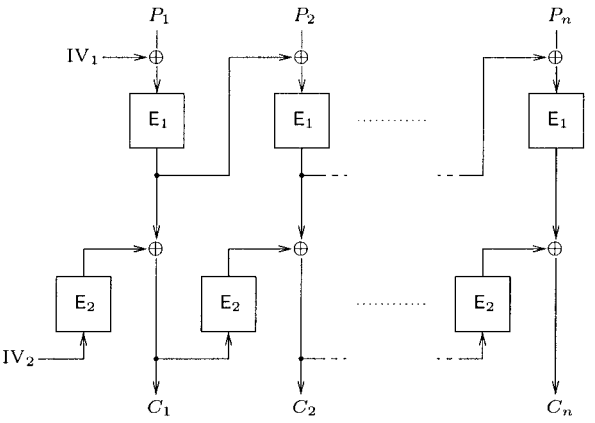


Figure 2.4. The CBC|CFB mode of operation

1 Draw the scheme corresponding to the inversion of the CBC|CFB mode represented in Figure 2.4.

Consider the ECB|ECB|CBC^{-1} mode of operation represented on Figure 2.5. We are going to mount a chosen *plaintext* attack against it. The plaintext P we choose, is the concatenation of three n-bit blocks such that $P = (A, A, B)$ (where $A, B \in \{0,1\}^n$ denote arbitrary blocks of n bits). The three blocks of the corresponding ciphertexts are denoted C_1, C_2, and C_3.

2 Using the notations of Figure 2.5, find a relation between A'', k_3, IV, and C_1. Similarly, find a relation between A'', IV, C_1, and C_2. Deduce a relation between k_3, IV, C_1, and C_2.

3 Deduce an attack which recovers k_3. Once k_3 is found, how do you recover k_1 and k_2? What is the complexity of the whole attack?

We now consider the OFB|CBC|ECB mode (see Figure 2.6). This time, we are going to mount a *chosen-ciphertext* attack. The ciphertext C we choose, is the concatenation of four n-bit blocks such that $C = (A, A, B, B)$ (where A, B denote arbitrary blocks of n bits). The four blocks of the corresponding plaintext are denoted P_1 to P_4.

4 Find a relation between k_1, k_3, IV_1, IV_2, P_1, P_2 and A. Similarly, find a relation between k_1, k_3, IV_1, P_3, P_4, A, and B.

5 Deduce a (smart) attack that recovers k_1 and k_3. Once this is done, how can k_2 be recovered? Compute the complexity of the attack.

▷ Solution on page 45

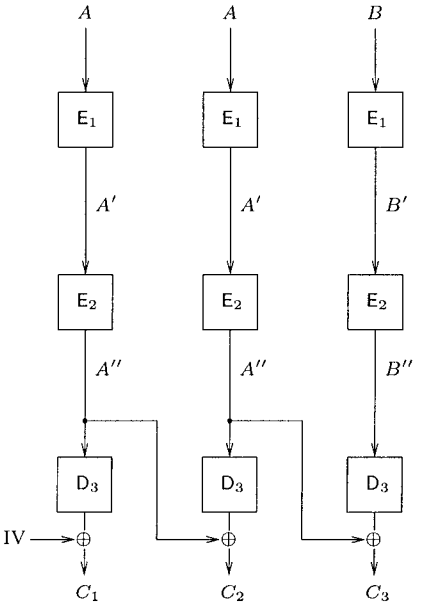


Figure 2.5. Attacking the ECB|ECB|CBC^{-1} mode of operation

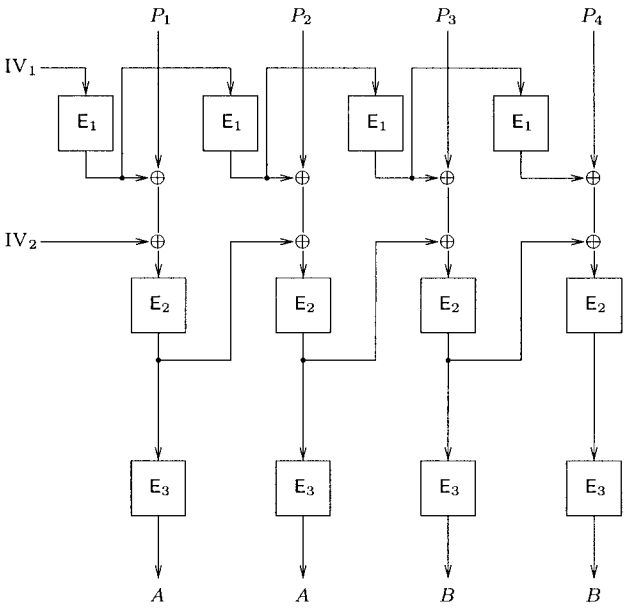


Figure 2.6. Attacking the OFB|CBC|ECB mode of operation

Exercise 12 Attacks on Encryption Modes II

We use the notations of the previous exercise. Here, we consider the CBC|CBC^{-1}|CBC^{-1} mode (represented on Figure 2.7 for two plaintext blocks). For this attack, we mount a chosen-ciphertext attack. Moreover, the adversary will have the ability to *choose* the value of IV_2 (the values of IV_1 and IV_3 are only known and fixed). The attack we will consider is described in Algorithm 3. We denote $C^{(i)} = (C_1^{(i)}, C_2^{(i)})$ the ith chosen ciphertext and $P^{(i)} = (P_1^{(i)}, P_2^{(i)})$ the corresponding plaintext. Similarly, $\mathrm{IV}_2^{(i)}$ denote the ith chosen value for IV_2.

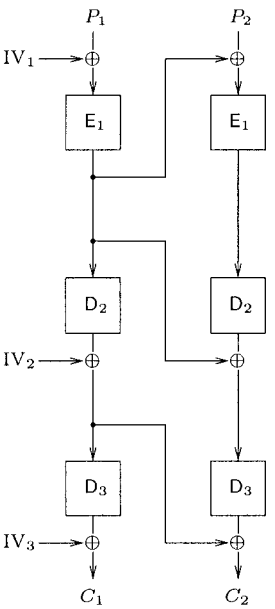


Figure 2.7. Attacking the CBC|CBC^{-1}|CBC^{-1} mode of operation

1 Give an approximation of the complexity of Algorithm 3.

2 Show that if $P_1^{(i)} = P_1^{(j)}$, then $P_2^{(i)} = P_2^{(j)}$
Hint: Use the fact that we set $C_2^{(i)}$ to $\mathrm{IV}_2^{(i)}$ in Algorithm 3.

3 Find a relation between $\mathrm{IV}_2^{(i)}$, $\mathrm{IV}_2^{(j)}$, K_3, IV_3, $C_1^{(i)}$, and $C_1^{(j)}$ equivalent to the condition $P_1^{(i)} = P_1^{(j)}$.

4 Deduce an attack that recovers the value of K_3. Once K_3 is found, how can K_1 and K_2 be recovered? What is the overall complexity of the attack?

Algorithm 3 Looking for collisions in CBC|CBC^{-1}|CBC^{-1}

Output: $P^{(i)}$, $P^{(j)}$, $C^{(i)}$, and $C^{(j)}$, such that $P_1^{(i)} = P_1^{(j)}$
Processing:
1: $i \leftarrow 1$
2: **repeat**
3: Choose $C_1^{(i)}$ and $\mathrm{IV}_2^{(i)}$ at random
4: $C_2^{(i)} \leftarrow \mathrm{IV}_2^{(i)}$
5: Obtain and store $P_1^{(i)}$ and $P_2^{(i)}$
6: $i \leftarrow i + 1$
7: **until** $P_1^{(i)} = P_1^{(j)}$ for some $j < i$
8: Display $P^{(i)}$, $P^{(j)}$, $C^{(i)}$, and $C^{(j)}$

▷ Solution on page 47

Exercise 13 ⋆A Variant of A5/1 I

In stream ciphers, the prevailing encryption is a bitwise XOR operation between the m-bit plaintext and the m-bit keystream which is the output of a so-called keystream generator fed by the ℓ-bit secret key, where m is much larger than ℓ. An ideal assumption for good stream ciphers is that any ℓ-bit window of the m-bit keystream is eventually modified when the ℓ-bit key is modified. This exercise aims at doing a small test of the above assumption, taking as an example the A5/1 keystream generator. A5/1 consists of three Linear Feedback Shift Registers (LFSRs) denoted by R_1, R_2, and R_3, with respective length of 19, 22, and 23 bits. The total content of all three LFSRs is $19+22+23 = 64$ bits. Hereafter we call the 64-bit initial content (also called initial state) of the three LFSRs as the key of A5/1. We denote by $R_i[n]$ the content of the nth cell of R_i, for $i = 1, 2, 3$, where n starts at 0. Each LFSR has one clocking tap: $R_1[8]$, $R_2[10]$, and $R_3[10]$. At each clock cycle, one keystream bit is generated according to the following procedures (see Figure 2.8):

- The three LFSRs make a clocking vote according to the majority of the current three clocking taps.
- Each R_i compares the voting result with its own clocking tap. If they are equal, R_i is shifted:
  - a feedback bit is computed by XORing the content of the fixed subset of cells of R_i, i.e., the feedback for R_1, R_2, and R_3 is

$R_1[18] \oplus R_1[17] \oplus R_1[16] \oplus R_1[13]$, $R_2[21] \oplus R_2[20]$, and $R_3[22] \oplus R_3[21] \oplus R_3[20] \oplus R_3[7]$ respectively;

- the content of all cells in R_i (except the leftmost) are shifted to the left by one position simultaneously;
- $R_i[0]$ is updated by the precomputed feedback;

- Output the bit $R_1[18] \oplus R_2[21] \oplus R_3[22]$.

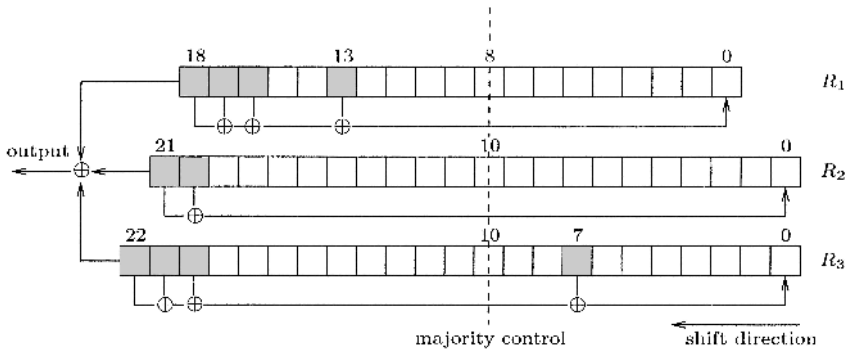


Figure 2.8. A5/1 keystream generator

1 Show that when R_1 is loaded with a special initial state, then, regardless of its movement in the future, its state never changes. Is it possible to extend your solution to R_2 and R_3?

2 Use the previous answer to disprove the aforementioned assumption in the following special case of A5/1: show that the all-zero 64-bit keystream can be generated by different 64-bit keys.

3 Compute a tight lower bound on the number of different keys that generate such a keystream.

Let us now consider a variant of A5/1, by replacing the majority function with the minority function for the clocking vote, where the minority function of three binary bits a, b, c is defined by

$$\text{minority}(a,b,c) = \begin{cases} \overline{a} & \text{if } a = b = c \\ a \oplus b \oplus c & \text{otherwise.} \end{cases}$$

4 Similarly to Question 2, show that several keys will produce the all-zero 64-bit keystream for this variant.

5 Recompute Question 3 under the constraint that initially two clocking taps out of three are both one.

6 Check whether the assumption is true or false now for this variant of A5/1.

7 Compare the lower bounds obtained in questions 3 and 5, and briefly discuss the security strength of A5/1 and its variant.

▷ Solution on page 49

Exercise 14 ⋆A Variant of A5/1 II

We consider the A5/1 keystream generator described in Exercise 13 and shown on Figure 2.8. We assume that the three initial values of the LFSRs are chosen independently and uniformly at random.

1 For $i = 1, 2, 3$, what is the probability that R_i is shifted at the first clock? What is the probability that it is not shifted?

2 What is the probability that exactly two LFSRs are shifted at the first clock?

3 What is the probability mass function for the movement of three LFSRs at the first clock?

4 What is the conditional probability mass function of the first clocking given the initial clocking?

We define the minority function between three binary bits a, b, c by

$$\text{minority}(a, b, c) = \begin{cases} \overline{a} & \text{if } a = b = c \\ a \oplus b \oplus c & \text{otherwise.} \end{cases}$$

We consider a variant of A5/1 where we replace the majority function with the minority function for the clocking vote.

5 Recompute the previous questions for this variant of A5/1.

6 What conclusion can you draw about the security strength of using majority and minority function for the clocking vote?

▷ Solution on page 51

Exercise 15 ⋆Memoryless Exhaustive Search

A cryptanalyst would like to break a keyed cryptographic system. Assume he has access to an oracle which, for each queried key, answers whether it is the correct one or not. We use the following notations.

- The total number of possible keys is denoted N. The list of all possible keys is denoted $\{k_1, k_2, \ldots, k_N\}$.

- The random variable corresponding to the key known by the oracle is denoted K, i.e., the correct key known by the oracle is k_i ($i \in \{1, \ldots, N\}$) with probability $\Pr[K = k_i]$. Unless specified, K is *not* assumed to be uniformly distributed.

- The random variable corresponding to the key chosen by the cryptanalyst is denoted $\widetilde{K}$, i.e., the probability that the cryptanalyst sends k_i ($i \in \{1, \ldots, N\}$) to the oracle is $\Pr[\widetilde{K} = k_i]$.

The cryptanalyst iteratively queries the oracle with randomly selected keys, in an independent way, until he finds the right one. Note that, as the queries are independent, the complexity could in principle be infinite (we say that the algorithm is memoryless). The strategy of the cryptanalyst is to select a distribution for his queries.

1 Compute the expected complexity $\mathrm{E}[C]$ (in terms of oracle queries) in general, and when the key distribution is uniform (i.e., when K is uniformly distributed). How do you improve the attack?

2 If the *a priori* distribution of the keys is not uniform (but known by the adversary), what is the best memoryless algorithm for finding the key with the oracle? Prove that its complexity relates to the Rényi entropy of coefficient $\frac{1}{2}$ defined by

$$H_{\frac{1}{2}}(K) = \left(\sum_{i=1}^{N} \sqrt{\Pr[K = k_i]} \right)^2 .$$

Reminder: Lagrange multipliers can be used to find the extremum of a function

$$\begin{array}{rccc} f: & \mathbf{R}^n & \longrightarrow & \mathbf{R} \\ & (x_1, x_2, \ldots, x_n) & \longmapsto & f(x_1, x_2, \ldots, x_n), \end{array}$$

subject to the $k < n$ constraints

$$\begin{cases} g_1(x_1, x_2, ..., x_n) &= 0, \\ g_2(x_1, x_2, ..., x_n) &= 0, \\ \qquad \vdots & \\ g_k(x_1, x_2, ..., x_n) &= 0, \end{cases} \tag{2.2}$$

where $f, g_1, \ldots, g_k$ are functions with continuous first partial derivatives. Consider the function $\Phi : \mathbf{R}^n \to \mathbf{R}$ defined by

$$\Phi(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_n) + \sum_{i=1}^{k} \lambda_i g_i(x_1, x_2, \ldots, x_n).$$

The λ_i's are the Lagrange multipliers. If a point $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbf{R}^n$ is an extremum of f under the conditions (2.2), it must satisfy

$$\begin{cases} g_1(\mathbf{a}) = g_2(\mathbf{a}) = \cdots = g_k(\mathbf{a}) = 0, \\ \dfrac{\partial \Phi}{\partial x_1}(\mathbf{a}) = \dfrac{\partial \Phi}{\partial x_2}(\mathbf{a}) = \cdots = \dfrac{\partial \Phi}{\partial x_n}(\mathbf{a}) = 0. \end{cases} \tag{2.3}$$

Therefore, in order to find an extremum of f under the conditions given by (2.2), one should solve (2.3) with respect to the variables $a_1, a_2, \ldots, a_n, \lambda_1, \ldots, \lambda_k$.

▷ Solution on page 53

Solutions

Solution 1 Weak Keys of DES

If the subkeys k_1 to k_{16} are equal, then the reversed and original key schedules are identical. In that case, DES_k clearly is an involution. The sixteen subkeys will be equal when the registers C and D are all-zero or all-one bit vectors, as the rotation of such bitstrings has no effect on them. Therefore, the four weak keys of DES can easily be computed by applying $\mathrm{PC1}^{-1}$ to the four possible combinations of these C and D values. We have represented the weak keys of DES on Table 2.1, where $\{b\}^n$ denotes a sequence of n bits all equal to b. The existence of weak keys is known at least since the publication of [14].

Table 2.1. Weak keys of DES

C	D	k
$\{0\}^{28}$	$\{0\}^{28}$	$\mathrm{PC1}^{-1}(\{0\}^{28}\|\{0\}^{28})$
$\{0\}^{28}$	$\{1\}^{28}$	$\mathrm{PC1}^{-1}(\{0\}^{28}\|\{1\}^{28})$
$\{1\}^{28}$	$\{0\}^{28}$	$\mathrm{PC1}^{-1}(\{1\}^{28}\|\{0\}^{28})$
$\{1\}^{28}$	$\{1\}^{28}$	$\mathrm{PC1}^{-1}(\{1\}^{28}\|\{1\}^{28})$

Solution 2 Semi-Weak Keys of DES

First, note that it is possible to generate a DES *decryption* schedule on-the-fly. After k_{16} is generated, the values of C and D are equal to the original ones, since they both have been submitted to a 28-bit rotation. Thus, provided that one exchanges the left rotations with right rotations and the amount of the *first* rotation to 0 (instead of 1), the same algorithm used to generate k_1 up to k_{16} can also generate the subkeys k_{16} down to k_1.

A pair of semi-weak keys occurs when the subkeys k_1 through k_{16} of the first key are respectively equal to the subkeys k'_{16} through k'_1 of the second one. This requires that the following system of equations is

verified.

$$\begin{cases} \mathrm{ROL}_{r_1}(C) & = \mathrm{ROL}_{r_1+\cdots+r_{16}}(C') \\ \mathrm{ROL}_{r_1+r_2}(C) & = \mathrm{ROL}_{r_1+\cdots+r_{15}}(C') \\ & \vdots \\ \mathrm{ROL}_{r_1+\cdots+r_{16}}(C) & = \mathrm{ROL}_{r_1}(C') \end{cases}$$

Of course, a similar system should also hold between D and D'. Replacing the r_i's by their values, it is easy to see that the systems imply that $C = \mathrm{ROL}_{2i+1}(C')$ and $D = \mathrm{ROL}_{2i+1}(D')$ for any integer i. From this, we deduce the possible shapes of subkeys registers. They are represented on Table 2.2, where $\{b\}^n$ denotes a sequence of n bits all equal to b and where $\{b_1b_2\}^n$ denotes a sequence of $2n$ bits having the following shape: $b_1b_2b_1b_2\cdots b_1b_2$. The final semi-weak keys are obtained by applying $\mathrm{PC1}^{-1}$ on (C, D) and on (C', D'). The existence of semi-weak keys is known at least since the publication of [14].

Table 2.2. Semi-weak key pairs of DES

C	D	C'	D'
$\{01\}^{14}$	$\{01\}^{14}$	$\{10\}^{14}$	$\{10\}^{14}$
$\{01\}^{14}$	$\{10\}^{14}$	$\{10\}^{14}$	$\{01\}^{14}$
$\{01\}^{14}$	$\{0\}^{28}$	$\{10\}^{14}$	$\{0\}^{28}$
$\{01\}^{14}$	$\{1\}^{28}$	$\{10\}^{14}$	$\{1\}^{28}$
$\{0\}^{28}$	$\{01\}^{14}$	$\{0\}^{28}$	$\{10\}^{14}$
$\{1\}^{28}$	$\{01\}^{14}$	$\{1\}^{28}$	$\{10\}^{14}$

Solution 3 Complementation Property of DES

1 First note that $\overline{x} \oplus y = \overline{x \oplus y}$ and that $\overline{x} \oplus \overline{y} = x \oplus y$. The initial and final permutations (IP and IP^{-1}) do not have any influence on our computations, so we will not consider them. We can write one round of DES as

$$(C_L, C_R) \leftarrow (P_R, P_L \oplus F(P_R, K))$$

where P_L and P_R denote the left and right half of the plaintext, respectively, where C_L and C_R denote the left and right half of the ciphertext and where K denotes the key. From the definition of the key schedule algorithm, we see that if we take the bitwise complement of the key, then each subkey will turn into its bitwise complement as well. Furthermore, from DES F-function definition, we can see that if we complement its input and the subkey, then the input of the

S-boxes and thus the output will remain the same. We can thus write

$$(C_L, C_R) \leftarrow (\overline{P_R}, \overline{P_L} \oplus F(P_R, K)) = \overline{(P_R, P_L \oplus F(P_R, K))}$$

If we extend this to the whole Feistel scheme, then we can conclude that $\mathsf{DES}_{\overline{K}}(\overline{x}) = \overline{\mathsf{DES}_K(x)}$.

2 Algorithm 4 describes a brute force attack that exploits the complementation property of DES. Note that in this algorithm, $\overline{c}$ corresponds to $\overline{\mathsf{DES}_k(x)} = \mathsf{DES}_{\overline{k}}(\overline{x})$. Therefore, if the condition of line 6 is true, we almost surely have $K = \overline{k}$. In the loop, the only *heavy* computation is the computation of $\mathsf{DES}_k(x)$, and we expect to perform 2^{54} such computations.

Algorithm 4 Brute force attack using the complementation property

Input: a plaintext x and two ciphertexts $\mathsf{DES}_K(x)$ and $\mathsf{DES}_K(\overline{x})$
Output: the key candidate for K
Processing:

```
1: for all non-tested key k do
2:    c ← DES_k(x)
3:    if c = DES_K(x) then
4:       output k and stop.
5:    end if
6:    if c̄ = DES_K(x̄) then
7:       output k̄ and stop.
8:    end if
9: end for
```

The complementation property of DES is known at least since the publication of [14].

Solution 4 3DES Exhaustive Search

1 As the total length of the key is 112 bits, the average complexity of an exhaustive search against two-key 3DES is $\frac{1}{2} \cdot 2^{112} = 2^{111}$.

2 It is easy to see that the complementation property of DES can be extended to 3DES:

$$\begin{aligned}
\mathsf{3DES}_{\overline{K_1},\overline{K_2}}\left(\overline{P}\right) &= \mathsf{DES}_{\overline{K_1}}\left(\mathsf{DES}^{-1}_{\overline{K_2}}\left(\mathsf{DES}_{\overline{K_1}}\left(\overline{P}\right)\right)\right) \\
&= \mathsf{DES}_{\overline{K_1}}\left(\mathsf{DES}^{-1}_{\overline{K_2}}\left(\overline{\mathsf{DES}_{K_1}\left(P\right)}\right)\right) \\
&= \mathsf{DES}_{\overline{K_1}}\left(\overline{\mathsf{DES}^{-1}_{K_2}\left(\mathsf{DES}_{K_1}\left(P\right)\right)}\right) \\
&= \overline{\mathsf{3DES}_{K_1,K_2}\left(P\right)}.
\end{aligned}$$

Using an algorithm very similar to Algorithm 4 (where we just replace DES_K by $\mathsf{3DES}_{K_1,K_2}$), we can reduce the complexity by a factor 2. The average complexity becomes 2^{110}.

Solution 5 2DES and Two-Key 3DES

1 (a) A naive exhaustive search has a worst-case complexity of 2^{112} DES evaluations and an average complexity of 2^{111} DES evaluations.

(b) A meet-in-the-middle attack has a memory complexity of 2^{56} 64-bit blocks and a computational complexity of approximately $2 \cdot 2^{56}$ DES evaluations.

2 (a) A naive exhaustive search for a two-key 3DES has a worst-case complexity of $3 \cdot 2^{112}$ DES evaluations and an average complexity of $3 \cdot 2^{111}$ DES evaluations.

(b) The attack is given in Algorithm 5. It focuses on the case where the result after the first encryption stage is the all-zero vector, denoted by 0. Note that in the algorithm,

$$C_{K_1} = \mathsf{DES}_{K_1}(\mathsf{DES}^{-1}_{K_2}(0)),$$

and thus,

$$B_{K_1} = \mathsf{DES}^{-1}_{K_2}(0) = P_{K_2}.$$

Consequently, the two keys k_1, k_2 found in line 10 in the algorithm (such that $B_{k_1} = P_{k_2}$) are indeed a candidate solution pair. The number of DES encryptions in Algorithm 5 is $2^{56} \cdot 5 < 2^{60}$. Both tables store 2^{56} entries of $56 + 64 = 120 < 2^7$ bits each. The memory requirements is thus $2 \cdot 2^{56} \cdot 2^7 \cdot 2^{-3} = 2^{61}$ bytes.

Solution 6 ⋆Exhaustive Search on 3DES

1 The algorithm successively tries each possible key. It does not stop until the last possible key is tried. Therefore, the number of iterations

Algorithm 5 Attacking two-key 3DES

Input: a box $\mathsf{3DES}_{K_1,K_2}(\cdot)$ encrypting 64-bit plaintexts according to (2.1), under the keys K_1 and K_2
Output: K_1 and K_2
Processing:

1: **for all** $k \in \{0,1\}^{56}$ **do**
2: $P_k \leftarrow \mathsf{DES}_k^{-1}(0)$
3: store (P_k, k) in a table T_1 (sorted according to P_k)
4: $C_k \leftarrow \mathsf{3DES}_{K_1,K_2}(P_k)$
5: $B_k \leftarrow \mathsf{DES}_k^{-1}(C_k)$
6: store (B_k, k) in a table T_2 (sorted according to B_k)
7: **end for**
8: sort the table T_1 according to the P_k's values
9: sort the table T_2 according to the B_k's values
10: Store the keys $k_1, k_2 \in \{0,1\}^{56}$ such that $B_{k_1} = P_{k_2}$ in another table T. This table contains candidate solution pairs $K_1 = k_1$ and $K_2 = k_2$.
11: If there are more than one candidate in T, test each key pair on a small number of plaintext/ciphertext pairs until only one remains. Display this solution.

is exactly equal to the number of possible keys times the number of DES encryptions for each (which is 3). Therefore, the number of DES encryptions/decryptions of the algorithm is $3 \cdot 2^{3 \cdot 56} = 3 \cdot 2^{168}$.

2 The random permutation C^* is uniformly distributed among all possible permutations, and there are $(2^{64})!$ of them. Consequently, if $\mathsf{c} : \{0,1\}^{64} \to \{0,1\}^{64}$ is a given permutation, we have $\Pr[\mathsf{C}^* = \mathsf{c}] = \frac{1}{(2^{64})!}$ (see Exercise 1 in Chapter 1). Now, we are given two (fixed) values $P, C \in \{0,1\}^{64}$. We have

$$\begin{aligned} \Pr[\mathsf{C}^*(P) = C] &= \sum_{\mathsf{c}} \mathbf{1}_{\mathsf{C}^*(P)=C} \Pr[\mathsf{C}^* = \mathsf{c}] \\ &= \frac{1}{(2^{64})!} \sum_{\mathsf{c}} \mathbf{1}_{\mathsf{C}^*(P)=C}, \end{aligned}$$

where the last sum simply is the number of permutations mapping P on C, which is the number of permutations of a set of cardinality $2^{64} - 1$. Finally,

$$\Pr[\mathsf{C}^*(P) = C] = \frac{(2^{64} - 1)!}{(2^{64})!} = 2^{-64}.$$

3 We assume that $\Pr_K[\mathsf{3DES}_K(P) = C] = \Pr_{\mathsf{C}^*}[\mathsf{C}^*(P) = C] = 2^{-64}$. Multiplying this probability by the number of tried keys, we obtain the number of keys that are displayed:

$$N = 2^{-64} \cdot 2^{168} = 2^{104}.$$

All the displayed keys (except one) are wrong keys!

4 We consider Algorithm 6. The algorithm clearly displays k as we do

---

Algorithm 6 Exhaustive key search algorithm on 3DES, using t plaintext/ciphertext pairs

---

Input: t plaintext/ciphertext pairs (P_i, C_i), for $i = 1, \ldots, t$, all encrypted under the same key k
Output: key candidate(s) for $k = (k_1, k_2, k_3)$
Processing:
1: **for** each possible key $K = (K_1, K_2, K_3)$ **do**
2: **if** $C_i = \mathsf{3DES}_K(P_i)$ for $i = 1, \ldots, t$ **then**
3: display $K = (K_1, K_2, K_3)$
4: **end if**
5: **end for**

---

have $C_i = \mathsf{3DES}_k(P_i)$ for all $i = 1, \ldots, t$. It reduces the number of wrong keys that are displayed because it is clearly more difficult to find a wrong key $\widetilde{k}$ satisfying $C_i = \mathsf{3DES}_{\widetilde{k}}(P_i)$ for $i = 1, \ldots, t$ (with $t > 1$) than to find a wrong key such that $C = \mathsf{3DES}_{\widetilde{k}}(P)$ (for only one pair). The total number of encryption/decryption steps that have to be performed is simply t times the number found in the first question (we assume that we always perform t times 3DES in the **if** statement of the algorithm). Therefore, this algorithm needs $3 \cdot 2^{168} \cdot t$ encryptions/decryptions.

5 Still assuming that $\Pr_K[\mathsf{3DES}_K(P) = C] = \Pr_{\mathsf{C}^*}[\mathsf{C}^*(P) = C] = 2^{-64}$, the mean value N of wrong keys displayed by Algorithm 6 is

$$\begin{aligned} N &= \text{number of tried keys} \times \prod_{i=1}^{t} \Pr_K[\mathsf{3DES}_K(P_i) = C_i] \\ &= 2^{168} \cdot (2^{-64})^t. \end{aligned}$$

Table 2.3 gives the approximate number N of wrong keys that are displayed, in terms of the number t of available plaintext/ciphertext pairs. According to this table, only 3 pairs are necessary to make almost sure that only the good key will be displayed.

Table 2.3. Average value N of wrong keys that are displayed by Algorithm 6, in terms of the number t of plaintext/ciphertext pairs

t	1	2	3
N	2^{104}	2^{40}	2^{-24}

Solution 7 An Extension of DES to 128-bit Blocks

1 The exhaustive search complexity is 2^{56} in the worst case. It is 2^{55} in average and can be reduced by a factor of 2 by using the complementation property (see Exercise 3 in this chapter).

2 A key for 3DES consists of two keys for DES, so the key length is 112. The exhaustive search complexity is thus 2^{112} in the worst case for 3DES. It is 2^{111} in average and can be further reduced by a factor of 2 by using the complementation property (see Exercise 4 in Chapter 2).

3 In CBC mode of operation, the ith ciphertext block y_i is

$$y_i = \mathsf{DES}_K(y_{i-1} \oplus x_i), \quad \text{with } i \neq j,$$

where x_i is the ith plaintext block. If it happens that $y_i = y_j$ (which is a collision), we deduce that $y_{i-1} \oplus x_i = y_{j-1} \oplus x_j$ which leads to

$$y_{i-1} \oplus y_{j-1} = x_i \oplus x_j.$$

Hence, we can thus deduce some plaintext information from the value $y_{i-1} \oplus y_{j-1}$. The complexity corresponds to the expected number of blocks after which we can expect a collision (see Exercise 1, Chapter 3). According to the Birthday Paradox, we know that we need a number of blocks within the order of magnitude of the square root of the cardinality of the output domain, i.e., $\sqrt{2^{64}} = 2^{32}$. We note that the complexity of this attack is not increased by using 3DES instead of DES as the block size remains the same. In order to thwart this attack, we thus need to enlarge the block size.

4 See Figure 2.9.

5 With $x_L = x_R$, we obtain $y_L = y_R = \mathsf{3DES}_{K_1,K_2}(x_L)$. So a circuit which computes this new scheme can be used to compute 3DES.

Similarly, with $K_1 = K_2$, we obtain compatibility with DES.

6 The previous question leads to the intuition that this new scheme is at least as strong as DES and 3DES. It seems more secure than DES

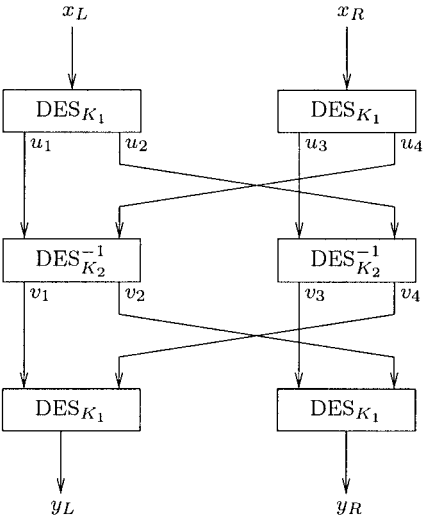


Figure 2.9. A 128 bit extention of DES

as the key size is increased and at least as secure as 3DES as the key size is the same. The advantage of this scheme is that it is protected against the collision attack in CBC mode.

7 If we choose x and x' such that $x_L = x'_L$, then

$$u_4 = u'_4 \text{ and } v_4 = v'_4 \quad \Leftrightarrow \quad y_L = y'_L.$$

8 We take an arbitrary (fixed) 64-bit string α. For many 64-bit strings β we encrypt $x = \alpha \| \beta$. With non-negligible probability, we will get a collision on the y_L's after a number of encryption within the order of magnitude of 2^{32}. We will thus get $x = \alpha \| \beta$ and $x' = \alpha \| \beta'$ such that $u_4 = u'_4$ and $v_4 = v'_4$. The complexity is within the order of magnitude of 2^{32}.

9 After the previous attack, the equation $u_4 = u'_4$ can be written as the equality between the 32 rightmost bits of $\mathsf{DES}_{K_1}(\beta)$ and $\mathsf{DES}_{K_1}(\beta')$. The equation $v_4 = v'_4$ can be written as the equality between the 32 rightmost bits of $\mathsf{DES}^{-1}_{K_1}(y_L)$ and $\mathsf{DES}^{-1}_{K_1}(y'_L)$. We can thus perform an exhaustive search in order to recover K_1, by testing both equalities. This attack requires 2^{56} operations. Note that with high probability, only the right key is raised. Once K_1 is found, 2^{56} additional operations are required in order to recover K_2.

We now see that this new scheme can be broken within about 2^{56} operations. Consequently, it is not more secure than DES and definitely less secure than 3DES.

Solution 8 Attack Against the OFB Mode

1 The OFB mode is nothing but a one-time pad with a sequence generated from the IV and the secret key. If they are both fixed, the sequence is always the same as it is independent from the plaintext. Therefore, from a known plaintext attack with only one known message, we can recover the key stream and decrypt any new ciphertext (of the same length or shorter).

2 The CFB mode is stronger against this issue, except for the first block. The first encrypted block is equal to the first plaintext block XORed with a value generated from IV and from the key only. The next values in the sequence depend on the plaintext. Similarly, note that if two plaintexts are equal on their first n blocks, the knowledge of one of the plaintexts allows to recover the $(n+1)$th block of the other plaintext.

3 The CBC mode is not vulnerable to this kind of attack.

Solution 9 ⋆Linear Feedback Shift Registers

1 The first eight elements of the sequence are given in Table 2.4, from which it is clear that the period is equal to 7.

Table 2.4. The first values of the simple LFSR sequence

i	$s_i(X)$	i	$s_i(X)$
0	1	4	X^2+X
1	X	5	X^2+X+1
2	X^2	6	X^2+1
3	$X+1$	7	1

2 We use a LSL (Logical Shift Left) instruction which shifts an integer one bit to the left. Furthermore, we suppose that we can test the bit in position d (the leftmost one being in the carry flag after a shift). If it is equal to 1, then one subtracts $P(X)$ in order to get the remainder. Note that subtracting $P(X)$ simply corresponds to XORing $P(X)$ as we work modulo 2 here.

3 We let $P(X) = P_0 + P_1X + \cdots + P_{d-1}X^{d-1} + X^d$. Let $Q(X) = Q_0 + \cdots + Q_{d-1}X^{d-1}$ be a polynomial of $\mathbf{K}$. It can be represented by a row

vector $\overline{Q} = (Q_0, \ldots, Q_{d-1}) \in \mathbf{Z}_2^d$. Consequently, the multiplication by X in $\mathbf{K}$ can be represented by a matrix multiplication. Indeed, if we denote by $R(X) = R_0 + \cdots + R_{d-1}X^{d-1} = X * Q(X)$, we have

$$R(X) = Q_{d-1} \cdot P_0 + (Q_0 \oplus Q_{d-1} \cdot P_1)X + \cdots + (Q_{d-2} \oplus Q_{d-1} \cdot P_{d-1})X^{d-1}$$

or equivalently

$$\overline{R} = (Q_{d-1} \cdot P_0, Q_0 \oplus Q_{d-1} \cdot P_1, \ldots, Q_{d-2} \oplus Q_{d-1} \cdot P_{d-1}).$$

From the previous equation, it is clear that the multiplication by X can be represented by

$$\overline{R} = \overline{Q} \times B \quad \text{where} \quad B = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ P_0 & P_1 & P_2 & \cdots & P_{d-1} \end{pmatrix}.$$

By definition of the sequence, we thus have $\overline{s}_{i+t} = \overline{s}_{i+t-1} \times B$ for all $i \geq 1$ and $t \geq 0$. Noting that the ith row of M_t corresponds to $\overline{s}_{i+t-1}$ and that the ith of M_{t+1} corresponds to $\overline{s}_{i+t}$, we deduce that

$$M_{t+1} = M_t \times B.$$

Noting that M_0 is the identity matrix, we can see that $M_t = B^t$ for all $t \geq 0$. Consequently M_t and B commute, so that $M_{t+1} = B \times M_t$. The linear recurrence is now given by

$$\begin{aligned} c_{t+d,j} &= (M_{t+1})_{d,j+1} \\ &= (B \times M_t)_{d,j+1} \\ &= P_0 \cdot c_{t,j} \oplus P_1 \cdot c_{t+1,j} \oplus \cdots \oplus P_{d-1} \cdot c_{t+d-1,j}. \end{aligned}$$

If we take the irreducible polynomial of degree 3 of the first question as an example, we obtain

$$c_{t+3,j} = c_{t,j} \oplus c_{t+1,j}$$

which can be computed by the circuit shown on Figure 2.10.

4 The natural subgroup $\mathbf{K}^*$ of the field $\mathbf{K}$ is of cardinality $(2^d - 1)$. The set $\{X^t \bmod P(X)\ ,\ t \geq 0\}$ being a subgroup of $\mathbf{K}^*$, its order must divide $(2^d - 1)$. Thus, the period of the sequence must be a divisor of $(2^d - 1)$. The period is maximal if X is a primitive element of $\mathbf{K}$, i.e., a generator of $\mathbf{K}^*$.

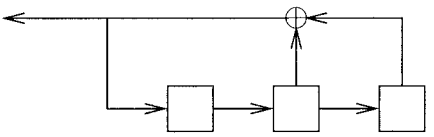


Figure 2.10. A circuit implementing the recurrence formula of the LFSR

Solution 10 ⋆Attacks on Cascade Ciphers

1 The time complexity is 2^ℓ. A cascade of L block ciphers can be viewed as a block cipher of key length $L \cdot \ell$ (as the L keys are independent), so that the time complexity would be $2^{L \cdot \ell}$.
When $L = 2$ the *meet-in-the-middle* attack reduces the time complexity from $2^{2\ell}$ down to $2 \cdot 2^\ell = 2^{\ell+1}$. In that case, the storage complexity is 2^ℓ.

2 As in Solution 6, we can prove that $\Pr[\mathsf{C}^*(x) = y] = 2^{-n}$. Assuming that E_K roughly behaves like a random permutation when K is randomly chosen among all possible wrong keys, we estimate $\Pr[\mathsf{E}_K(P) = C] \approx 2^{-n}$. Thus, the number of wrong keys displayed by the algorithm is approximately $2^\ell \cdot 2^{-n}$, that is $\mathcal{O}(2^{\ell-n})$. For a cascade cipher, a total of $\mathcal{O}(2^{L \cdot \ell - n})$ wrong keys are displayed.

3 Algorithm 7 exploits the t pairs at disposal. Considering that E_K roughly behaves like a random permutation when K is chosen among all possible wrong keys, we obtain

Algorithm 7 Exhaustive key search algorithm with t plaintext/ciphertext pairs

Input: t plaintext/ciphertext pairs (P_i, C_i), such that $C_i = \mathsf{E}_k(P_i)$, with $i = 1, \ldots, t$
Output: key candidate(s) for k
Processing:

```
for each possible key K do
    if C_i = E_K(P_i) for all i = 1, ..., t then
        display K
    end if
end for
```

$$\Pr[\mathsf{E}_K(P_i) = C_i \quad \text{for all } i = 1, \ldots, t] \approx \prod_{i=1}^{t} \Pr[\mathsf{E}_K(P_i) = C_i] \approx 2^{-tn} .$$

Table 2.5. Exhaustive key search on 3DES

t	1	2	3	4
Approx. number of wrong keys	$2 \cdot 10^{31}$	10^{12}	$6 \cdot 10^{-8}$	$3 \cdot 10^{-27}$

The number of wrong keys displayed by Algorithm 7 is thus $\mathcal{O}(2^{\ell - tn})$.

4 The number of wrong keys in this case is $\mathcal{O}(2^{L \cdot \ell - tn})$. For 3DES, $L = 3$, $\ell = 56$, and $n = 64$. The number of wrong keys displayed are given in Table 2.5 for different values of t. With 3 pairs, the adversary makes almost sure that only the good key is displayed.

More details about cascade ciphers and their security can be found in [29].

Solution 11 Attacks on Encryption Modes I

1 The inverse of the CBC|CFB mode is represented on Figure 2.11.

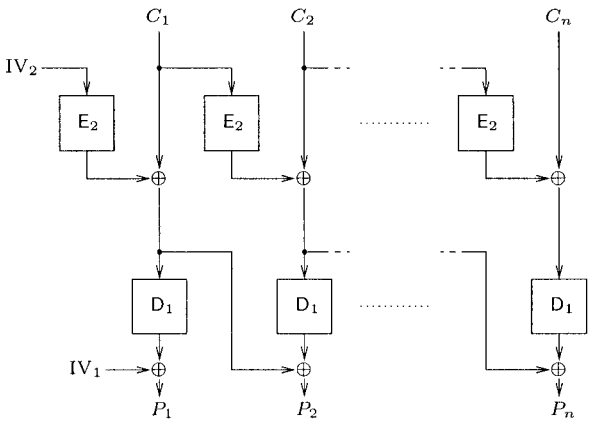


Figure 2.11. The inverse of the CBC|CFB mode

2 It can easily be checked that $\mathsf{D}_3(A'') \oplus \mathrm{IV} = C_1$ and that $\mathrm{IV} \oplus C_1 = A'' \oplus C_2$, so that

$$\mathsf{D}_3(\mathrm{IV} \oplus C_1 \oplus C_2) = C_1 \oplus \mathrm{IV} \,. \tag{2.4}$$

3 Algorithm 8 recovers k_3 with a time complexity of $\mathcal{O}(2^{\ell})$. As $n > \ell$,

Algorithm 8 Recovering k_3 in ECB|ECB|CBC^{-1} mode

Input: the initial vector IV and two ciphertext blocks C_1 and C_2
Output: key candidate(s) for k_3
Processing:

```
1: for each possible key K_3 do
2:   if Equation (2.4) holds then
3:     display K_3
4:   end if
5: end for
```

it does not yield any wrong key (with high probability). Once k_3 is found, the adversary can peel the third layer off, and do a meet-in-the-middle attack on the last two layers. Note that we typically need both plaintext blocks A and B in order to eliminate wrong key candidates during the meet-in-the-middle. The complexity of this part of the attack is $\mathcal{O}(2^\ell)$ in time and $\mathcal{O}(2^\ell)$ in storage. The complexity of the whole attack is $\mathcal{O}(2^\ell)$ in time, $\mathcal{O}(2^\ell)$ in storage, and we need 3 chosen plaintext blocks.

4 It can easily be checked that

$$\mathrm{IV}_2 \oplus \mathsf{E}_1(\mathrm{IV}_1) \oplus \mathsf{E}_1(\mathsf{E}_1(\mathrm{IV}_1)) \oplus P_1 \oplus P_2 = \mathsf{D}_3(A) \tag{2.5}$$

and that

$$\mathsf{E}_1(\mathsf{E}_1(\mathsf{E}_1(\mathrm{IV}_1))) \oplus \mathsf{E}_1(\mathsf{E}_1(\mathsf{E}_1(\mathsf{E}_1(\mathrm{IV}_1)))) \oplus P_3 \oplus P_4 = \mathsf{D}_3(A) \oplus \mathsf{D}_3(B). \tag{2.6}$$

5 Algorithm 9 uses a technique similar to a meet-in-the-middle attack in order to recover k_1 and k_3. The time complexity is $\mathcal{O}(2^\ell)$ and the storage complexity is $\mathcal{O}(2^\ell)$. As $n > \ell$ and as two equations have to hold before a key pair can be displayed, the algorithm does not yield any wrong key pair (with high probability). Once k_1 and k_3 are found, the adversary can peel off the first and third layers and perform a simple exhaustive search on k_2 in $\mathcal{O}(2^\ell)$. The overall complexity of the attack is $\mathcal{O}(2^\ell)$ in time, $\mathcal{O}(2^\ell)$ in storage, using four chosen ciphertext blocks.

A detailed study of cryptanalysis of multiple modes of operation can be found in [3, 4]. More recently, known-IV attacks against triple modes of operation were proposed in [20].

Algorithm 9 Recovering k_1 and k_3 in OFB|CBC|ECB mode with a meet-in-the-middle attack

Input: the initial vectors IV_1 and IV_2, the plaintext blocks P_1, P_2, P_3, and P_4, the two ciphertext blocks A and B

Output: key candidate(s) for k_1 and k_3

Processing:

1: **for** each possible key K_3 **do**
2: insert $(\mathsf{D}_3(A), \mathsf{D}_3(A) \oplus \mathsf{D}_3(B), K_3)$ in a table (keyed with the first entries)
3: **end for**
4: **for** each possible key K_1 **do**
5: **if** equations (2.5) and (2.6) hold **then**
6: display (K_1, K_3)
7: **end if**
8: **end for**

Solution 12 Attacks on Encryption Modes II

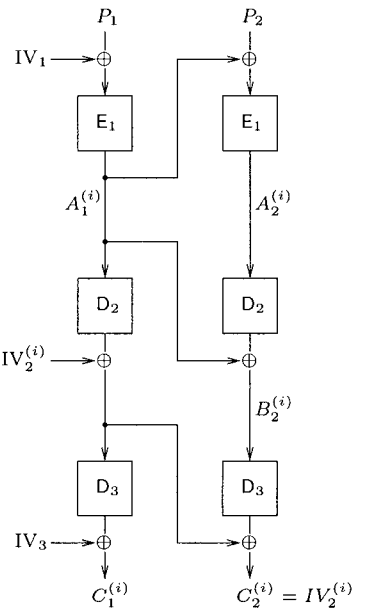


Figure 2.12. Collisions in CBC|CBC^{-1}|CBC^{-1} mode

1 The algorithm stops when a collision between two strings of n bits occurs. Therefore, its time complexity is $\mathcal{O}(2^{n/2})$.

2 We use the notations of Figure 2.12. We assume that $P_1^{(i)} = P_1^{(j)}$ for some $i \neq j$. As IV_1 is a constant, this implies that

$$A_1^{(i)} = A_1^{(j)}. \tag{2.7}$$

We also have

$$\begin{aligned} B_2^{(i)} &= \mathsf{E}_3\left(C_2^{(i)} \oplus \mathrm{IV}_2^{(i)} \oplus \mathsf{D}_2(A_1^{(i)})\right) \\ &= \mathsf{E}_3\left(\mathsf{D}_2(A_1^{(i)})\right) \qquad (\text{as } C_2^{(i)} = \mathrm{IV}_2^{(i)}) \end{aligned}$$

so that $B_2^{(i)} = B_2^{(j)}$ because of (2.7). Thus, by using (2.7) again, we obtain

$$A_2^{(i)} = A_2^{(j)}. \tag{2.8}$$

From (2.7) and from (2.8) we conclude that

$$P_2^{(i)} = P_2^{(j)}.$$

3 As IV_1 is constant,

$$\begin{aligned} P_1^{(i)} = P_1^{(j)} &\Leftrightarrow A_1^{(i)} = A_1^{(j)} \\ &\Leftrightarrow \mathrm{IV}_2^{(i)} \oplus \mathsf{E}_3(\mathrm{IV}_3 \oplus C_1^{(i)}) = \mathrm{IV}_2^{(j)} \oplus \mathsf{E}_3(\mathrm{IV}_3 \oplus C_1^{(j)}). \end{aligned} \tag{2.9}$$

4 Algorithm 10 recovers K_3 in 2^k time complexity. Once K_3 is found, the adversary can peel the third layer off and mount a meet-in-the-middle attack on the first two layers. The overall complexity of the

Algorithm 10 Recovering k_3 CBC|CBC^{-1}|CBC^{-1} mode

Input: $\mathrm{IV}_2^{(i)}$, $\mathrm{IV}_2^{(j)}$, IV_3, $C_1^{(i)}$, and $C_1^{(j)}$
Output: key candidate(s) for k_3
Processing:

```
for each possible key K3 do
    if Equation (2.9) holds then
        display K3
    end if
end for
```

attack is $\mathcal{O}(2^k)$ in time, $\mathcal{O}(2^k)$ in storage, and needs $\mathcal{O}(2^{n/2})$ chosen ciphertexts.

A detailed study of cryptanalysis of multiple modes of operation can be found in [3, 4]. More recently known-IV attacks against triple modes of operation were proposed in [20].

Solution 13 ⋆A Variant of A5/1 I

1 When R_1 is loaded with all zeros, no matter which subset of cells is chosen to compute the feedback, the feedback is always zero, hence, the next state of all zeros does not change. Of course, this also applies to R_2 and R_3.

2 When two LFSRs out of three are initialized by all zeros, we can view A5/1 equivalently as consisting of the remaining single LFSR, which outputs the leftmost bit at each clock pulse and is shifted if and only if its clocking tap is zero. Note that the clocking tap of a non-zero LFSR cannot always be zero. Thus, after a limited number of clock pulses, the clocking tap of the equivalent LFSR would be equal to 1 so that the LFSR will stop forever and output the same bit. So, as long as the non-zero LFSR outputs zero before (and when) its clocking tap turns to 1, A5/1 generates the all-zero keystream (including the special case of three all-zero LFSRs initially).

3 We consider the following four different cases:

- For $R_1 = R_2 = R_3 = \mathbf{0}$: There is only one (trivial) possibility.
- For $R_1 \neq \mathbf{0}$ and $R_2 = R_3 = \mathbf{0}$: If $R_1[8] = 1$, R_1 is never shifted. In that case, it is sufficient to also have $R_1[18] = 0$ to obtain a keystream with only zeros. This leaves $2^{19-2} = 2^{17}$ different initialization states. We can also consider the case where $R_1[8] = 0$ and $R_1[7] = 1$, so that R_1 will be shifted exactly once. Here, it is sufficient to have $R_1[18] = R_1[17] = 0$ to obtain a keystream with only zeros. This leaves $2^{19-4} = 2^{15}$ different initialization states. Following the same reasoning, we deduce the following lower bound on the number of possible initializations states in this case:

$$2^{17} + 2^{15} + 2^{13} + 2^{11} + 2^9 + 2^7 + 2^5 + 2^3 + 2 = \frac{2^{19} - 2}{3}.$$

- For $R_2 \neq \mathbf{0}$ and $R_1 = R_3 = \mathbf{0}$: We similarly obtain a lower bound equal to

$$2^{20} + 2^{18} + \cdots + 2^0 = \frac{2^{22} - 1}{3}.$$

- For $R_3 \neq \mathbf{0}$ and $R_1 = R_2 = \mathbf{0}$: We similarly obtain a lower bound equal to

$$2^{21} + 2^{19} + \cdots + 2^1 = \frac{2^{23} - 2}{3}.$$

Summing these values, we conclude that there are at least 2^{22} such initialization states.

4 When the initial clocking taps of the three LFSRs are all equal, none of the three LFSRs will ever be shifted. Hence, provided that the XOR of the three LFSRs output bits is zero at some time, we will obtain the all-zero keystream.

Alternatively, when one LFSR out of three is all-zero initially and the initial clocking taps of the other two LFSRs are both one, then only the all-zero LFSR is shifted (without changing its state however). It is actually shifted forever, while the remaining two LFSRs would stop forever. So, as long as the leftmost bits of two non-zero LFSRs are equal and the clocking taps are both one, the variant A5/1 generates the all-zero keystream.

5 We consider the following four different cases:

- Case where the three LFSRs all stop forever: we have $2^{64-2-1} = 2^{61}$ different initial states that satisfy two linear relations: one clocking constraint and one output constraint.
- For $R_1 = \mathbf{0}$: In this case, if $R_2[10] = R_3[10] = 1$ and $R_2[21] = R_3[22]$ we know that we obtain the all-zero keystream. There are $2^{22+23-3} = 2^{42}$ different initial states that satisfy these constraints.
- For $R_2 = \mathbf{0}$: Similarly, we find $2^{19+23-3} = 2^{39}$ different initial states that produce the all-zero keystream.
- For $R_3 = \mathbf{0}$: Similarly, we find $2^{19+22-3} = 2^{38}$ different initial states that produce the all-zero keystream.

Summing up these values, we obtain a lower bound between 2^{62} and 2^{63} on the number of possible initial states that produce the all-zero keystream.

6 Obviously, the assumption does not hold for this variant of A5/1.

7 A keystream generator should avoid generating the same keystream under several keys. These kind of keys are called "*weak* keys". Although we only computed lower bounds on the number of weak keys for both A5/1 and its variant, the huge difference between the two bounds (2^{22} for the real A5/1 against 2^{62} for its variant) suggests that the variant is much weaker.

Solution 14 ⋆A Variant of A5/1 II

1 Let T_i denote the value of the clocking tap of R_i just before it is clocked, for $i = 1, 2, 3$. We denote by $\mathrm{P}_i^{\text{shifted}}$ the probability that R_i is shifted at the next clock, and $\mathrm{P}_i^{\text{fixed}}$ the probability that it is not. By symmetry, it is sufficient to compute this probability for R_1. As R_1 is *not* shifted if and only if $T_1 \neq T_2 = T_3$, we have

$$\mathrm{P}_1^{\text{fixed}} = \frac{1}{2^3} \sum_{T_1,T_2,T_3} \mathbf{1}_{T_1 \neq T_2 = T_3} = \frac{1}{4}.$$

So that the probability that it is shifted is $\mathrm{P}_1^{\text{shifted}} = 1 - \mathrm{P}_1^{\text{fixed}} = \frac{3}{4}$. By symmetry, we obtain the same probabilities for R_2 and R_3, i.e.,

$$\mathrm{P}_i^{\text{shifted}} = \frac{3}{4} \quad \text{and} \quad \mathrm{P}_i^{\text{fixed}} = \frac{1}{4} \quad \text{for } i = 1, 2, 3.$$

2 Clearly, either 2 or 3 LFSRs are shifted at each clock. In other words, when one LFSR is fixed, the two others are shifted. The probability that exactly two LFSRs are shifted is thus equal to the probability that exactly one is fixed. This probability is simply equal to $P_1^{\text{fixed}} + P_2^{\text{fixed}} + P_3^{\text{fixed}} = \frac{3}{4}$ as the three events are disjoint.

3 We denote by $c^t \in \{0, 1, 2, 3\}$ the way the LFSRs are shifted at time t. More precisely, we denote by $c^t = 0$ the case where all three LFSRs are shifted, and by $c^t = i$ the case where R_i is fixed (the two others being necessarily shifted). From the previous questions, we immediately obtain

$$\Pr[c^0 = i] = \frac{1}{4} \quad \text{for } i = 0, 1, 2, 3.$$

4 If all LFSRs are shifted at time 0, we know that all three taps had the same value. But as we assumed that the cells of the LFSRs were drawn independently, this tells us nothing about c^1, and thus

$$\Pr[c^1 = c | c^0 = 0] = \Pr[c^1 = c] = \frac{1}{4}, \quad \text{for all } c \in \{0, 1, 2, 3\}.$$

When $c^0 \neq 0$, exactly two LFSRs are shifted. As the two new values of the clocking taps are uniformly distributed and independent random values, then we have no information whatsoever about the next majority value and hence, neither about c^1. Therefore,

$$\Pr[c^1 = c | c^0 \neq 0] = \Pr[c^1 = c] = \frac{1}{4}, \quad \text{for all } c \in \{0, 1, 2, 3\}.$$

We conclude that, for all $c, c' \in \{0,1,2,3\}$,

$$\Pr[c^1 = c | c^0 = c'] = \frac{1}{4},$$

which corresponds to a uniform distribution.

5 We consider the variant of A5/1. We first note that in this case, either exactly one LFSR is clocked (when its clocking tap is different from the two others) or no LFSR is clocked at all (when all three clocking taps are equal). Using the notations of Question 1, we have

$$\mathrm{P}_1^{\text{fixed}} = \Pr[T_2 \neq T_3] + \Pr[T_1 = T_2 = T_3] = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}.$$

Consequently, by symmetry,

$$\begin{aligned} \mathrm{P}_i^{\text{fixed}} &= \frac{3}{4} \quad \text{and} \\ \mathrm{P}_i^{\text{shifted}} &= 1 - \mathrm{P}_i^{\text{fixed}} = \frac{1}{4} \quad \text{for all } i = 1,2,3. \end{aligned}$$

The probability that all three LFSRs stay still during next clock is $\Pr[T_1 = T_2 = T_3] = \frac{1}{4}$, and the probability that exactly one LFSR is shifted is $P_1^{\text{shifted}} + P_2^{\text{shifted}} + P_3^{\text{shifted}} = \frac{3}{4}$.

We denote $c^t \in \{0,1,2,3\}$ the way the LFSRs are shifted at time t. This time, we denote by $c^t = 0$ the case where all three LFSRs stay still at time t, and by $c^t = i$ the case where R_i is clocked (the remaining two LFSRs staying necessarily still). We verify that $\Pr[c^0 = 0] = \frac{1}{4}$ and that $\Pr[c^0 = i] = \mathrm{P}_i^{\text{shifted}} = \frac{1}{4}$. Therefore, the distribution of c^0 is uniform.

Obviously, if no LFSR is shifted at time t, no LFSR will ever be shifted. Therefore $\Pr[c^1 = 0 | c^0 = 0] = 1$ and $\Pr[c^1 \neq 0 | c^0 = 0] = 0$. Moreover, if two taps have the same value at time t, the corresponding LFSRs will never be clocked (as they will never be in a minority). Therefore, letting $c \neq 0$, $\Pr[c^1 \notin \{0, c\} | c^0 = c] = 0$ and, by independence of the LFSRs cells,

$$\Pr[c^1 = c | c^0 = c] = \Pr[c^1 = 0 | c^0 = c] = \frac{1}{2}.$$

6 For the majority control, the conditional mass function is identical to the mass function, which means the next clocking and the current clocking are independent. We notice that this is definitely not the case for the minority control. In terms of entropy, we can see that

$H(c^1|c^0) = \frac{3}{4}$ (resp. 2), and $H(c^0) = 2$ (resp. 2) under minority (resp. majority) control. In other words, in the case of minority control, if we try to recover the initial state of the LFSRs by guessing the clocking sequence, then after guessing two bits for the first clocking, we only need to guess 3/4 bit every clock afterwards on average. In the case of majority control, the knowledge of the previous clocking tells us nothing about the next one. We conclude that the majority control (the actual one used in A5/1) is a better choice from the security point of view.

Solution 15 ⋆Memoryless Exhaustive Search

1 We first compute the expected complexity $\mathrm{E}[C]$ in the general case, i.e., without making any assumption about the distribution of K. As the queries are independent, the worst case complexity is infinite (e.g., the case where the algorithm always tries the same wrong key). We have by definition

$$\mathrm{E}[C] = \sum_{c=1}^{+\infty} c \Pr[C = c]. \tag{2.10}$$

Using the Total Probability Theorem, we have

$$\Pr[C = c] = \sum_{i=1}^{N} \Pr[C = c \mid K = k_i] \Pr[K = k_i]. \tag{2.11}$$

We can easily compute $\Pr[C = c \mid K = k_i]$ as it is the probability that the cryptanalyst chooses the right key after $(c-1)$ wrong guesses

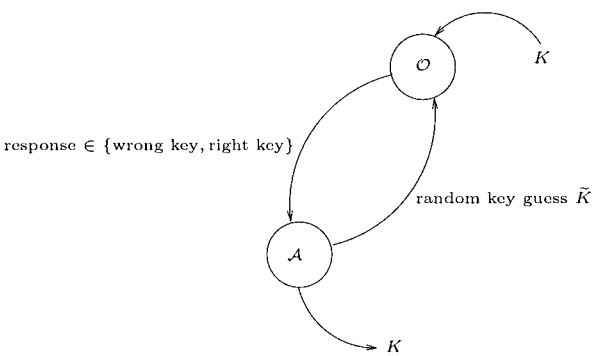


Figure 2.13. Adversary modeling a memoryless exhaustive search

(this is a geometrical distribution)

$$\Pr[C = c \mid K = k_i] = \Big(1 - \Pr[\widetilde{K} = k_i]\Big)^{c-1} \Pr[\widetilde{K} = k_i], \qquad (2.12)$$

where $\widetilde{K}$ denotes the key chosen by the cryptanalyst. From (2.10), (2.11), and (2.12) we deduce

$$\begin{aligned}
\mathrm{E}[C] &= \sum_{c=1}^{+\infty} c \sum_{i=1}^{N} \Big(1 - \Pr[\widetilde{K} = k_i]\Big)^{c-1} \Pr[\widetilde{K} = k_i] \Pr[K = k_i] \\
&= \sum_{i=1}^{N} \Pr[\widetilde{K} = k_i] \Pr[K = k_i] \underbrace{\sum_{c=1}^{+\infty} c \Big(1 - \Pr[\widetilde{K} = k_i]\Big)^{c-1}}_{=\Pr[\widetilde{K}=k_i]^{-2} \text{ as shown below}} \\
&= \sum_{i=1}^{N} \frac{\Pr[K = k_i]}{\Pr[\widetilde{K} = k_i]}.
\end{aligned}$$

Note that we needed a classical result, namely that we have

$$\sum_{c=1}^{+\infty} c x^{c-1} = \frac{d}{dx} \left(\sum_{c=0}^{+\infty} x^c \right) = \frac{d}{dx} \frac{1}{1-x} = \frac{1}{(1-x)^2},$$

when x is a real value such that $|x| < 1$. In the particular case where the key distribution is uniform, we have

$$\Pr[K = k_i] = \frac{1}{N} \quad \text{for all } i \in \{1, \ldots, N\},$$

so that

$$\mathrm{E}[C] = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\Pr[\widetilde{K} = k_i]}.$$

This is minimal when all the $\Pr[\widetilde{K} = k_i]$ are equal, and in this case

$$\mathrm{E}[C] = N.$$

As this algorithm is memoryless, the same wrong key can be queried twice. In order to improve the algorithm, one can use a memory to remember previous queries. This is called an *exhaustive search.* In that situation, we would obtain an average complexity

$$\mathrm{E}[C] = \frac{N-1}{2}.$$

2 We go back to the general case where K does not necessarily follow the uniform distribution. The cryptanalyst wants to minimize

$$\mathrm{E}[C] = \sum_{i=1}^{N} \frac{\Pr[K = k_i]}{\Pr[\widetilde{K} = k_i]}.$$

We set $p_i = \Pr[K = k_i]$ (which are considered to be a fixed values, as they cannot be chosen by the cryptanalyst, but are only known to him) and $Q_i = \Pr[\widetilde{K} = k_i]$ (which are N real variables). The Q_i's can be chosen by the adversary, but still have to sum to 1 (as they correspond to a probability distribution). Therefore, we must compute

$$\begin{cases} \min\limits_{\{Q_1,\ldots,Q_N\}} \mathrm{E}[C] = \sum\limits_{i=1}^{N} \dfrac{p_i}{Q_i} \\ \text{s.t.} \quad \sum\limits_{i=1}^{N} Q_i = 1. \end{cases}$$

In order to compute this, we use the theory of the Lagrange Multipliers. Let Φ be defined by

$$\Phi(Q_1, Q_2, \ldots, Q_N) = \sum_{i=1}^{N} \frac{p_i}{Q_i} + \lambda \left(\sum_{i=1}^{N} Q_i - 1 \right),$$

where λ is the Lagrange multiplier. If $(q_1, \ldots, q_N)$ is an extremum, it must satisfy

$$\begin{cases} \sum\limits_{i=1}^{N} q_i = 1 \\ \\ \dfrac{\partial \Phi}{\partial Q_j}(q_1, \ldots, q_N) = 0 \quad \text{for all } j \in \{1, \ldots, N\}, \end{cases} \tag{2.13}$$

that is

$$(2.13) \quad \Leftrightarrow \quad \begin{cases} \sum_{i=1}^{N} q_i = 1 \\ \\ \lambda = \dfrac{p_j}{q_j^2} \quad \text{for all} \quad j \in \{1, \ldots, N\} \end{cases}$$

$$\Leftrightarrow \quad \begin{cases} \sum_{i=1}^{N} q_i = 1 \\ \\ q_j = q_d \sqrt{\dfrac{p_j}{p_d}} \quad \text{for all} \quad d, j \in \{1, \ldots, N\} \end{cases}$$

so we obtain, for all $d \in \{1, \ldots, N\}$,

$$q_d = \frac{1}{\sum_{j=1}^{N} \sqrt{p_j/p_d}} = \sqrt{\frac{p_d}{H_{\frac{1}{2}}(K)}}. \tag{2.14}$$

The best strategy for the cryptanalyst is therefore to draw the queries according to the distribution defined by (2.14). In that case, the average complexity is

$$\begin{aligned} \mathrm{E}[C] &= \sum_{i=1}^{N} p_i \sqrt{\frac{H_{\frac{1}{2}}(K)}{p_i}} \\ &= H_{\frac{1}{2}}(K). \end{aligned}$$