Cryptology Part II: Bitblock Ciphers

Klaus Pommerening
Fachbereich Physik, Mathematik, Informatik
der Johannes-Gutenberg-Universität
Saarstraße 21
D-55099 Mainz

April 7, 1997—English version August 14, 2014
last change March 6, 2021

For the encryption of computer data we need algorithms that operate on bitblocks, are fast—and thereby suited for large data sets and high transfer speeds—and optimally resist sophisticated cryptanalysts. Therefore subjects of Part II of these cryptology lecture notes are:

- Construction principles for bitblock ciphers
  - Product ciphers, SP networks, FEISTEL networks, nonlinearity
- Example ciphers of special relevance
  - LUCIFER, DES, AES
- The most important cryptanalytic approaches
  - Algebraic, linear, and differential cryptanalysis

The mathematical appendix C gives a gentle introduction to Boolean functions and Boolean maps for beginners that might not feel comfortable with the algebraic treatment of Boolean algebra.

The mathematical appendix under the title "Fourier Analysis of Boolean Maps" focusses on nonlinearity: How to recognize, measure, and prevent unwanted linearity in encryption functions or in their building blocks.

Our goals are:

1. Describe the general mathematical framework for defining encryption functions on bitblocks, that is on vectors in a vector space $\mathbb{F}_2^l$ over the two element field $\mathbb{F}_2$.
2. Develop some criteria that help in assessing the strength of encryption functions.

Chapter 1

Composition of Ciphers

A first approach to constructing strong ciphers is the composition of several simple transformation steps. We saw some examples of this approach already in classical cryptography, and we saw how this often resulted in much stronger ciphers. But this effect is not guaranteed, and does not crop up in all cases. In this section we consider some basic aspects of this approach that is an essential ingredient of the construction of strong bitblock ciphers.

Terminologically we distinguish between

Multiple ciphers: combinations of instances of the same encryption function but with different keys.

Cascades: combinations of different encryption functions (also called product ciphers).

1.1 Multiple Ciphers and Group Structures

Multiple Ciphers

Let $F = (f_k)_{k \in K}$ be a cipher over the alphabet Σ, where $f_k\colon \Sigma^* \longrightarrow \Sigma^*$ is the encryption function corresponding to the key $k \in K$. The set of all of encryption functions is denoted by

$$\tilde{F} = \{f_k \mid k \in K\} \subseteq \mathrm{Map}(\Sigma^*, \Sigma^*).$$

By forming the **double cipher**

$$F^{(2)} = (f_h \circ f_k)_{h,k \in K}$$

the key space is significantly enlarged from K to $K \times K$. In the same way we can construct the triple cipher $F^{(3)}$, ..., the n-fold cipher $F^{(n)}$. All this makes sense only when

(A) $\tilde{F}$ is not a semigroup.

If $\tilde{F}$ *is* a semigroup, then for each pair of keys $h, k \in K$ there exists a key $x \in K$ such that $f_h \circ f_k = f_x$, and we don't get any new encryption functions by this kind of composition—a typical case of an "illusory complication", the effective keysize didn't increase at all!

We observe an even better effect when

(B) $\tilde{F}$ generates a subsemigroup of $\mathrm{Map}(\Sigma^*, \Sigma^*)$ of large size.

And the best we can hope for is:

(C) The map $K \times K \longrightarrow \widetilde{F^{(2)}} \subseteq \mathrm{Map}(\Sigma^*, \Sigma^*)$ is injective.

For a finite key space K we can express this also in the form:

(C') $\#\widetilde{F^{(2)}} = \#\{f_h \circ f_k \mid h, k \in K\} = (\#K)^2$.

The Group Property of a Block Cipher

A block cipher is uniquely characterized by its effect on Σ^r for a given exponent r, the blocksize. (For the moment we don't care about continuing it to strings of arbitrary lengths or about "padding" shorter strings to full blocklength.)

A block cipher **preserves lengths** if it transforms Σ^r to itself. Then in a canonical way $\tilde{F}$ is a subset of the symmetric group $\mathcal{S}(\Sigma^r)$, hence finite, and without restriction we may assume that also the keyspace K is finite. For such block ciphers the semigroup property (the converse of (A) above) is equivalent with the group property. This follows from the well-known simple lemma:

Lemma 1 *Let G be a finite group, $H \leq G$ a subsemigroup, that is $H \neq \emptyset$ and $HH \subseteq H$. Then H is a group, in particular $\mathbf{1} \in H$.*

Proof. Each $g \in G$ has finite order, $g^m = \mathbf{1}$ for some m. If $g \in H$, then $\mathbf{1} = g^m \in H$, and $g^{-1} = g^{m-1} \in H$. $\diamond$

This proves:

Proposition 1 *Let F be a length preserving block cipher over a finite alphabet. Then the following statements are equivalent:*

(i) *For any two keys $h, k \in K$ there exists an $x \in K$ such that $f_h \circ f_k = f_x$.*

(ii) *The set $\tilde{F}$ of encryption functions is a group.*

Remark

The probability that two random elements of the symmetric group $\mathcal{S}_n$ generate the whole group $\mathcal{S}_n$ or at least the alternating group $\mathcal{A}_n$ is

$$> 1 - \frac{2}{(\ln \ln n)^2} \quad \text{for large } n.$$

Source: John Dixon, *The probability of generating the symmetric group.* Mathematische Zeitschrift 110 (1969), 199–205.

For $n = 2^{64}$, a typical size for a block cipher, this lower bound is ≈ 0.86. With high probability it should generate the full or at least the "half" permutation group on the blocks. The concrete proof however might be difficult. One would try to determine the order of some concrete encryption functions by their effect on certain concrete messages, and then take the lowest common multiple as a lower bound for the group order.

In any case it seems that in general a multiple cipher is stronger than the underlying simple cipher. We'll discuss this again in Sections 1.3 and 1.4.

1.2 Examples of Multiple Ciphers

Examples of Groups

Each of the following length preserving ciphers forms a group:

- The shift ciphers over Σ with respect to a group structure on Σ
- The monoalphabetic substitutions over Σ
- The BELLASO ciphers with a fixed period
- The block transpositions of a fixed length

DES

DES is a block cipher on $\mathbb{F}_2^{64}$ with keyspace $\mathbb{F}_2^{56}$. CAMPBELL and WIENER in (CRYPTO 92) proved that DES generates the alternating group of order 2^{64}. Shortly before COPPERSMITH had shown that the group order is at least 10^{277}. Only much later someone noted that MOORE and SIMMONS in CRYPTO 86 had published the lengths of several cycles that would have sufficed to show that DES is not a group—a fact that for several years was viewed as an open conjecture.

Historical Examples

The composition of a polyalphabetic cipher of period l and another one of period q has period $\mathrm{lcm}(l, q)$. **Application:** Key generating machines as mentioned in Part I, see the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/4_Cylinder/LongPeriods.html`.

Another historical example: the double columnar transposition that is considerably stronger than the simple columnar transposition.

Composition of BELLASO Cipher

The composition of two BELLASO ciphers of periods l and q has period $\mathrm{lcm}(l, q)$, essentially the product lq. However its security amounts at most to the sum $l + q$ in view of an attack with known plaintext:

Assume known plaintext of length $l + q$ (over the alphabet $\mathbb{Z}/n\mathbb{Z}$). This yields $l + q$ linear equations for $l + q$ unknows—the characters of the two keys. Assume that $l < q$. Then the situation is

Plaintext	a_0	a_1	$\ldots$	a_{l-1}	a_l	$\ldots$	a_{q-1}	$\ldots$
Key 1	h_0	h_1	$\ldots$	h_{l-1}	h_0	$\ldots$	$\ldots$	$\ldots$
Key 2	k_0	k_1	$\ldots$	k_{l-1}	k_l	$\ldots$	k_{q-1}	$\ldots$
Ciphertext	c_0	c_1	$\ldots$	c_{l-1}	c_l	$\ldots$	c_{q-1}	$\ldots$

Taken together this is a BELLASO cipher with key

$$(h_0 + k_0, h_1 + k_1, \ldots)$$

and period $\operatorname{lcm}(l, q)$.

Let the known plaintext be $(a_0, \ldots, a_{l+q-1})$. Then the system of linear equations for the $l + q$ unknowns $h_0, \ldots, h_{l-1}, k_0, \ldots, k_{q-1} \in \mathbb{Z}/n\mathbb{Z}$ is:

$$\begin{array}{rcl} h_0 + k_0 & = & c_0 - a_0, \\ h_1 + k_1 & = & c_1 - a_1, \\ & \vdots & \\ h_{l-1} + k_{l-1} & = & c_{l-1} - a_{l-1}, \\ h_0 + k_l & = & c_l - a_l, \\ & \vdots & \\ h_{l+q-1 \bmod l} + k_{l+q-1 \bmod q} & = & c_{l+q-1} - a_{l+q-1}. \end{array}$$

This cannot have a unique solution: If we add a fixed value x to all h_i, and subtract x from all k_j, then we get another solution. Therefore for simplicity we may assume $h_0 = 0$. If the keys are not randomly chosen but built from keywords, then a simple “CAESAR exhaustion” will reveal the “true” keys later. For decryption the shifted keys are equivalent. And since we eliminated one unknown quantity, in general even $l + q - 1$ known plaintext letters are enough for uniquely solving the remaining $l + q - 1$ equations. We won't go into the details but give an exercise for interested readers.

Exercise

Consider the ciphertext

```
CIFRX KSYCI IDJZP TINUV GGKBD CWWBF CGWBC UXSNJ LJFMC
LQAZV TRLFK CPGYK MRUHO UZCIM NEOPP LK
```

For an attack with known plaintext assume that

- the plaintext (is in German and) starts with “Sehr geehrter ...” (a common beginning of a letter)
- some keylengths are already ruled out by trial & error; the actual lengths to test for a double BELASO cipher are $42 = 6 \times 7$.

(A coincidence analysis, even if it doesn't give enough confidence in a definite period, should suffice to exclude all but a few combinations of possible keylengths.)

1.3 Cryptanalysis of Double Ciphers

Meet in the Middle

The name of this attack against double encryption goes back to MERKLE and HELLMAN in 1981. (Don't confuse it with the "Man in the Middle" attack against cryptographic protocols.) They formalized an attack that worked in "classical times" against rotor machines, see the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/5_Rotor/AnalRot.html`.

Consider the composition of two encryption functions with different keys:

$$\begin{array}{ccccc} \Sigma^* & \xrightarrow{f_k} & \Sigma^* & \xrightarrow{f_h} & \Sigma^* \\ a & \mapsto & b & \mapsto & c. \end{array}$$

Assume a pair (a, c) of corresponding plaintext and ciphertext is known, and assume that the exhaustion of the simple cipher is feasible. Then the attacker builds two tables:

- all $f_k(a)$, $k \in K$,
- all $f_h^{-1}(c)$, $h \in K$,

and compares them. Each coincidence yields a possible pair $(h, k) \in K^2$ of keys that can be further inspected, say with further known (or guessed) plaintext.

Expenses

This attack needs

- $2 \cdot \#K$ encryptions (*not* $(\#K)^2$),
- $2 \cdot \#K$ memory cells.

Noting that we need only store one of the two tables we even halve the number of memory cells.

With the usual prefixes for memory sizes

2^{10}	2^{20}	2^{30}	2^{40}	2^{50}	2^{60}
Kilo	Mega	Giga	Tera	Peta	Exa

and using 1 byte = 8 bits we see that 60 bit keys need memory that exceeds the (actually) available capacities. However for cryptanalysis the time requirements are more critical than memory requirements. Therefore as a general finding we may state:

> *The security of a double cipher is not significantly better than the security of the underlying simple cipher.* In particular the bitlength of a key exhaustion is not doubled but only increased by 1 bit.

False Alarms

One question yet awaits an answer: How many of the coincidences in comparing the two tables lead to a wrong pair of suspected keys? That is, how likely are false alarms?

Here is a heuristic consideration: Assume we encrypt n-bit blocks with l-bit keys. Then the tables have 2^l entries, resulting in 2^{2l} comparisions. Since the number of possible values is 2^n we expect about $N_1 = 2^{2l-n}$ coincidences. (Implicitly assuming that the values behave like random. By the Birthday Paradox we expect the first coincidence after $2^{n/2}$ trials, but this is irrelevant in the present context.)

If we test the pitched key pairs with a second known plaintext block, then we are left with $N_2 = N_1/2^n = 2^{2l-2n}$ candidates. After testing t known plaintext blocks we expect to keep $N_t = 2^{2l-tn}$ candidates—but of course at least one, the right one.

Thus in general we find a unique solution as soon as

$$t \geq \frac{2l}{n}.$$

Examples

1. DES, $n = 64$, $l = 56$: $N_1 = 2^{48}$, $N_2 = 2^{-16}$. *We need about* 2 *blocks of known plaintext.*

2. IDEA, $n = 64$, $l = 128$: $N_1 = 2^{192}$, $N_2 = 2^{128}$, $N_3 = 2^{64}$, $N_4 = 1$. *We need about* 4 *blocks.*

3. AES, $n = 128$, $l = 128$: $N_1 = 2^{128}$, $N_2 = 1$. *We need about* 2 *blocks.* But the number $\#K = 2^{128}$ will by far exceed our time and memory resources (as in Example 2).

Time-Memory-Tradeoff

A more general consideration yields a “Time Memory Tradeoff”: Undertaking a Meet in the Middle attack we may spare memory, allowing more execution time, by generating only partial tables:

If during a pass we fix s bits of both h and k, then we need 2^{l-s} memory cells for both of the tables of $f_k(a)$’s and $f_h^{-1}(c)$’s. As a compensation we have to go through 2^{2s} passes. The expenses are:

$2 \cdot 2^{l-s}$	encryptions for building one pair of tables,
2^{2s}	comparisions of one pair of tables, in total
$2 \cdot 2^{l+s}$	encryptions,
$2 \cdot 2^{l-s}$	memory cells.

Multiplying the number of encryptions and the number of needed memory cells we get $4 \cdot 2^{2l}$, independently from s. *This gives the attacker some freedom in using her resources in a flexible way.*

Example DES: If the attacker owns 128 terabytes of memory, she can generate 2 tables of 2^{40} blocks each, hence choose $s = 56 - 40 = 16$. Then she needs $2 \cdot 2^{72}$ encryptions. This is feasible, at least for the world's largest secret service.

Summary

Double ciphers don't improve the security of encryption in a worthwile way.

1.4 Triple Ciphers

The last section unveiled a principal weakness of double encryption. Therefore, to get a real improvement, we move on to triple encryption. An often used scheme is “EDE” (Encryption, Decryption, Encryption)

$$f_g \circ f_h^{-1} \circ f_k \quad \text{for } g, h, k \in K.$$

Why is f_h inverted? The advantage of this scheme is its compatibility with simple encryption by choosing keys $g = h = k$.

The Meet in the Middle attack also applies to this scheme. Thus the effective key length (for exhaustion) is not tripled but only doubled, but that’s OK for 56 or 64-bit keys.

Often the scheme is somewhat simplified as “two-key triple encryption”:

$$f = f_k \circ f_h^{-1} \circ f_k \quad \text{for } h, k \in K.$$

This scheme has a weakness under an attack with *chosen* plaintext that however worries only paranoiacs. Consider the scenario

$$\begin{array}{ccccccc} \Sigma^* & \xrightarrow{f_k} & \Sigma^* & \xrightarrow{f_h^{-1}} & \Sigma^* & \xrightarrow{f_k} & \Sigma^*, \\ a & \mapsto & b & \mapsto & b' & \mapsto & c. \end{array}$$

Step 1: Using $\#K$ encryptions and $\#K$ memory cells precalculate the table

$$\{f_h^{-1}(b_0) \mid h \in K\}$$

for a fixed intermediate value of b_0.

Step 2: Then calculate for all keys $k \in K$:

$$\begin{aligned} a_k &:= f_k^{-1}(b_0) \quad \text{(the chosen plaintext)}, \\ c_k &:= f(a_k), \\ b_k &:= f_k^{-1}(c_k). \end{aligned}$$

The second assignment is possible in an attack with chosen plaintext, which implies that we can evaluate f with any plaintexts. The expenses are $5 \cdot \#K$ simple encryptions. If $b_k = f_h^{-1}(b_0)$, then we keep the pair (h, k) of keys for further examination.

$$\begin{tikzcd}
\Sigma^* \arrow[r, "f_k"] & \Sigma^* \arrow[r, "f_h^{-1}"] & \Sigma^* \arrow[r, "f_k"] & \Sigma^* \\
& b_0 \arrow[r, mapsto, "f_h^{-1}"] \arrow[dl, "f_k^{-1}"'] & \textcircled{b_k} \arrow[dl, dash] & \\
a_k \arrow[rrr, mapsto, "f"'] & f_h^{-1}(b_0) \overset{\textcircled{?}}{=} f_k^{-1}(c_k) & & c_k \arrow[ul, "f_k^{-1}"']
\end{tikzcd}$$

The most efficient known attack is described in:

- Van Oorschot/Wiener: A known plaintext attack on two-key triple encryption. Eurocrypt 90.

1.5 Cascades of Different Ciphers

Examples

1. Monoalphabetic substitutions and transpositions commute. Combining more than one of each doesn't make sense since each of these two types forms a group. Composing one monoalphabetic substitution and one (simple) transposition makes a weak cipher. Solving it by a ciphertext only attack starts with a frequency count that reveals the most common letters.

2. The same remark applies to periodic polyalphabetic ciphers and transpositions. But if we take different period lengths for each step we get a fairly complex cipher, however it is too complex for manual operation.

3. The Enigma composed a monoalphabetic cipher with several polyalphabetic substitutions of different periods, followed by one more monoalphabetic substitution. The result was a single polyalphabetic substitution with a very large period.

4. The ADFGVX cipher used by the German army in WW I consisted of a substitution followed by a columnar transposition. For the substitution the 26 letters and 10 digits were distributed into a 6-by-6 square in an order defined by the key. Then each character was replaced by its coordinates in this square that were denoted by A, D, F, G, V, X. The French (PAINVIN und GIVIERGE) had many successes in breaking this cipher.

5. Composing a monoalphabetic cipher with an autokey cipher is one of the “modes” that make block ciphers a little bit harder, see Chapter 3.

6. Finally recall that PORTA's disk cipher had a representation as composition of a monoalphabetic substitution with a BELLASO (aka VIGENÈRE) cipher.

As a résumé we may state that cascades of different ciphers in general increase the security, but not always. In any case the situation requires a careful analysis before we trust a newly constructed product cipher.

Chapter 2

Bitblock Ciphers and Feistel Networks

This chapter contains basic facts about bitblock ciphers and some approaches to their construction.

2.1 Bitblock Ciphers—Introduction

Description

Bitblock ciphers operate over the alphabet $\Sigma = \mathbb{F}_2 = \{0, 1\}$, and basically encrypt blocks of fixed length conserving this length, controlled by a key that itself is a bitblock of a certain length l. The encryption functions are defined as maps of the set $\mathbb{F}_2^n$ into itself, and the set $\mathbb{F}_2^l$ serves as key space.

For constructing and analyzing bitblock ciphers we usually view $\mathbb{F}_2^n$ as a vector space of dimension n over the two-element field $\mathbb{F}_2$. Sometimes we equip $\mathbb{F}_2^n$ with the structure of the field $\mathbb{F}_{2^n}$, on rare occasions we structure it as cyclic group of order 2^n, thinking of integer addition "with carry" $\bmod 2^n$.

Thus we describe a bitblock cipher as a map

$$F\colon \mathbb{F}_2^n \times \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^n$$

or as a family $(F_k)_{k\in K}$ of maps

$$F_k\colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n \quad \text{for } k \in K = \mathbb{F}_2^l$$

where $F_k(a) = F(a, k)$.

Note In this chapter the mathematical symbol n is used for the length of the bitblocks, not for the size of the alphabet.

We might also view a bitblock cipher as a monoalphabetic substitution over the alphabet $\Sigma' = \mathbb{F}_2^n$.

The extension of a cipher to bitstrings of arbitrary lengths is subject of Chapter 3 on "modes", and is of no concern for the moment, and likewise we don't care how to pad shorter bitstrings.

Choice of the Block Length

The block length should be large enough to preclude the methods that break monoalphabetic substitutions, in particular analyses of patterns and frequencies. Moreover we would like to avoid any kind of leaks that reveal information about the plaintext, for example repeated ciphertext blocks.

If the sender didn't systematically prevent repetitions, an attacker could mount a **codebook attack** by collecting pairs of ciphertext and known plaintext for a fixed (but unknown) key. In this way she would construct her own codebook. A large codebook would allow breaking many future messages even if it didn't reveal the key. To prevent this attack we require:

- $\#\Sigma' = 2^n$ should be larger than the number of available memory cells, even assuming a very powerful attacker.

- Keys should be changed quite regularly.

In view of the Birthday Paradox an even stronger criterion is adequate: If the attacker has collected in her codebook about $\sqrt{\#\Sigma'} = 2^{n/2}$ plaintext-ciphertext pairs, the probability of a "collision" is approximately $\frac{1}{2}$. Therefore we require that the number $2^{n/2}$ surpasses the available storage. And keys should be changed long before this number of blocks is encrypted.

In the "pre-AES" age bitblock ciphers usually had 64-bit blocks. From our point of view this is by far insufficient, at best justified by frequent key changes. We prefer 128 bits as block length. This is also the block length of the new standard AES.

This consideration might look somewhat paranoid. But it is a typical example of the security measures in modern cryptography: The cipher designers work with large security margins and avoid any weaknesses even far away from a practical use by an attacker. Thus the security requirements of modern cryptography by far surpass the requirements typical for classical cryptography. This may sound exaggerated. But the modern algorithms—that in fact offer these huge security margins—can also resist future progress of cryptanalytic capabilities.

2.2 Polynomials over Finite Fields

In this section bitblock cryptography is "reduced" to algebra with polynomials.

Let K be a field. Given a polynomial $\varphi \in K[T_1, \ldots, T_n]$ in n indeterminates $T_1, \ldots, T_n$, we define a function $F_\varphi\colon K^n \longrightarrow K$ by evaluating the polynomial φ at n-tuples $(x_1, \ldots, x_n) \in K^n$,

$$F_\varphi(x_1, \ldots, x_n) := \varphi(x_1, \ldots, x_n).$$

Note that we carefully distinguish between polynomials and polynomial functions. Polynomials are elements of the polynomial ring $K[T_1, \ldots, T_n]$ where the elements T_i—the "indeterminates"—are a set of algebraically independent elements. That means that the infinitely many monomials $T_1^{e_1} \cdots T_n^{e_n}$ are linearly independent over K.

In general (for infinite fields) there are many more ("non-polynomial") functions on K^n. But not so for finite fields—in other words, over a finite field all functions are polynoms:

Theorem 1 *Let K be a finite field with q elements, and $n \in \mathbb{N}$. Then every function $F\colon K^n \longrightarrow K$ is given by a polynomial $\varphi \in K[T_1, \ldots, T_n]$ of partial degree $\leq q-1$ in each T_i.*

The proof of Theorem 1 is in Appendix B, a more elementary proof for the case $K = \mathbb{F}_2$ is in Appendix C.

Corollary 1 *Let $m, n \in \mathbb{N}$. Then every map $F : K^n \longrightarrow K^m$ is given by an m-tuple $(\varphi_1, \ldots, \varphi_m)$ of polynomials $\varphi_i \in K[T_1, \ldots, T_n]$ of partial degree $\leq q-1$ in each T_i.*

Corollary 2 *Every map $F\colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^m$ is given by an m-tuple $(\varphi_1, \ldots, \varphi_m)$ of polynomials $\varphi_i \in \mathbb{F}_2[T_1, \ldots, T_n]$ all of whose partial degrees are ≤ 1.*

From this the **algebraic normal form (ANF)** of a Boolean function $F : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ derives: For a subset $I = \{i_1, \ldots, i_r\} \subseteq \{1, \ldots, n\}$ let x^I be the monomial

$$x^I = x_{i_1} \cdots x_{i_r}.$$

Then F has a unique representation as

$$F(x_1, \ldots, x_n) = \prod_I a_I x^I \quad \text{for all } x = (x_1, \ldots, x_n) \in K^n \text{ where } a_I = 0 \text{ or } 1.$$

In particular the 2^n monomial functions $x \mapsto x^I$ constitute a basis of the vector space $\mathrm{Map}(\mathbb{F}_2^n, \mathbb{F}_2)$ over $\mathbb{F}_2$, and the number of these functions is 2^{2^n}.

2.3 Algebraic Cryptanalysis

Attacks with Known Plaintext

Consider a bitblock cipher, given by the map

$$F\colon \mathbb{F}_2^n \times \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^n$$

Then F is an n-tuple $F = (F_1, \ldots, F_n)$ of polynomial functions in $n + l$ variables all of whose partial degrees are ≤ 1.

An attack with known plaintext $a \in \mathbb{F}_2^n$ and corresponding ciphertext $c \in \mathbb{F}_2^n$ leads to a system

$$F(a, x) = c$$

of n polynomial equations for the unknown key $x \in \mathbb{F}_2^l$.

Systems of polynomial equations (over arbitrary fields) are one of the subjects of algebraic geometry. A rule of thumb says

The solution set for x has dimension 0 *"in general" for $n \geq l$.*

(I. e. it consists of a few isolated solutions. Otherwise, if the solution set allows for free parameters—or has dimension $\geq$ 1—, the attacker needs some more blocks of known plaintext.)

The general theory of polynomial equations is quite deep, in particular if we search for concrete solution procedures. But maybe the observation that only partial degrees ≤ 1 occur makes a difference?

Examples

Example 1: Let $n = l = 2$,

$$F(T_1, T_2, X_1, X_2) = (T_1 + T_2X_1, T_2 + T_1X_2 + X_1X_2),$$

$a = (0, 1)$, $c = (1, 1) \in \mathbb{F}_2^2$. Then the system of equations for the key $(x_1, x_2) \in \mathbb{F}_2^2$ is

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 + x_1 \\ 1 + 0 + x_1x_2 \end{pmatrix}.$$

The obvious solution is $x_1 = 1$, $x_2 = 0$.

Example 2, linear maps: If F is a *linear* map, then the system of equations has an efficient solution by the methods of linear algebra (n linear equations in l unknowns). For this method to work F needs to be linear only in x.

Example 3, substitution: The complexity (or simplicity) of a polynomial equation is not always clear at first sight. Here is an example (over $\mathbb{F}_2$):

$$x_1x_2x_3 + x_1x_2 + x_1x_3 + x_2x_3 + x_2 + x_3 = 0.$$

The substitutions $x_i = u_i + 1$ transform it to

$$u_1u_2u_3 + u_1 = 0$$

(for an easy proof look in the reverse direction). The solutions are

$$u_1 = 0,\ u_2, u_3 \text{ arbitrary } \textit{or } u_1 = u_2 = u_3 = 1.$$

Thus the complete solution of the original equation is

$$x_1 = 1,\ x_2, x_3 \text{ arbitrary } \textit{or } x_1 = x_2 = x_3 = 0.$$

The Complexity of the Algebraic Attack

In the examples the solutions were easily found. But in general this task is too complex.

There are two powerful general approaches for solving systems of (polynomial) equations over $\mathbb{F}_2$:

- SAT solvers. SAT denotes the satisfiability problem of propositional logic. Consider a logical expression in Boolean variables $x_1, \ldots, x_n$ and ask if there exist values of the variables that make the expression "True". In other words consider a Boolean function f and ask if it assumes the value 1. A SAT solver is an algorithm that takes a logical expression and decides the satisfiability by finding a solution x, or showing there's no solution. The naive algorithm uses the truth table and exhausts the 2^n possible arguments. However there are much faster algorithms, the most popular being the DPLL algorithm (after Davis, Putnam, Logemann, and Loveland) and BDD based algorithms (Binary Decision Diagram).
- Elimination using Groebner bases.

Both methods work well for a small number of unknowns. With a growing number of unknowns their complexity becomes unmanageable. Of course we always find a solution by searching through the complete value table. But this naive method is inefficient (exponential in the number of unknowns, hopeless for 80 or more unknowns). But also the costs of SAT solvers and Groebner-basis methods grow exponentially with the number of unknowns. Not even the fact that all partial degrees are ≤ 1 is of vital help. The basic resault is:

Theorem 2 (Garey/Johnson) *The problem of finding a common zero of a system of polynoms $f_1, \ldots, f_r \in \mathbb{F}_2[T_1, \ldots T_n]$ is* **NP***-complete.*

Proof. See [2]. $\diamond$

What "**NP**-complete" means will be answered later in this lecture (see Part III, Chapter 6). In fact SAT was the first problem in history shown to be **NP**-complete.

Interpretation

A common interpretation of this theorem is: For an appropriately chosen block cipher $F\colon \mathbb{F}_2^n \times \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^n$ the attack with known plaintext (against the key $k \in \mathbb{F}_2^l$) is not efficient. However from a strict mathematical point of view the theorem *doesn't prove anything* of practical relevance:

1. It relates to an algorithm for *arbitrary* polynomial equations (over $\mathbb{F}_2$). It doesn't contain any assertion for special classes of polynomials, or for a concrete system of equations.

2. It gives a pure proof of (non-) existence, and provides no hint as how to construct a concrete example of a "difficult" system of equations. Note that we know that some concrete systems admit easy solutions.

3. Even if we could find concrete examples of "difficult" systems the theorem would not make any assertion whether only some rare instances (the "worst cases") are difficult, or almost all (the "generic cases")—and this is what the cryptologist wants to know. Maybe there is an algorithm that solves polynomial systems for almost all tuples of unknowns in an efficient way, and only fails for a few exceptional tuples.

Despite these critical comments the theorem raises hope that there are "secure" bitblock ciphers, and the designers of bitblock ciphers follow the

Rule of thumb *Systems of linear equations for bits admit very efficient solutions. Systems of nonlinear equations for bits in almost all cases admit no efficient solution.*

A recent article on the difficulty of systems of polynom equations is

- D. Castro, M. Giusti, J. Heintz, G. Matera, L. M. Pardo: The hardness of polynomial equation solving. Found. Comput. Math. 3 (2003), 347–420.

Interpolation Attack

A variant of algebraic cryptanalysis with known plaintext is the interpolation attack, developed in

- Thomas Jakobsen, Lars R. Knudsen: The interpolation attack on block ciphers, FSE 1997.

The idea is simple: Equip the vector space $\mathbb{F}_2^n$ with a suitable multiplication and interpret it as the finite field $K = \mathbb{F}_{2^n}$ of characteristic 2. An encryption function with a fixed key $k \in \mathbb{F}_2^l$ then is a function $F_k : K \longrightarrow K$, hence a polynomial in one indeterminate over K. Let d be its degree. Then using interpolation this polynomial is determined by $d+1$ known plaintext blocks. The same is true for the inverse function. Using this the attacker can encrypt and decrypt without knowing the key explicitly.

The cipher designer who wants to prevent this attack should take care that encryption and decryption functions for every fixed key have large degrees as polynomials over K. This is realistic since polynomials over K may have (effective) degrees up to $2^n - 1$.

But beware that this attack may also work for some polynomials of high degree, for example for "sparse" polynomials having only a few coefficients $\neq 0$.

Linearisation of Overdetermined Systems of Equations

Systems of equations of higher order are sometimes solvable, if they are overdetermined, consisting of much more equations than unknowns. Then one simply treats some monomials as additional independent unknowns. Let's illustrate this by a simple example of three equations with two unknowns x and y:

$$\begin{aligned} x^3 + xy + y^5 &= 1, \\ 2x^3 - xy &= 0, \\ xy + 3y^5 &= 3. \end{aligned}$$

We substitute all occuring monomials: $u := x^3$, $v := xy$, $w := y^5$ and get the linear system

$$\begin{aligned} u + v + w &= 1 \\ 2u - v &= 0 \\ v + 3w &= 3 \end{aligned}$$

consisting of three equations involving three unknowns. The solution (in this case even manually derived) is $u = 0$, $v = 0$, $w = 1$. It is unique over a field K of characteristic $\neq 7$. From here we get the complete solution of the original system: $x = 0$, $y = 1$ or any 5th root of unity of K.

This attack gained some popularity in 2002 when there was a rumor that the new AES be vulnerable under this attack. However this rumor didn't survive a closer examination. As it turned out there were much too many dependencies between the linear equations.

2.4 SP Networks

In an ideal world we would know how to reliably measure the security of a bitblock cipher

$$F\colon \mathbb{F}_2^n \times \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^n$$

for realistic values of the block length n and the key length l, say of an order of magnitude of 128 bits or more.

In fact we know explicit measures of security, for example the linear potential, or the differential potential, that quantify the deviation from linearity, or the algebraic immunity, or others. Unfortunately all of these only give necessary, not sufficient, conditions for security, and moreover the efficient computability of these measures is limited to small block lengths n, about 8 or slightly larger.

Lacking a general efficient approach to security the design of bitblock ciphers usually relies on a structure that, although not obligatory, in practice seems to provide plausible security according to verifiable criteria. Most of the generally approved standard ciphers, such as DES and AES, follow this approach.

Rounds of Bitblock Ciphers

This common design scheme starts by constructing Boolean maps of small dimensions and then extending them to the desired block length in several steps:

1. Define one or more Boolean maps of small dimension q (= block length of the definition domain), say $q = 4$, 6, or 8, that are good for several security criteria. These maps are called **S-boxes** ("S" stands for Substitution), and are the elementary building blocks of the cipher.

2. Mix the round input with some of the key bits and then apply m S-boxes in parallel (or apply the one S-box m times in parallel) to get a map with the desired input width $n = mq$.

3. Then permute the complete resulting bitblock over its total width.

4. These steps together are a **"round"** of the complete scheme. Asset the weaknesses of the round map, that mainly result from using S-boxes of small dimension. Then reduce these weaknesses in a reasonably controlled way by iterating the scheme over several rounds of the same structure but with a changing choice of key bits.

5. Don't stop as soon as the security measures give satisfying values but add some surplus rounds to get a wide security margin.

Figure 2.1 outlines the scheme for a single round.

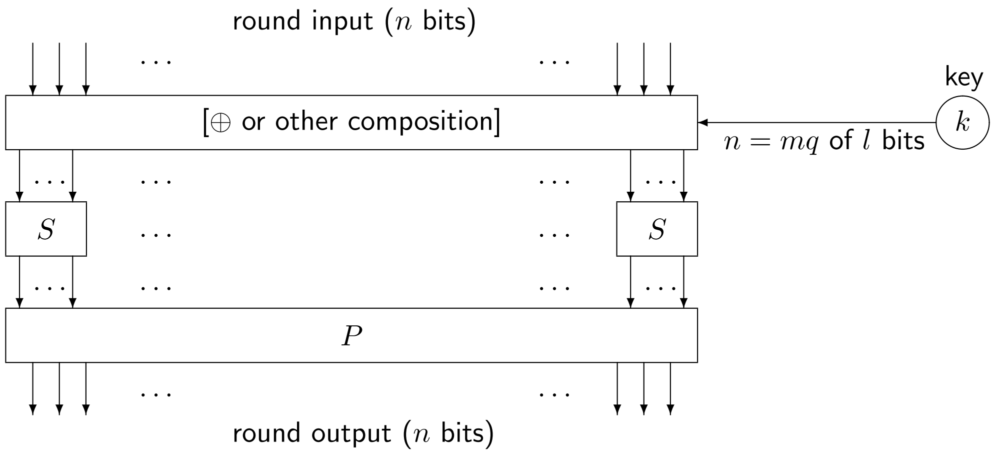


Figure 2.1: A single round of a bitblock cipher (S is a, maybe varying, S-box, P, a permutation, k, the key)

Shannon's Design Principles

The complete scheme is a special case of a somewhat more general proposal that goes back to SHANNON who required two basic features of block ciphers:

Diffusion The bits of the plaintext block "smear" over all parts of the block. This is done by applying permutations (a. k. a. as transpositions).

Confusion (complex dependencies) The interrelation between plaintext block and key on the one hand, as well as ciphertext block on the other hand should be as complex as possible (in particular as nonlinear as possible). Basic building blocks for this are substitutions.

The overall effect of both requirements, taken together, should result in an unforeseeable change of ciphertext bits for a slight change of the key.

The attacker should have no means to recognize whether a guessed key is "nearly correct".

Product Ciphers after Shannon

For the construction of strong block ciphers SHANNON proposed an alternating sequence of **S**ubstitutions and transpositions (= **P**ermutations), so-called **SP-networks**:

$$\mathbb{F}_2^n \xrightarrow{S_1(\bullet,k)} \mathbb{F}_2^n \xrightarrow{P_1(\bullet,k)} \mathbb{F}_2^n \longrightarrow \ldots$$
$$\ldots \longrightarrow \mathbb{F}_2^n \xrightarrow{S_r(\bullet,k)} \mathbb{F}_2^n \xrightarrow{P_r(\bullet,k)} \mathbb{F}_2^n$$

depending on a key $k \in \mathbb{F}_2^l$. In this scheme

$$\begin{aligned} S_i &= i\text{-th substitution} \\ P_i &= i\text{-th permutation} \\ P_i \circ S_i &= i\text{-th } \textbf{round} \end{aligned}$$

Alltogether the encryption function consists of r rounds.

Example: LUCIFER I (FEISTEL 1973)

Note that the permutations are special linear maps $P \colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$. Some recent bitblock ciphers, the most prominent being AES, replace permutations by more general linear maps that provide an even better diffusion. However the proper term **"LP-network"** is not yet in use.

2.5 Feistel Networks

Horst Feistel was the first (in the open world) who explicitly applied Shannon's design principles when he constructed the Lucifer ciphers.

The Kernel Map

Assume the blocksize is even: $n = 2s$. Decompose blocks $a \in \mathbb{F}_2^n$ into their left and right halves:

$$a = (L, R) \in \mathbb{F}_2^s \times \mathbb{F}_2^s$$

(We use uppercase letters to avoid confusion with the dimension l of the keyspace.) Moreover we have to agree on the order of the bits in a block:

- The **natural order** has the LSB (Least Significant Bit) always at the right end and assigns it the index 0, the MSB (Most Significant Bit) at the left end with index $n - 1$:

  $$b = (b_{n-1}, \ldots, b_0) \in \mathbb{F}_2^n.$$

  This corresponds to the base 2 representation of natural numbers in the integer interval $[0 \ldots 2^n[$:

  $$b_{n-1} \cdot 2^{n-1} + \cdots + b_1 \cdot 2 + b_0 \in \mathbb{N}$$

  This is the order we use in most situations.

- The **IBM order** has the bits in reverse (LSB at left, MSB at right) and assigns them the indices 1 to n:

  $$a = (a_1, \ldots, a_n) \in \mathbb{F}_2^n.$$

  This corresponds to the usual indexing of the components of a vector. Sometimes, in exceptional cases, the indices 0 to $n - 1$ are used.

The elemantery building blocks of a Feistel cipher are represented by a **kernel map**

$$f \colon \mathbb{F}_2^s \times \mathbb{F}_2^q \longrightarrow \mathbb{F}_2^s,$$

that need not fulfill any further formal requirements. In particular we don't require that the $f(\bullet, k)$ be bijective.

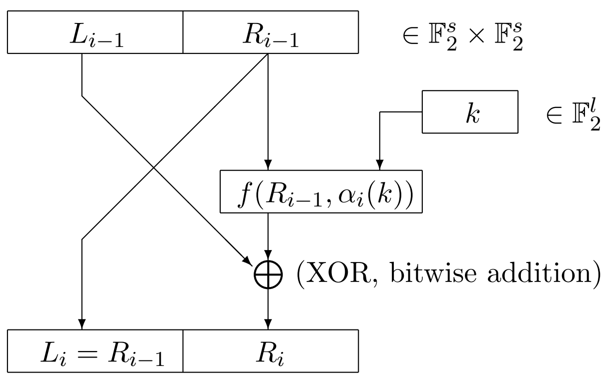
> However to get a useful cipher we should choose a kernel map f that already provides good confusion and diffusion. It should consist of a composition of substitutions and transpositions and be highly nonlinear.

Description of the Rounds

A Feistel cipher consists or r rounds. Each round uses a q-bit round key that is derived from the key $k \in \mathbb{F}_2^l$ by a process called the **key schedule**:

$$\alpha_i \colon \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^q \quad \text{for } i = 1, \ldots, r.$$

Then round i has this form:



We recognize the autokey principle in form of the addition of the left half and the transformed right half of a bitblock.

Algorithmic Description

From the graphical description we easily derive an algorithmic description:

$$\begin{array}{rcccl}
\textbf{Input} & \longrightarrow & a & = & (a_0, a_1) \in \mathbb{F}_2^s \times \mathbb{F}_2^s \\
 & & a_2 & := & a_0 + f(a_1, \alpha_1(k)) \\
 & & & & \qquad - \text{1st round, result } (a_1, a_2) \\
 & & \vdots & & \vdots \\
 & & a_{i+1} & := & a_{i-1} + f(a_i, \alpha_i(k)) \\
 & & & & \qquad - i\text{-th round, result } (a_i, a_{i+1}) \\
 & & & & \qquad - [a_i = R_{i-1} = L_i,\ a_{i+1} = R_i] \\
 & & \vdots & & \vdots \\
\textbf{Output} & \longleftarrow & c & = & (a_r, a_{r+1}) =: F(a, k)
\end{array}$$

Decryption

The decryption is done by the formula

$$a_{i-1} = a_{i+1} + f(a_i, \alpha_i(k)) \quad \text{for } i = 1, \ldots, r.$$

This boils down to the same algorithm, but the rounds in reverse order. Or in other words: The key schedule follows the reverse direction.

In particular we proved:

Theorem 3 (Feistel) *Let $F\colon \mathbb{F}_2^{2s} \times \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^{2s}$ be the block cipher with kernel map $f\colon \mathbb{F}_2^s \times \mathbb{F}_2^q \longrightarrow \mathbb{F}_2^s$ and key schedule $\alpha = (\alpha_1, \ldots, \alpha_r)$, $\alpha_i\colon \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^q$.*

Then the encryption function $F(\bullet, k)\colon \mathbb{F}_2^{2s} \longrightarrow \mathbb{F}_2^{2s}$ is bijective for every key $k \in \mathbb{F}_2^l$.

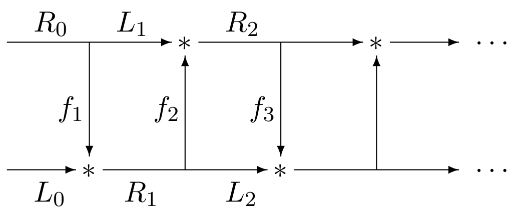
Addendum. *Decryption follows the same algorithm with the same kernel map f but the reverse key schedule $(\alpha_r, \ldots, \alpha_1)$.*

Note When the deryption starts with $c = (a_r, a_{r+1})$, then as a first step *the two halves must be swapped* because the algorithm starts with (a_{r+1}, a_r). To simplify this, in the last round of a Feistel cipher the interchange of L and R is usually dropped.

Remarks

- If f and the α_i are linear so is F.
- Usually the α_i are only selections, hence as maps projections $\mathbb{F}_2^l \longrightarrow \mathbb{F}_2^q$.
- Other graphical descriptions of the Feistel scheme are:

a) a ladder

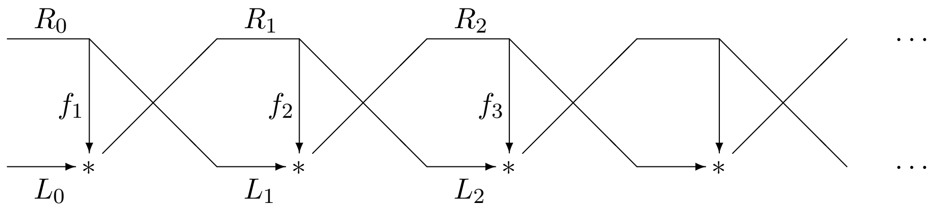


$$R_i = L_{i-1} * f_i(R_{i-1})$$

$$f_i = f(\bullet, \alpha_i(k))$$

$$L_i = R_{i-1}$$

b) a twisted ladder



Generalizations

1. Replace the group $(\mathbb{F}_2^s, +)$ by an arbitrary group $(G, *)$. Then the formulas for encryption and decryption are:

$$\begin{aligned} a_{i+1} &= a_{i-1} * f(a_i, \alpha_i(k))), \\ a_{i-1} &= a_{i+1} * f(a_i, \alpha_i(k)))^{-1}. \end{aligned}$$

2. Unbalanced FEISTEL ciphers (SCHNEIER/KELSEY): Divide the blocks into two different halves: $\mathbb{F}_2^n = \mathbb{F}_2^s \times \mathbb{F}_2^t$, $x = (\lambda(x), \rho(x))$. Then the encryption formula is:

$$\begin{aligned} L_i &= \rho(L_{i-1}, R_{i-1}) && \in \mathbb{F}_2^s, \\ R_i &= \lambda(L_{i-1}, R_{i-1}) + f(L_i, \alpha_i(k))) && \in \mathbb{F}_2^t. \end{aligned}$$

Examples

1. LUCIFER II (FEISTEL 1971, published in 1975),
2. DES (COPPERSMITH et al. for IBM in 1974, published as US standard in 1977),
3. many newer bitblock ciphers.

The usefulness of FEISTEL networks relies on the empirical observations:

- By the repeated execution through several rounds the "(s, q)-bit security" (or "local security") of the kernel map f is expanded to "(n, l)-bit security" (or "global security") of the complete FEISTEL cipher F.
- The complete cipher is composed of manageable pieces that may be "locally" optimized for security.

LUBY/RACKOFF underpinned the first of these observations by a theoretical result: A FEISTEL cipher with at least four rounds is not efficiently distinguishable from a random permutation, if its kernel map is random. This means that by FEISTEL's construction a map with good random properties but too small block length expands to a map with good random properties and sufficient block length.

Michael LUBY, Charles RACKOFF: How to construct pseudorandom permutations from pseudorandom functions. SIAM Journal on Computing 17 (1988), 373–386

Two words of caution about the LUBY/RACKOFF result:

- It doesn't say anything about an attack with known or chosen plaintext.
- It holds for true random kernel maps. However concrete FEISTEL ciphers usually restrict the possible kernel maps to a subset defined by a choice of 2^q keys.

2.6 Algebraic Attacks for Few Rounds

Formulas for Few Rounds

We write the recursion formula for a Feistel cipher as

$$(L_i, R_i) = (R_{i-1}, L_{i-1} + f(R_{i-1}, k_i))$$

where $k_i = \alpha_i(k)$ is the round key.

Proposition 2 *The results (L_r, R_r) of a* Feistel *cipher after $r = 2$, 3, or 4 rounds satisfy the equations*

$$\begin{aligned} L_2 - L_0 &= f(R_0, k_1), \\ R_2 - R_0 &= f(L_2, k_2); \end{aligned}$$

$$\begin{aligned} L_3 - R_0 &= f(L_0 + f(R_0, k_1), k_2), \\ R_3 - L_0 &= f(L_3, k_3) + f(R_0, k_1); \end{aligned}$$

$$\begin{aligned} L_4 - L_0 &= f(R_0, k_1) + f(R_4 - f(L_4, k_4), k_3), \\ R_4 - R_0 &= f(L_4, k_4) + f(L_0 + f(R_0, k_1), k_2). \end{aligned}$$

We used minus signs in order to make the formulas valid also for a generalization to abelian groups. In the (present) binary case plus and minus coincide. The purpose of the formulas is that beside the round keys k_i they involve only the plaintext (L_0, R_0) and the ciphertext (L_r, R_r), data that are assumed as known for algebraic cryptanalysis.

Proof. In the case of two rounds the equations are

$$\begin{aligned} L_1 &= R_0, \\ R_1 &= L_0 + f(R_0, k_1), \\ L_2 &= R_1 = L_0 + f(R_0, k_1), \\ R_2 &= L_1 + f(R_1, k_2) = R_0 + f(L_2, k_2); \end{aligned}$$

the assertion follows immediately.

In the case of three rounds we have

$$\begin{aligned} L_1 &= R_0, \\ R_1 &= L_0 + f(R_0, k_1), \\ L_2 &= R_1 = L_0 + f(R_0, k_1), \\ R_2 &= L_1 + f(R_1, k_2) = R_0 + f(L_2, k_2), \\ L_3 &= R_2 = R_0 + f(L_0 + f(R_0, k_1), k_2), \\ R_3 &= L_2 + f(R_2, k_3) = L_0 + f(R_0, k_1) + f(L_3, k_3). \end{aligned}$$

The case of four rounds is left to the reader. ◇

Two-Round Ciphers

For a known plaintext attack assume that L_0, R_0, L_2, R_2 are given. We have to solve the equations

$$\begin{aligned} L_2 - L_0 &= f(R_0, k_1) \\ R_2 - R_0 &= f(L_2, k_2) \end{aligned}$$

for k_1 and k_2. Thus the security of the cipher only depends on the difficulty of inverting the kernel function f. Since usually q, the bitlength of the partial keys, is much smaller than the total key length l the 2^{q+1} evaluations of f for an exhaustion could be feasible. Note that this consideration doesn't depend on the key schedule α—the attacker simply determines the actually used keybits (k_1, k_2).

Example: We equip $\mathbb{F}_2^s$ with the multiplication "$\cdot$" of the field $\mathbb{F}_t$, $t = 2^s$, [see Appendix A] and take

$$f(x, y) = x \cdot y.$$

(Note that f is non-linear as a whole, but linear in the key bits.) Assume the key schedule is defined by $l = 2q$ and k_i = left or right half of k, depending on whether i is odd or even. Then the equations become

$$\begin{aligned} L_2 - L_0 &= R_0 \cdot k_1, \\ R_2 - R_0 &= L_2 \cdot k_2, \end{aligned}$$

hence are easily solved. (If one of the factors R_0 or L_2 vanishes, we need another known plaintext block.)

Of course chosing a kernel map f that is linear in the key bits was a bad idea anyway. But we could solve also slightly more complicated equations, say quadratic, cubic, or quartic.

Three-Round Ciphers

In the case of three rounds the equations are considerably more complex because f is iterated. However the attacker can mount a Meet-in-the-Middle attack with a single known plaintext, if the bit length q of the partial keys is not too large: She calculates the intermediate results (L_1, R_1) of the first round for all possible partial keys k_1, and stores them in a table. Then she performs an exhaustion over the last two rounds as described for two-round ciphers above. The total expenses are $3 \cdot 2^q$ evaluations of f, and 2^q memory cells.

These considerations suggest that FEISTEL *ciphers should have at least four rounds* and support the above mentioned result by LUBY and RACKOFF. We see how the resistance of the scheme against an algebraic attack increases with the number of rounds, at least if the kernel map f is sufficiently complex.

For the example above with kernel map = multiplication of $\mathbb{F}_{2^s}$ the equations become:

$$\begin{aligned} L_3 - R_0 &= [L_0 + R_0 \cdot k_1)] \cdot k_2, \\ R_3 - L_0 &= [R_0 + R_3] \cdot k_1. \end{aligned}$$

They are nonlinear in the key bits but easily solved in the field $\mathbb{F}_{2^s}$.

Four-Round Ciphers

The equations are much more complex. Even in the example they are quadratic in two unknowns:

$$\begin{aligned} L_4 - L_0 &= [R_0 + R_4 + L_4 \cdot k_2] \cdot k_1, \\ R_4 - R_0 &= [L_4 + L_0 + R_0 \cdot k_1] \cdot k_2. \end{aligned}$$

However in this trivial example they can be solved: eliminating k_1 yields a quadratic equation for k_2 [**Exercise**].

2.7 Lucifer

History and Relevance

Lucifer was the first published bitblock cipher. Horst Feistel at IBM developed it around 1970. It is in fact a Feistel cipher, and is a predecessor of the standard cipher DES that was developed shortly after. Compared with DES Lucifer seems stronger at first sight, but has some decisive weaknesses that became apparent in the meantime.

> Here we describe the variant that is called "Lucifer II" because of its publication date in 1975. First published (1973 in *Scientific American*) was a somewhat different variant called "Lucifer I".

These are the characteristics of Lucifer (compare the figures below):

- 128-bit key. This is a large enough key space even for today's requirements.
- 128-bit blocks. This is also considered sufficent down to the present day.
- For processing the 128-bit blocks (of keys and texts) are divided into 16 bytes. (From a historic point of we should say "octets" instead of "bytes" since in early computers a byte not necessarily consisted of exactly 8 bits.)
- The Feistel scheme consists of 16 rounds.
- In each round the 8 bytes of the right half R_i of a block (= 64 Bits) are processed quasi in parallel. In other words, every round processes 8 blocks à 1 byte, each one independently from the other ones.
- Each of the rounds consists of a substitution and a permutation. In between some key bits are added (XORed).
- Nonlinearity enters the algorithm only by the substitution. The newly added key bits are processed in a linear way in the actual round, but then are input to the nonlinear substitution of the next round.
- The substitution of a byte starts with a decomposition into two halves à 4 bit each of which is separately transformed by a substitution

  $$S_0, S_1 \colon \mathbb{F}_2^4 \longrightarrow \mathbb{F}_2^4.$$

  S_0 and S_1 are fixed substitutions used during the whole encryption process. Only the assignement of S_0 and S_1 to the 4-bit halves varies depending on a certain key bit. It is common use to call such non-linear BOOLEean maps **"S-boxes"** when they occur as even more elementary building blocks of a kernel map.

The algorithm is very well suited for an implementation in hardware, in particular for 8-bit architectures. It is not so efficient in software for its many bit permutations.

Composing the kernel map of many small (identical or similar) S-boxes is a way followed often also today. A *rule of thumb*: the smaller the S-boxes, the more rounds are needed to achieve security.

The presentation of the algorithm follows the paper

- Arthur Sorkin: Lucifer, a cryptographic algorithm. Cryptologia 8 (1984), 22–41.

The Key Schedule

Denote the 16 bytes of the key $k \in \mathbb{F}_2^{128}$ by

$$k = (k_0, \ldots, k_{15}) \in (\mathbb{F}_2^8)^{16}$$

(IBM order but beginning with 0). Round i involves the selection

$$\alpha_i(k) = (\alpha_{ij}(k))_{0 \le j \le 7} \quad \text{of 8 bytes } \alpha_{ij}(k) = k_{7i+j-8 \bmod 16}.$$

This formula looks complicated but describes a quite simple selection scheme:

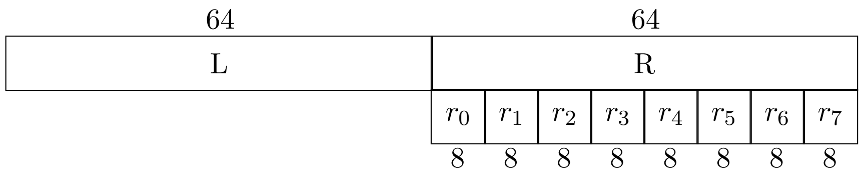
Round	Position							
	0	1	2	3	4	5	6	7
1	0	1	2	3	4	5	6	7
2	7	8	9	10	11	12	13	14
3	14	15	0	1	2	3	4	5
4	5	6	7	8	9	10	11	12
5	12	13	14	15	0	1	2	3
6	3	4	5	6	7	8	9	10
7	10	11	12	13	14	15	0	1
8	1	2	3	4	5	6	7	8
9	8	9	10	11	12	13	14	15
10	15	0	1	2	3	4	5	6
11	6	7	8	9	10	11	12	13
12	13	14	15	0	1	2	3	4
13	4	5	6	7	8	9	10	11
14	11	12	13	14	15	0	1	2
15	2	3	4	5	6	7	8	9
16	9	10	11	12	13	14	15	0

With each new round the selection is cyclically shifted by 7 positions. Note that each byte occurs at each position exactly once. The position defines to which byte of the actual 64-bit blocks the actual key byte applies. Furthermore the byte $\alpha_{i0}(k)$ in position 0 serves as “transformation control byte”.

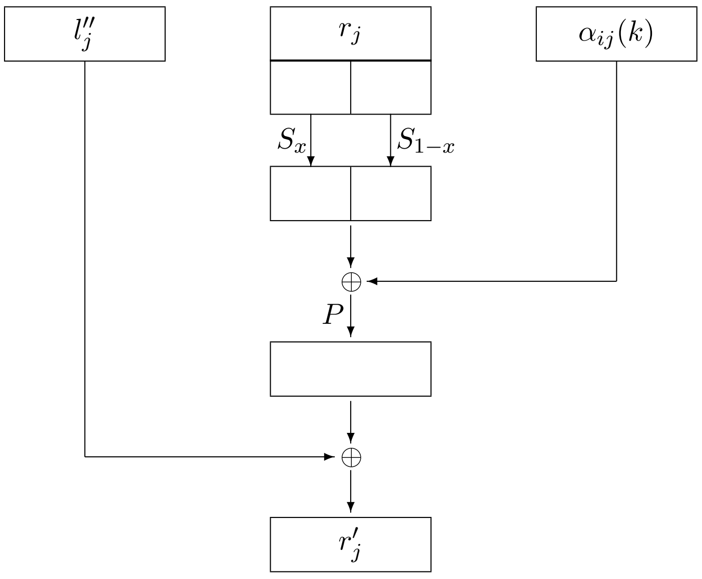
The Round Map

In round i the input, that is the right 64-bit part of the actual block, is divided into eight bytes

$$r = (r_0, \ldots, r_7).$$



Byte number j is transformed as follows by this round:



Here l''_j is a fixed selection of eight bits from the left part of the actual block. The transformation control byte

$$\alpha_{i1}(k) = (b_0, \ldots, b_7)$$

is taken from right to left, and $x = b_{7-j}$.

The definition of the kernel map f is clear from this picture. The explicit formula is not quite compact, therefore we omit it.

The S-Boxes

The S-boxes

$$S_0, S_1 \colon \mathbb{F}_2^4 \longrightarrow \mathbb{F}_2^4$$

are given by their value tables. Here the 4-bit blocks are written in hexadecimal notation, for example $1011 = B$.

$x =$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_0(x) =$	C	F	7	A	E	D	B	0	2	6	3	1	9	4	5	8

$x =$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_1(x) =$	7	2	E	9	3	B	0	4	C	D	1	A	6	F	8	5

The Permutations

The permutation P permutes the bits of a byte as follows:

$$\begin{aligned} P\colon \mathbb{F}_2^8 &\longrightarrow \mathbb{F}_2^8, \\ (z_0, z_1, z_2, z_3, z_4, z_5, z_6, z_7) &\mapsto (z_2, z_5, z_4, z_0, z_3, z_1, z_7, z_6). \end{aligned}$$

The bitwise addition of the left half to the transformed right half follows a permutation whose description is: Divide the left half of the actual block into eight bytes:

$$L = (l_0, l_1, l_2, l_3, l_4, l_5, l_6, l_7).$$

Rotate these cyclically after each step. Then for r_j they are in the positions

$$(l'_0, \ldots, l'_7) = (l_j, \ldots, l_{7+j \bmod 8}).$$

Finally construct the byte l''_j as

$$l''_j = (\text{Bit 0 of } l'_7, \text{Bit 1 of } l'_6, \text{Bit 2 of } l'_2, \ldots)$$

etc. in the order $(7, 6, 2, 1, 5, 0, 3, 4)$, and add it to r_j.

Chapter 3

Modes of Operation of Block Ciphers

A bitblock encryption function $f\colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ is primarily defined on blocks of fixed length n. To encrypt longer (or shorter) bit sequences the sender must

1. split the sequence into n-bit blocks,
2. pad the last block if necessary with
   - zeroes or
   - random values or
   - context information.

Then each block is encrypted by f, but in general one uses some sort of "chaining". Four chaining procedures, called "modes of operation" were standardized together with DES:

- ECB,
- CBC,
- CFB,
- OFB.

These chaining procedures apply to each block cipher. The standardization in the context of AES added two more modes:

- CTR,
- XTS.

For a description of the modes a suitable general framework is a “block alphabet” Σ, with $\mathbb{F}_2^n$ as most important example, equipped with a group composition $*$. Furthermore we fix an encryption function

$$f\colon \Sigma \longrightarrow \Sigma.$$

The dependence on the key doesn’t matter in this context and therefore is dropped in the notation.

3.1 ECB = Electronic Code Book

Description

Let r be the number of blocks of the plaintext $(a_1, \ldots, a_r)$.

Encryption: In ECB mode each block is encrypted independently of the other blocks:

$$a = (a_1, \ldots, a_r) \mapsto c = (c_1, \ldots, c_r) \in \Sigma^r \quad \text{with } c_i = f(a_i).$$

$$\begin{tikzcd}
\boxed{a_1} \arrow[r] & \boxed{c_1} \\
\boxed{a_2} \arrow[r] & \boxed{c_2} \\
\vdots & \vdots \\
\boxed{a_r} \arrow[r] & \boxed{c_r}
\end{tikzcd}$$

Decryption: $a_i = f^{-1}(c_i)$.

Properties

ECB mode simply is a monoalphabetic substitution on Σ. For sufficiently large $\#\Sigma$ this is secure from a ciphertext-only attack. But there are several disadvantages:

- ECB encryption leaks information on identical blocks. Even if the plaintext is not random, the rule of thumb from the Birthday Paradox applies in the interpretation (for $\Sigma = \mathbb{F}_2^n$): "After $2^{n/2}$ bits ECB encryption begins to leak information." Wikipedia has a nice illustration of this effect, see `http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation` The other modes significantly enlarge this bound.

- Building a "codebook" from known plaintext blocks is not unrealistic. For structured messages, say bank transactions, there occur many blocks of known plaintext.

- An active attack by exchanging or inserting single blocks of ciphertext (for example with known, "sympathic" plaintext) is possible. For example an attacker who knows which block contains the receiver of a money transfer could exchange this block with a corresponding block from another transfer for another receiver. He doesn't need to know the key.

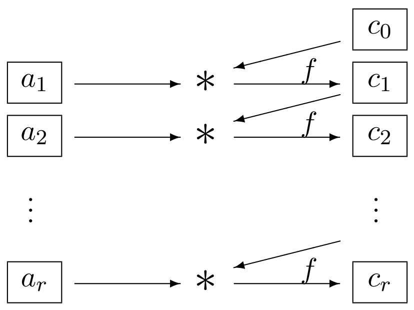
- If the situation allows for an attack with *chosen* plaintext (as in a black box analysis), trial encryption and dictionary attacks can be mounted.

In view of these problems generating diffusion between the plaintext blocks seems a much better approach. In the following sections we look at modes of operation that achieve this effect.

3.2 CBC = Cipher Block Chaining

Description

Choose a start value c_0 at random (also called IV = "Initialization Vector"). Then the procedure looks like this:



Encryption: In CBC mode the formula for encryption is:

$$\begin{aligned} c_i &:= f(a_i * c_{i-1}) \quad \text{for } i = 1, \ldots, r \\ &= f(a_i * f(a_{i-1} * \cdots f(a_1 * c_0) \ldots)). \end{aligned}$$

Decryption: $a_i = f^{-1}(c_i) * c_{i-1}^{-1}$ for $i = 1, \ldots, r$.

Properties

- Each ciphertext block depends on *all previous* plaintext blocks (diffusion).
- An attacker is not able to replace or insert text blocks unnoticeably.
- Identical plaintext blocks in general encrypt to different ciphertext blocks.
- On the other side an attack with known plaintext is not more difficult, compared with ECB mode.
- Each plaintext block depends on two ciphertext blocks.
- As a consequence a transmission error in a single ciphertext block results in (only) two corrupted plaintext blocks ("self synchronisation" of CBC mode).

Question: *Does it make sense to treat the initialization vector c_0 as secret and use it as an additional key component?* (Then for the example of DES we had 56 proper key bits plus a 64 bit initialization vector, making a total of 120 key bits.)

Answer: No!

Reason: In the decryption process only a_1 depends on c_0. This means that keeping c_0 secret conceals known plaintext only for the first block. If the attacker knows the second or a later plaintext block, then she may determine the key as in ECB mode (by exhaustion, or by an algebraic attack, or by any other attack with known plaintext).

Remarks

1. CBC is the composition $f\circ$ (ciphertext autokey). In the trivial case $f = \mathbf{1}_\Sigma$ only the (completely unsuited) ciphertext autokey cipher with key length 1 is left.

2. (John Kelsey in the mailing list `cryptography@c2.net`, 24 Nov 1999) If there occurs a "collision" $c_i = c_j$ for $i \neq j$, then $f(a_i * c_{i-1}) = f(a_j * c_{j-1})$, hence $a_i * c_{i-1} = a_j * c_{j-1}$ and therefore $a_j^{-1} * a_i = c_{j-1} * c_{i-1}^{-1}$. In this way the attacker gains some information on the plaintext.

   By the Birthday Paradox this situation is expected after about $\sqrt{\#\Sigma}$ blocks.

   The longer the text, the more such collisions will occur. This effect reassures the rule of thumb for the frequency of key changes: change the key in good time before you encrypt $\sqrt{\#\Sigma}$ blocks.

3.3 Variants of CBC

Plaintext Autokey

Replacing the ciphertext autokey encryption for CBC mode by plaintext autokey yields the following scheme:

$$\begin{tikzcd}
\boxed{a_0} \arrow[dr] & & \\
\boxed{a_1} \arrow[r] \arrow[dr] & * \arrow[r, "f"] & \boxed{c_1} \\
\boxed{a_2} \arrow[r] \arrow[dr] & * \arrow[r, "f"] & \boxed{c_2} \\
\vdots & & \vdots \\
\boxed{a_r} \arrow[r] & * \arrow[r, "f"] & \boxed{c_r}
\end{tikzcd}$$

that sometimes is called PBC = Plaintext Block Chaining.

Encryption: After choosing an initialization vector a_0 the formula for encryption is:

$$c_i := f(a_i * a_{i-1}) \quad \text{for } i = 1, \ldots, r.$$

Decription: The formula is:

$$a_i = f^{-1}(c_i) * a_{i-1}^{-1} \quad \text{for } i = 1, \ldots, r.$$

However this method seems not to be in widely accepted use, and there seem to be no relevant results on its security.

PCBC = error-Propagating CBC

This procedure mixes CBC and PBC. It follows the scheme:

$$\begin{tikzcd}
 & & \boxed{c_0} \arrow[dl] \\
\boxed{a_1} \arrow[r] \arrow[dr] & * \arrow[r, "f"] & \boxed{c_1} \arrow[dl] \\
\boxed{a_2} \arrow[r] & * \arrow[r, "f"] & \boxed{c_2} \\
\vdots \arrow[dr] & & \vdots \arrow[dl] \\
\boxed{a_r} \arrow[r] & * \arrow[r, "f"] & \boxed{c_r}
\end{tikzcd}$$

Encryption: After choosing the initialization vector $a_0 = e$ (neutral element of the group) encryption is by the formula

$$c_i := f(a_i * a_{i-1} * c_{i-1}) \quad \text{for } i = 1, \ldots, r.$$

In the case of a bitblock cipher we choose $a_0 = 0$, the null block.

Decryption: The formula is

$$a_i = f^{-1}(c_i) * c_{i-1}^{-1} * a_{i-1}^{-1} \quad \text{for } i = 1, \ldots, r.$$

This mode was implemented in early versions of Kerberos but then abandoned.

Generalization by Meyer/Matyas

$$c_i := f(a_i * h(a_{i-1}, c_{i-1})) \quad \text{for } i = 1, \ldots, r,$$

where in the case $\Sigma = \mathbb{F}_2^n$ addition modulo 2^n is suggested for h.

BCM = Block Chaining Mode

This mode follows the scheme:

$$\begin{tikzcd}
 & & \boxed{c_0} \arrow[rr] & & \boxed{d_1} \arrow[d] \arrow[dlll] \\
\boxed{a_1} \arrow[r] & * \arrow[r, "f"] & \boxed{c_1} \arrow[r] & * & \boxed{d_2} \arrow[d] \arrow[dlll] \\
\boxed{a_2} \arrow[r] & * \arrow[r, "f"] & \boxed{c_2} \arrow[r] & * & \boxed{d_3} \\
\vdots & & \vdots & & {} \arrow[dlll] \\
\boxed{a_r} \arrow[r] & * \arrow[r, "f"] & \boxed{c_r} & &
\end{tikzcd}$$

Formula for encryption:

$$\begin{aligned} d_i &:= c_0 * \ldots * c_{i-1}, \\ c_i &:= f(a_i * d_i) \quad \text{for } i = 1, \ldots, r. \end{aligned}$$

An Application of CBC

CBC-MAC (= “Message Authentication Code”) is a key-dependent “hash function” that serves for checking the integrity of messages. It is standardized in ISO/IEC 9797 and used in electronic banking.

Sender and receiver of the message—these could be the same person if the MAC used for securing the integrity of a stored file—share the key k and use the encryption function $f = f_k$.

The MAC of a text $a = (a_1, \ldots, a_r)$ is the last ciphertext block where a is encrypted in CBC mode. Hence

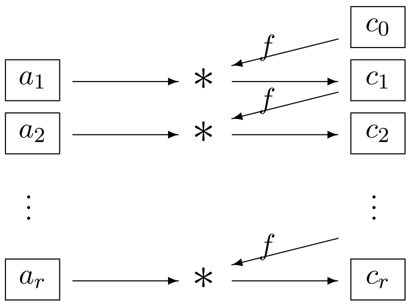
$$\mathrm{MAC}(a) = c_r = f(a_r * f(a_{r-1} * \cdots f(a_1 * c_0) \ldots)).$$

If $\mathrm{MAC}(a)$ is sent together with a, then the receiver may check the authenticity of the sender and the integrity of the content. Only someone who has the key can calculate this value correctly.

The disadvantage of this procedure is the need of sharing a secret k. In a legal dispute each of the two parties can contend a forgery by the other one.

3.4 CFB = Cipher Feedback

Description (of the simplest version)



Encryption in CFB mode is by the formula

$$\begin{aligned} c_i &:= a_i * f(c_{i-1}) \quad \text{for } i = 1, \ldots, r \\ &= a_i * f(a_{i-1} * f(\cdots a_1 * f(c_0) \ldots)). \end{aligned}$$

Decryption: $a_i = c_i * f(c_{i-1})^{-1}$ for $i = 1, \ldots, r$.

Properties

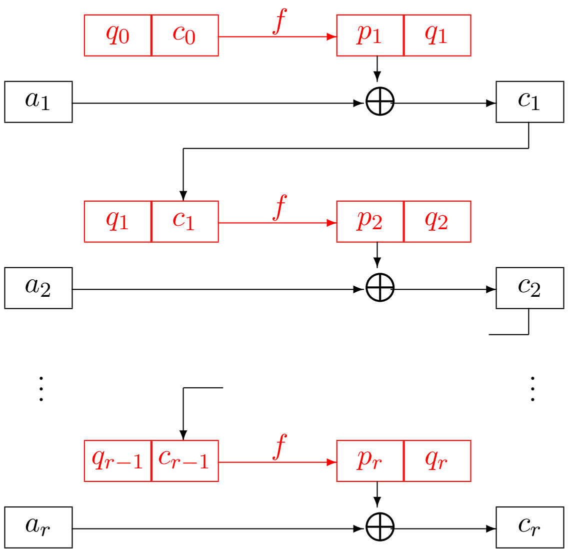
- As before the initialization vector is unsuited as additional key component.
- As before this mode doesn't make an attack with known plaintext more difficult.
- Note that also decryption uses f, not f^{-1}. Therefore:
  - CFB mode doesn't make sense for asymmetric ciphers.
  - On the other hand CFB mode may be used with a (key dependent) one-way or hash function f.
- For the identical map $f = \mathbf{1}_\Sigma$ CFB again reduces to ciphertext autokey.
- (David WAGNER) ECB $\circ$ CFB = CBC:

For a proof take c_0 as initialization vector for CFB, and $c_0' := f(c_0)$ as initialization vector for CBC. Then

$$\begin{aligned} c_1 &= \text{CFB}(a_1) = a_1 * f(c_0), \\ c_1' &= \text{ECB}(c_1) = f(a_1 * f(c_0)) = f(a_1 * c_0') = \text{CBC}(a_1), \\ c_2 &= \text{CFB}(a_2) = a_2 * f(c_1), \\ c_2' &= \text{ECB}(c_2) = f(a_2 * f(c_1)) = f(a_2 * c_1') = \text{CBC}(a_2), \\ &\text{etc.} \end{aligned}$$

The Standardized Version

…uses a shift register, hence is defined only in the case of $\Sigma = \mathbb{F}_2^n$. Here $1 \leq t \leq n$, and the encryption procedure uses blocks $a_i \in \mathbb{F}_2^t$ of length t. The current ciphertext block c_i of length t is shifted from the right into the shift register (drawn in red):



The q_i are bitblocks of length $n-t$.

As it turned out later the security of this more general version decreases with t. Therefore its use is not recommended.

3.5 OFB = Output Feedback

Description (of the simplest version)

$$\begin{tikzcd}
 & & \boxed{s_0} \arrow[d, "f"'] & & \\
\boxed{a_1} & * & \boxed{s_1} \arrow[d, "f"'] & = & \boxed{c_1} \\
\boxed{a_2} & * & \boxed{s_2} \arrow[d] & = & \boxed{c_2} \\
\vdots & & \vdots \arrow[d, "f"'] & & \vdots \\
\boxed{a_r} & * & \boxed{s_r} & = & \boxed{c_r}
\end{tikzcd}$$

This mode also was originally defined as shift register version. Here too using a blocklength of $t < n$ weakens the security [Jueneman, Crypto 82].

Encryption in OFB mode is by the formula

$$c_i := a_i * s_i, \quad s_i := f(s_{i-1}) \quad \text{for } i = 1, \ldots, r.$$

Decryption by the formula

$$a_i = c_i * s_i^{-1}, \quad s_i := f(s_{i-1}) \quad \text{for } i = 1, \ldots, r.$$

Properties

- There is no diffusion. However identical plaintext blocks in general yield different ciphertext blocks.
- In the case $\Sigma = \mathbb{F}_2^s$ OFB simply is a bitstream cipher where f serves as "random generator".
- If encryption or decryption is time critical, the sender or the receiver (or both) might precalculate the "key stream" s_i.
- Here too the decryption uses only f, not f^{-1}.
- For $\Sigma = \mathbb{F}_2^s$ the cipher is an involution, that is encryption and decryption are the same function. More generally this holds when the group Σ has exponent 2.

- Under an attack with known plaintext the pair (a_1, c_1) reveals the value of s_1, the next pair (a_2, c_2), the value of $s_2 = f(s_1)$. This leads to an attack with known plaintext against the function f itself.
- Keeping the initialization vector s_0 secret doesn't increase the security of the cipher for OFB (like for the other modes).

Variant: Counter Mode CTR

The simplest case is

$$c_i := a_i * f(i) \quad \text{for } i = 1, \ldots, r.$$

There are same slight variants, for example starting with another number than 1.

Chapter 4

DES

The “Data Encryption Standard” (DES) is essentially a development by an IBM research group around Feistel and Coppersmith. The NSA was involved: It arranged for a modification of the S-boxes and a reduction of the key length to 56 bits. Contrary to all speculations both of these changes didn’t weaken the security.

DES was published in 1975, and standardized by NBS (National Bureau of Standards—now NIST) in the USA in 1977. The objective was to provide a reliable cipher for sensitive (but not top secret) data of the administration for the next 10 or 15 years.

The standard requires a hardware implementation of the algorithm. The proper name of the algorithm is DEA, but usually also software implementations are denoted by DES. From 1989 to 1998 the US administration restricted the export of DES chips.

DES encrypts 64 bit blocks using a 56 bit key. The encryption of a block starts with a fixed (known) permutation, and ends with the inverse permutation. Although this permutation is known it yields a first bit of diffusion. In between there are 16 rounds that increase diffusion and confusion. The only difference between the single rounds consists in the selection of a different 48 bit subset from the key.

The decryption algorithm is almost identical with the encryption algorithm with the only difference that it runs through the key selection in the reverse direction.

In the following sections we describe the algorithmus in steps “outwards from the interior”. In the figures $\oplus$ denotes the bitwise addition $\bmod 2$ (XOR).

4.1 The Kernel Map

The innermost layer of DES is the “kernel map”

$$f\colon \mathbb{F}_2^{32} \times \mathbb{F}_2^{48} \longrightarrow \mathbb{F}_2^{32},$$

that takes 32 text bits and 48 key bits as input. First some of the 32 text bits are repeated, blowing them up to 48 bits. This “expansion map”

$$E\colon \mathbb{F}_2^{32} \longrightarrow \mathbb{F}_2^{48}$$

is given by the following table:

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

The correct interpretation of this table is

$$E(b_1 b_2 \ldots b_{32}) = b_{32} b_1 b_2 b_3 \ldots b_{31} b_{32} b_1.$$

The expanded 48 bits and the 48 bit partial key are added (as binary vectors). The resulting 48 bits are divided into 8 groups each consisting of 6 bits. These groups are fed into the S-boxes 1 to 8:

$$S_j\colon \mathbb{F}_2^{6} \longrightarrow \mathbb{F}_2^{4} \quad (j = 1, \ldots, 8).$$

The description of the S-boxes is in the next section.

The S-boxes together make up the substitution

$$S\colon \mathbb{F}_2^{48} \longrightarrow \mathbb{F}_2^{32}.$$

Finally we apply the “P-box” (permutation)

$$P\colon \mathbb{F}_2^{32} \longrightarrow \mathbb{F}_2^{32}$$

that is given by the following table meaning

$$P(b_1 b_2 \ldots b_{32}) = b_{16} b_7 \ldots b_4 b_{25}.$$

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

The complete kernel map is in the following figure:

$$\begin{tikzcd}
\text{32-bit input} & & & & \text{48-bit partial key} & \\
\mathbb{F}_2^{32} \arrow[d, "E"'] & & & & & E = \text{expansion map} \\
\mathbb{F}_2^{48} \arrow[dr] & & & & \mathbb{F}_2^{48} \arrow[dlll] & \\
& \bigoplus \arrow[d] & & & & \\
& \mathbb{F}_2^{48} = & \mathbb{F}_2^{6} \arrow[d, "S_1"'] & \times \ldots \times & \mathbb{F}_2^{6} \arrow[d, "S_8"] & S_j = j\text{-th S-box} \\
& \mathbb{F}_2^{32} = \arrow[d, "P"'] & \mathbb{F}_2^{4} & \times \ldots \times & \mathbb{F}_2^{4} & P = \text{P-box} \\
& \mathbb{F}_2^{32} & & & &
\end{tikzcd}$$

4.2 The S-Boxes

Each of the eight S-boxes S_j is given by a 4×16-matrix defined by the table

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Each row is a permutation of the numbers $0, \ldots, 15$. To get $S_j(b_1 \ldots b_6)$ we interpret $b_1 b_6$ as binary representation of a number in $\{0, 3\}$, and $b_2 b_3 b_4 b_5$ as binary representation of a number in $\{0, 15\}$. Then in the matrix for S_j we go to row $b_1 b_6$, column $b_2 b_3 b_4 b_5$, and find a number there. We take the binary representation of this number. Example:

$$S_3(101100) = 0011 \quad \rightarrow \quad \text{Row 2, Column 6, number is 3.}$$

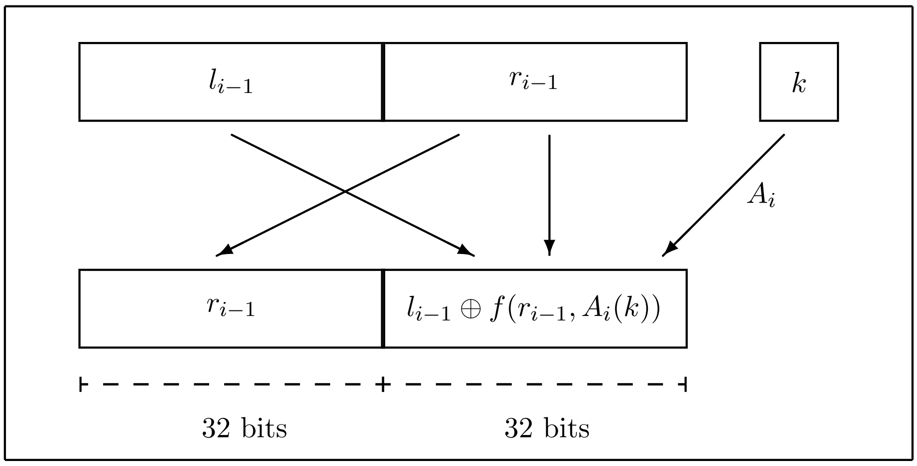
4.3 The Rounds

The 16 rounds of DES consist of maps

$$R_i \colon \mathbb{F}_2^{64} \times \mathbb{F}_2^{56} \longrightarrow \mathbb{F}_2^{64} \quad (i = 1, \ldots, 16),$$

that are defined in the following figure, using the i-th key selection

$$A_i \colon \mathbb{F}_2^{56} \longrightarrow \mathbb{F}_2^{48} \quad (i = 1, \ldots, 16).$$



The rounds only differ by their key selections $A_i(k)$. We recognize the FEISTEL scheme.

4.4 The Key Selection

To complete the description of the rounds we have yet to describe the key selection. First we expand the 56 bit key to 64 bits by appending a parity bit after each 7 bit subblock. However it doesn't matter which bit we append: the additional bits never enter the algorithm. In any case the first step is a map

$$\mathrm{Par}\colon \mathbb{F}_2^{56} \longrightarrow \mathbb{F}_2^{64}.$$

In the second step we extract the original 56 bits, but in a different order, given by the following table.

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

We have constructed a map

$$\mathrm{PC}_1\colon \mathbb{F}_2^{64} \longrightarrow \mathbb{F}_2^{56}$$

("Permuted Choice 1"). Now we divide the 56 bits into two 28 bit halves, and cyclically shift these to the left, all in all 16 times. This gives 16 maps

$$\mathrm{LS}_i\colon \mathbb{F}_2^{28} \longrightarrow \mathbb{F}_2^{28} \quad (i = 1, \ldots 16).$$

the amount of the shifts is given by the table:

1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

The first two shifts are by one bit, the next 6 ones by two bits, and so on. To get the i-th key selection A_i we apply the "Permuted Choice 2" after the i-th shift:

$$\mathrm{PC}_2\colon \mathbb{F}_2^{56} \longrightarrow \mathbb{F}_2^{48}$$

where the 48 bits are chosen in the order of the following table (omitting the bits 9, 18, 22, 25, 35, 38, 43, 54).

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

The complete key selection is

$$A_i = \mathrm{PC}_2 \circ \mathrm{LS}_i \circ \cdots \circ \mathrm{LS}_1 \circ \mathrm{PC}_1 \circ \mathrm{Par}\,.$$

We summarize this construction in the following diagram:

$$\begin{tikzcd}
\mathbb{F}_2^{56} \arrow[d, "\mathrm{Par}"] \\
\mathbb{F}_2^{64} \arrow[d, "\mathrm{PC}_1"] \\
\mathbb{F}_2^{56} & = & \mathbb{F}_2^{28} \arrow[d, "\mathrm{LS}_1"'] & \times & \mathbb{F}_2^{28} \arrow[d, "\mathrm{LS}_1"] \\
 & & \mathbb{F}_2^{28} \arrow[d, "\mathrm{LS}_2"'] & \times & \mathbb{F}_2^{28} \arrow[d, "\mathrm{LS}_2"] \arrow[r, "\mathrm{PC}_2"] & \mathbb{F}_2^{48} \\
 & & {} & \vdots & {} \\
 & & {} \arrow[d, "\mathrm{LS}_{16}"'] & & {} \arrow[d, "\mathrm{LS}_{16}"] \\
 & & \mathbb{F}_2^{28} & \times & \mathbb{F}_2^{28} \arrow[r, "\mathrm{PC}_2"] & \mathbb{F}_2^{48}
\end{tikzcd}$$

4.5 The Complete Algorithm

The last thing to do is to describe the initial permutation

$$\mathrm{IP}\colon \mathbb{F}_2^{64} \longrightarrow \mathbb{F}_2^{64}.$$

This is done by the following table:

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

The inverse of IP is the final permutation IP^{-1}. For convenience here is the corresponding table:

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Now the complete DES algorithm DES_k with key $k \in \mathbb{F}_2^{56}$ is the composition

$$\mathbb{F}_2^{64} \xrightarrow{\mathrm{IP}} \mathbb{F}_2^{64} \xrightarrow{R_1(\bullet,k)} \ldots \xrightarrow{R_{16}(\bullet,k)} \mathbb{F}_2^{64} \xrightarrow{T} \mathbb{F}_2^{64} \xrightarrow{\mathrm{IP}^{-1}} \mathbb{F}_2^{64}.$$

Here T is the interchange of the left and right 32 bit halves. The effect of this additional interchange is that DES_k^{-1} looks exactly like DES_k except that the order of the rounds is reversed.

Remark. The initial and final permutations maybe lead to a convenient wiring of input and output contacts on small processors. They have no cryptological effect because the cryptanalyst simply may strip them off. For a software implementation they function as brakes—but one must not omit them for a standard conforming implementation.

Chapter 5

Cryptanalysis of Bitblock Ciphers

For cryptanalyzing bitblock ciphers we know some basic approaches:

1. exhaustion = brute-force searching the complete key space
2. algebraic attack, see Chapter 2
3. statistical attacks against hidden linearity:
   (a) linear cryptanalysis (MATSUI/YAMAGISHI 1992), the subject of the following sections
   (b) differential cryptanalysis (MURPHY, SHAMIR, BIHAM 1990)
   (c) generalizations and mixtures of (a) and (b)

Differential cryptanalysis was known at IBM and NSA already in 1974 when designing DES. In contrast apparently linear cryptanalysis—though conceptually simpler—was unknown to the designers of DES. Accordingly the resistance of DES against linear cryptanalysis is suboptimal. However an important design criterion was:

- *The S-boxes should be as nonlinear as possible.*

In the following years many people developed generalizations and combinations of linear and differential cryptanalysis:

- related keys attack (BIHAM 1992, SCHNEIER)
- differentials of higher order (HARPES 1993, BIHAM 1994, LAI 1994)
- differential-linear cryptanalysis (LANGFORD/HELLMAN 1994)
- partial differentials (KNUDSEN 1995)

- I/O-sum analysis (Harpes/Kramer/Massey 1995)
- S-box-pair analysis (Davies/Murphy 1995, Mirza 1996)
- boomerang attack (Wagner 1999)
- slide attack against (maybe hidden) periodicity in ciphers or key schedules (Biryukov/Wagner 1999)
- impossible differentials (Biham/Biryukov/Shamir 1999)

All these statistical attacks—including linear and differential cryptanalysis—hardly break a cipher in the sense of classical cryptanalysis. They usually assume lots of known plaintexts, much more than an attacker could gather in a realistic scenario. Therefore a more adequate term is "analysis" instead of "attack". The analyses make sense for finding measures for some partial aspects of security of bitblock ciphers. They measure security for example by the number of known plaintext blocks needed for the attack. If a cipher resists an attacker even with exaggerated assumptions on her capabilities, then we feel safe to trust it in real life.

Given an SP-network the analysis starts with the nonlinear components of the single rounds, in particular with the S-boxes. The next step extends the potential attack over several rounds. This shows how the cost of the attack grows with the number of rounds. In this way we find criteria for the number of rounds for which the cipher is "secure"—at least from this special attack.

By the way we should never forget that the attack always relates to a certain fixed algebraic structure; in most cases to the structure of the plaintext space as a vector space over $\mathbb{F}_2$. Of course a similar attack could relate to another structure. A seemingly complex map could look simple if considered with the structure as cyclic group $\mathbb{Z}/n\mathbb{Z}$ in mind—or even with "exotic" structures invented for the analysis of this single map. In the following however we only consider the structure as a vector space over $\mathbb{F}_2$, the structure that is best understood.

Security Criteria for Bitblock Ciphers

To escape attacks bitblock ciphers, or their round maps, or their S-boxes, should fulfill some requirements. For background theory see the mathematical Appendix D.

- **Balance** All preimages have the same number of elements, or in other words, the values of the map are uniformly distributed. Irregularities of the distribution would provide hooks for statistical cryptanalysis.

- **Diffusion/avalanche effect** If a single plaintext bit changes, about 50% of the ciphertext bits change. This effect conceals similarity of plaintexts.
- **Algebraic complexity** The determination of preimages or parts thereof should lead to equations whose solution is as difficult as possible. This requirement is related to the algebraic degree of the map, but only in an indirect way. A suitable measure is “algebraic immunity”.
- **Nonlinearity** We know several criteria that measure linearity, also “hidden” linearity, and are relatively easy to describe and to handle. For example they quantify how susceptible Boolean maps are for linear or differential cryptanalysis.
  - The “linear potential” should be as low as possible, the “linear spectrum” (or “linear profile”) as balanced as possible.
  - The “differential potential” should be as low as possible, the “differential spectrum” (or “differential profile”) as balanced as possible.
  - The “nonlinearity” (in a narrow sense as the HAMMING distance from affine maps) should be as large as possible.
  - The “linearity distance”, the HAMMING distance from maps with “linear structure”, should be as large as possible.

Some of these criteria are compatible with each other, some criteria contradict other ones. Therefore the design of a bitblock cipher requires a balance between different criteria. Instead of optimizing a map for a single criterion the designer should aim at a uniformly high level for all criteria.

Cipher designers usually decide the conflict between balance and nonlinearity in favour of balance. But there is no really convincing reason for this—the psychological reason seems to be that statistical attacks that use the nonuniform distribution of the output of non-balanced maps are easier to understand and therefore taken more seriously. The trade-off for nonlinearity is then handled by increasing the number of rounds.

In this section we freely use the notations and results from the mathematical appendices A to E—often without explicit reference.

5.1 The Idea of Linear Cryptanalysis

Consider a bitblock cipher F of block length n and key length l,

$$F\colon \mathbb{F}_2^n \times \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^n.$$

Imagine the arguments of F as plain texts $a \in \mathbb{F}_2^n$ and keys $k \in \mathbb{F}_2^l$, the values of F as cipher texts $c \in \mathbb{F}_2^n$. A **linear relation** between a plaintext $a \in \mathbb{F}_2^n$, a key $k \in \mathbb{F}_2^l$, and a ciphertext $c = F(a,k) \in \mathbb{F}_2^n$ is described by three linear forms

$$\alpha\colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2, \quad \beta\colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2, \quad \text{and} \quad \kappa\colon \mathbb{F}_2^l \longrightarrow \mathbb{F}_2$$

as an equation

$$\kappa(k) = \alpha(a) + \beta(c). \tag{1}$$

If $I = (i_1, \ldots, i_r)$ is the index set that corresponds to the linear form κ—that is $\kappa(k) = k_{i_1} + \cdots + k_{i_r}$—, then writing (1) more explicitly we get an equation for the sum of the involved key bits $k_{i_1}, \ldots, k_{i_r}$:

$$k_{i_1} + \cdots + k_{i_r} = \alpha(a) + \beta(c),$$

For an attack with known plaintext a this reduces the number of unknown key bits to $l-1$ by elimination of one of these bits.

In general the odds of the relation (1) for concrete random values of k, a, and c are about fifty-fifty: both sides evaluate to 0 or 1 with probability $\frac{1}{2}$. Best for security is a frequency of 50% plaintexts a that make the relation true for a fixed key k, where $c = F(a,k)$ is the corresponding ciphertext. This would make the relation indistinguishable from a pure accidental one. If the probability of the relation,

$$p_{F,\alpha,\beta,\kappa}(k) := \frac{1}{2^n} \cdot \#\{a \in \mathbb{F}_2^n \mid \kappa(k) = \alpha(a) + \beta(F(a,k))\},$$

is conspicuously larger than $\frac{1}{2}$, this reveals a biased probability for the values of the bits of k, and would result in a small advantage for the cryptanalyst. If on the other hand the probability is noticeably smaller than $\frac{1}{2}$, then the complementary relation $\kappa(k) = \alpha(a) + \beta(c) + 1$ is true more often than by pure chance. This also is a weakness. Because the situation concerning the deviation of the probabilities from the ideal value $\frac{1}{2}$ is symmetric (and because the I/O-correlation and the potential are multiplicative, see Proposition 6) it makes sense to consider symmetric quantities, the **input-output correlation**:

$$\tau_{F,\alpha,\beta,\kappa}(k) := 2p_{F,\alpha,\beta,\kappa}(k) - 1$$

(in short: I/O-correlation) and the **potential** of a linear relation:

$$\lambda_{F,\alpha,\beta,\kappa}(k) := \tau_{F,\alpha,\beta,\kappa}(k)^2.$$

The I/O-correlation takes values between -1 and 1. It is the correlation of two Boolean functions on $\mathbb{F}_2^n$, namely $\alpha + \kappa(k)$ and $\beta \circ F_k$. (For fixed k the value of $\kappa(k)$ is constant, i. e. 0 or 1.) The first of these functions picks input bits, the second one, output bits. In general the correlation of Boolean functions $f, g\colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ is the difference

$$c(f,g) := \frac{1}{2^n} \cdot [\#\{x \in \mathbb{F}_2^n \mid f(x) = g(x)\} - \#\{x \in \mathbb{F}_2^n \mid f(x) \neq g(x)\}]\,.$$

The potential takes values between 0 and 1, and measures the deviation of the probability from $\frac{1}{2}$. In the best case it is 0, in the worst, 1. This "bad" extreme case would provide an exact and directly useable relation for the key bits. Figure 5.1 illustrates the connection.

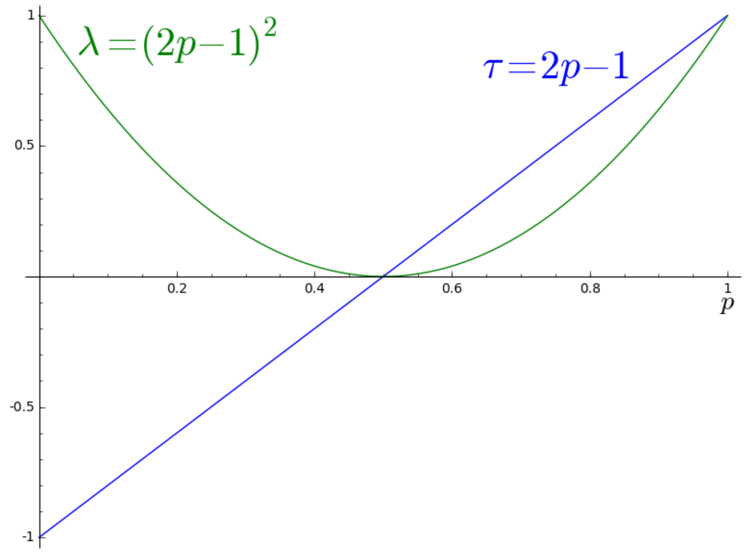


Figure 5.1: Connection between probability p, I/O-correlation τ, and potential λ

Note that the key k is the target of the attack. As long as it is unknown, the value of $p_{F,\alpha,\beta,\kappa}(k)$ is also unknown. Thus for cryptanalysis it makes sense to average the probabilities of a linear relation over all keys:

$$p_{F,\alpha,\beta,\kappa} := \frac{1}{2^{n+l}} \#\{(a,k) \in \mathbb{F}_2^n \times \mathbb{F}_2^l \mid \kappa(k) = \alpha(a) + \beta(F(a,k))\}. \qquad (2)$$

This average probability is determined by the definition of the cipher F alone, at least theoretically, neglecting efficiency. Calculating it however amounts to an exhaustion of all plaintexts and keys, and thus is unrealistic for a realistic cipher with large block lengths. We extend the definition for the "average case" also to I/O-correlation and potential:

$$\tau_{F,\alpha,\beta,\kappa} := 2p_{F,\alpha,\beta,\kappa} - 1,$$

$$\lambda_{F,\alpha,\beta,\kappa} := \tau^2_{F,\alpha,\beta,\kappa}.$$

Note that the I/O-correlation is also a mean value, but the potential is not!

Shamir already in Crypto 85 noticed that the S-boxes of DES admit linear relations with conspicuous probabilities. However it took another seven years until Matsui (after first attempts by Gilbert and Chassé 1990 with the cipher FEAL) succeeded in making systematic use of this observation. For estimating $\kappa(k)$ he proceeded as follows (in the case $p_{F,\alpha,\beta,\kappa} > \frac{1}{2}$; in the case $p_{F,\alpha,\beta,\kappa} < \frac{1}{2}$ take the bitwise complement, in the case $p_{F,\alpha,\beta,\kappa} = \frac{1}{2}$ the method is useless):

1. **Collect** N pairs of plaintexts and corresponding ciphertexts $(a_1, c_1), \ldots, (a_N, c_N)$.
2. **Count** the number
$$t := \#\{i = 1, \ldots, N \mid \alpha(a_i) + \beta(c_i) = 0\}.$$
3. **Decide** by majority depending on t:
   - If $t > \frac{N}{2}$, estimate $\kappa(k) = 0$.
   - If $t < \frac{N}{2}$, estimate $\kappa(k) = 1$.

The case $t = \frac{N}{2}$ is worthless, however scarce—we might randomize the decision between 0 and 1.

If we detect a linear relation whose probability differs from $\frac{1}{2}$ in a sufficient way, then this procedure will have a good success probability for sufficiently large N. This allows to reduce the number of unknown key bits by 1, applying elimination.

As a theoretical result from these considerations we'll get a connection between the number N of needed plaintext blocks and the success probability, see Table 5.4.

The more linear relations with sufficiently high certainty the attacker finds, the more she can reduce the size of the remaining key space until finally an exhaustion becomes feasible. A concrete example in Section 5.7 will illustrate this.

Example

For a concrete example with $n = l = 4$ we consider the Boolean map f that is given by the values in Table 5.1—by the way this is the S-box S_0 of Lucifer—and define the bitblock cipher

$$F\colon \mathbb{F}_2^4 \times \mathbb{F}_2^4 \longrightarrow \mathbb{F}_2^4 \quad \text{by } F(a,k) := f(a+k).$$

x	$y = f(x)$	x_4	$y_1 + y_2 + y_4$
0 0 0 0	1 1 0 0	0	0
0 0 0 1	1 1 1 1	1	1
0 0 1 0	0 1 1 1	0	0
0 0 1 1	1 0 1 0	1	1
0 1 0 0	1 1 1 0	0	0
0 1 0 1	1 1 0 1	1	1
0 1 1 0	1 0 1 1	0	0
0 1 1 1	0 0 0 0	1	0
1 0 0 0	0 0 1 0	0	0
1 0 0 1	0 1 1 0	1	1
1 0 1 0	0 0 1 1	0	1
1 0 1 1	0 0 0 1	1	1
1 1 0 0	1 0 0 1	0	0
1 1 0 1	0 1 0 0	1	1
1 1 1 0	0 1 0 1	0	0
1 1 1 1	1 0 0 0	1	1

Table 5.1: An S-box for $f\colon \mathbb{F}_2^4 \longrightarrow \mathbb{F}_2^4$ and two linear forms (the S-box S_0 of Lucifer)

a	$a + k$	c	$\alpha(a)$	$\beta(c)$	$\alpha(a) + \beta(c)$
0010	1010	0011	0	1	1
0101	1101	0100	1	1	0
1010	0010	0111	0	0	0

Table 5.2: Estimating a key bit after Matsui

We encrypt using the key k = `1000` (that we'll attack later as a test case). For a linear relation we consider the linear forms

$$\alpha(a) = a_4, \quad \beta(c) = c_1 + c_2 + c_4, \quad \kappa(k) = k_4.$$

In Section 5.2 we'll see that with these linear forms the relation $\kappa(k) = \alpha(a) + \beta(c)$ for F has a quite large probability. Table 5.2 shows the ciphertexts belonging to three plaintexts a (that later we'll assume as known plaintexts). The values of c are taken from Table 5.1. The number t of observed values 0 of $\alpha(a) + \beta(c)$ is $t = 2$. Hence the majority decision gives the estimate $k_4 = 0$ (being in cheat mode we know it's correct).

How successful will this procedure be in general? We have to analyse the problems:

1. How to find linear relations of sufficiently high probabilities?

2. Since in general bitblock ciphers consist of several rounds we ask:
   (a) How to find useful linear relations for the round function of an iterated bitblock cipher?
   (b) How to combine these over the rounds as a linear relation for the complete cipher?
   (c) How to calculate the probability of a combined linear relation for the complete cipher from the probabilities for the single rounds?

The answer to the first question and part (a) of the second one is: from the linear spectrum, see Section 5.3, that is by Fourier analysis, see Appendix D. The following partial questions lead to the analysis of linear paths, see Section 5.5, and the cumulation of probabilities, see Proposition 7. For (c) finally we only find a coarse rule of thumb.

Fourier analysis is quite efficient if the cost (time and space) is considered as function of the input size. Unfortunately this grows exponentially with the dimension. Therefore Fourier analysis soon becomes infeasible for dimensions more than 10. For serious block ciphers we have dimensions, or block and key sizes, of 64 or 128 bits, far out of reach.

At first sight this objection concerns also question 2 (a). However the single rounds usually consist of processing much smaller pieces, the S-boxes, in parallel. Hence one tries to reduce the problem to the analysis of the S-boxes, and this is feasible: Even AES uses S-boxes of dimension 8 only.

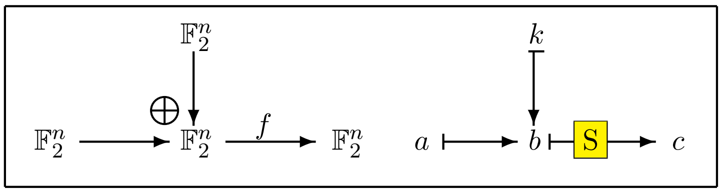


Figure 5.2: A (much too) simple example—The graphics here and later represent the map f sometimes by the S-box S in the elementwise assignments.

5.2 Example A: A One-Round-Cipher

We consider examples that are much too simple for real world applications but illustrate the principles of linear cryptanalysis in an easily intelligible way. We always assume round functions of the type $f(a+k)$, that is we add the key—or an n-bit part of it—to the plaintext before applying a bijective S-box $f\colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$. This is a quite special method of bringing the key into play but nevertheless realistic. The paradigmatic sample ciphers LUCIFER, DES, and AES do so, the term used with AES [1] is "key-alternating cipher structure".

The simplest model is encryption by the formula

$$c = f(a+k),$$

see Figure 5.2. This example is pointless because one block of known plaintext gives a solution for k:

$$k = f^{-1}(c) + a.$$

Note that the attacker knows the inverse map f^{-1} that is part of the decryption algorithm. (One-way encryption methods that assume that f^{-1} is not efficiently deducible from f are the subject of another part of cryptography, see Part III, Chapter 6, of these lecture notes.)

The somewhat more involved example A stops this attack:

$$c = f(a + k^{(0)}) + k^{(1)}$$

(see Figure 5.3). This is the simplest example for which the method of linear cryptanalysis makes sense: Let (α, β) be a pair of linear forms with

$$\beta \circ f(x) \overset{p}{\approx} \alpha(x), \tag{3}$$

where the symbol $\overset{p}{\approx}$ reads as "equal with probability p", or in other words

$$p = p_{f,\alpha,\beta} := \frac{1}{2^n} \cdot \#\{x \in \mathbb{F}_2^n \mid \beta \circ f(x) = \alpha(x)\}.$$

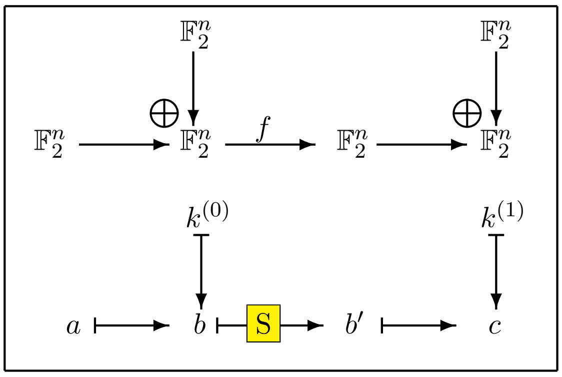


Figure 5.3: Example A

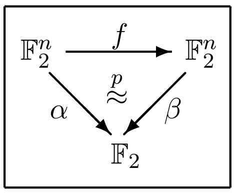


Figure 5.4: Diagram for an "approximative" linear relation

The diagram in Figure 5.4 illustrates Formula (3). Note that the linear form κ of the general theory is implicit in the present context: Since the key bits are simply added to plaintext and ("intermediary") ciphertext we have $\kappa = \alpha$ for $k^{(0)}$, and $\kappa = \beta$ for $k^{(1)}$, hence $\kappa(k^{(0)}, k^{(1)}) = \alpha(k^{(0)}) + \beta(k^{(1)})$.

How does this scenario fit the general situation from Chapter 2? In example A we have

- key length $l = 2n$, key space $\mathbb{F}_2^{2n}$, and keys of the form $k = (k^{(0)}, k^{(1)})$ with $k^{(0)}, k^{(1)} \in \mathbb{F}_2^n$.
- The cipher is defined by the map
$$F\colon \mathbb{F}_2^n \times \mathbb{F}_2^n \times \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n, \quad (a, k^{(0)}, k^{(1)}) \mapsto f(a + k^{(0)}) + k^{(1)}.$$
- The linear form $\kappa\colon \mathbb{F}_2^n \times \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ is $\kappa(k^{(0)}, k^{(1)}) = \alpha(k^{(0)}) + \beta(k^{(1)})$.

Hence the probability of a linear relation for a fixed key $k = (k^{(0)}, k^{(1)})$ is

$$\begin{aligned}
p_{F,\alpha,\beta,\kappa}(k) &= \frac{1}{2^n} \cdot \#\{a \in \mathbb{F}_2^n \mid \kappa(k) = \alpha(a) + \beta(F(a,k))\} \\
&= \frac{1}{2^n} \cdot \#\{a \in \mathbb{F}_2^n \mid \alpha(k^{(0)}) + \beta(k^{(1)}) = \alpha(a) + \beta(f(a + k^{(0)}) + k^{(1)})\} \\
&= \frac{1}{2^n} \cdot \#\{a \in \mathbb{F}_2^n \mid \alpha(k^{(0)}) = \alpha(a) + \beta(f(a + k^{(0)}))\},
\end{aligned}$$

where we omitted $\beta(k^{(1)})$ that occurs on both sides of the equation inside the curly set brackets.

This expression is independent of $k^{(1)}$, and the slightly rewritten equation

$$p_{F,\alpha,\beta,\kappa}(k) = \frac{1}{2^n} \cdot \#\{a \in \mathbb{F}_2^n \mid \alpha(a + k^{(0)}) = \beta(f(a + k^{(0)}))\}$$

shows that it assumes the same value for all $k^{(0)}$: With a also $a + k^{(0)}$ runs through all of $\mathbb{F}_2^n$ for a fixed $k^{(0)}$. Therefore this value must agree with the mean value over all k:

$$p_{F,\alpha,\beta,\kappa}(k) = p_{F,\alpha,\beta,\kappa} = \frac{1}{2^n} \cdot \#\{x \in \mathbb{F}_2^n \mid \alpha(x) = \beta(f(x))\} = p.$$

This consideration shows:

Proposition 3 *In the scenario of example A the probability $p_{F,\alpha,\beta,\kappa}(k)$ assumes the same value*

$$p = \frac{1}{2^n} \cdot \#\{x \in \mathbb{F}_2^n \mid \alpha(x) = \beta(f(x))\}$$

for all keys $k \in \mathbb{F}_2^{2n}$. In particular p coincides with the mean value from Equation (2)*.*

Using the notations from Figure 5.3 we have

$$\begin{aligned} \beta(c) &= \beta(b' + k^{(1)}) = \beta(b') + \beta(k^{(1)}) \\ &\overset{p}{\approx} \alpha(b) + \beta(k^{(1)}) = \alpha(a + k^{(0)}) + \beta(k^{(1)}) = \alpha(a) + \alpha(k^{(0)}) + \beta(k^{(1)}). \end{aligned}$$

This yields a linear relation for the bits of the key $k = (k_1, k_2)$:

$$\alpha(k^{(0)}) + \beta(k^{(1)}) \overset{p}{\approx} \alpha(a) + \beta(c).$$

Treating the complementary relation

$$\beta \circ f(x) \overset{1-p}{\approx} \alpha(x) + 1$$

in an analoguous way we get:

Proposition 4 *In the scenario of example A let (α, β) be a pair of linear forms for f with probability p as in Formula* (3)*. Then $\hat{p} = \max\{p, 1 - p\}$ is the success probability for determing a single key bit by this linear relation given* one *known plaintext block.*

a	b	b'	c	$\alpha(a)+\beta(c)$
0000	1000	0010	0011	1
0001	1001	0110	0111	1
0010	1010	0011	0010	0
0011	1011	0001	0000	1
0100	1100	1001	1000	1
0101	1101	0100	0101	1
0110	1110	0101	0100	1
0111	1111	1000	1001	1
1000	0000	1100	1101	1
1001	0001	1111	1110	1
1010	0010	0111	0110	1
1011	0011	1010	1011	1
1100	0100	1110	1111	1
1101	0101	1101	1100	1
1110	0110	1011	1010	1
1111	0111	0000	0001	0

Table 5.3: A linear relation for the key bits

Example

Take $n = 4$, and for f take the S-box S_0 of LUCIFER. As the two rightmost columns of Table 5.1 show the linear relation defined by (α, β), where $\alpha(x) = x_4$ and $\beta(y) = y_1 + y_2 + y_4$, has probability $p_{f,\alpha,\beta} = \frac{14}{16} = \frac{7}{8}$ (providing strong evidence that the designers of LUCIFER weren't aware of linear cryptanalysis).

As concrete round keys take $k_0 = \texttt{1000}$ and $k_1 = \texttt{0001}$. Table 5.3, running through all possible 16 plaintexts, shows that $\alpha(a) + \beta(c)$ assumes the value $1 = \alpha(k_0) + \beta(k_1)$ for this partial sum of key bits exactly 14 times—as expected.

How large is the success probability p_N of correctly estimating this partial sum, assuming $N = 1, 2, \ldots$ random known plaintexts from the set of 2^n possible plaintexts? (For given linear forms α and β with $p = p_{f,\alpha,\beta}$.) This is exactly the scenario of the hypergeometric distribution (for an explanation of the hypergeometric distribution see Appendix E). Therefore we have:

Proposition 5 *In example A let (α, β) be a pair of linear forms that defines a linear relation for f with probability p. Then the success probability for determining a key bit by this linear relation from N known plaintexts is the cumulated probability $p_N = p_N^{(s)}$ of the hypergeometric distribution with parameters 2^n, $s = \hat{p} \cdot 2^n$, and N where $\hat{p} = \max\{p, 1 - p\}$.*

If we neglect exact mathematical reasoning and work with asymptotic

$N\lambda$	1	2	3	4	...	8	9
p_N	84,1%	92,1%	95,8%	97,7%	...	99,8%	99,9%

Table 5.4: Dependence of the success probability on the number of known plaintexts

approximations (as is common in applied statistics), then we can replace the hypergeometric distribution by the normal distribution. The usual (quite vaguely stated) conditions for this approximation are “p not too different from $\frac{1}{2}$, $N \ll 2^n$, but N not too small.” This gives the formula

$$p_N \approx \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{\sqrt{N\lambda}} e^{-t^2/2}\, dt, \tag{4}$$

where $\lambda = (2p-1)^2$ is the potential of the linear relation. The values associated with the normal distribution are well-known and yield Table 5.4. Instead of the approximation by the normal distribution we could directly use the hypergeometric distribution. This would, in particular for small N, give a more precise value but not a closed formula as simple as (4).

To get a success probability of about 95% we need $N \approx \frac{3}{\lambda}$ known plaintexts according to the table. In the concrete example above we had $p = \frac{7}{8}$, hence $\lambda = \frac{9}{16}$, and the number of known plaintexts needed for a 95% success probability is $N \approx 5$. Using Table 5.2 we succeeded with only $N = 3$ plaintexts. This is not a great surprise because the a-priori probability of this success is about 90% (for $N\lambda = \frac{27}{16} \approx 1,68\ldots$).

In this example the condition “N not too small” for the approximation by the normal distribution is more than arguable. However determining the exact values for the hypergeometric distribution is easy: Consider an urn containing 16 balls, 14 black ones and 2 white ones, and draw 3 balls by random. Then the probability of all of them being black is $\frac{26}{40}$, the probability of two being black and one being white is $\frac{13}{40}$. Hence the probability of at least two balls being black is $\frac{39}{40} = 97,5\%$. This is clearly more than the 90% from the approximation (4). The remaining probabilities are $\frac{1}{40}$ for exactly one black ball, and 0 for three white balls.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	16	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
1	8	6	6	8	8	6	6	8	8	6	6	8	8	14	6	8
2	8	10	8	6	4	6	8	6	6	12	6	8	10	8	6	8
3	8	12	10	6	12	8	10	6	6	6	8	8	10	10	8	8
4	8	8	4	8	8	8	8	4	10	6	6	6	10	6	10	10
5	8	10	10	12	8	10	6	8	10	8	4	10	10	8	8	6
6	8	10	8	10	8	10	8	10	8	10	8	2	8	10	8	10
7	8	8	10	6	8	8	2	6	8	8	10	6	8	8	10	6
8	8	8	6	10	6	10	8	8	4	8	10	10	10	10	12	8
9	8	10	8	10	6	4	10	8	8	6	8	6	6	8	10	4
10	8	6	10	8	6	8	8	10	6	4	8	6	12	6	6	8
11	8	12	8	8	6	6	6	10	10	6	10	10	8	8	8	12
12	8	8	10	10	6	10	8	4	6	6	8	8	4	8	6	10
13	8	6	12	6	6	8	10	8	10	8	6	8	8	10	12	8
14	8	6	10	12	10	4	8	6	8	10	10	8	10	8	8	10
15	8	8	8	8	10	6	6	10	4	8	4	8	6	6	10	10

Table 5.5: Approximation table of the S-box S_0 of LUCIFER. Row and column indices are linear forms represented by integers. To get the probabilities divide by 16.

5.3 Approximation Table, Correlation Matrix, and Linear Spectrum of a Boolean Map

Linear relations for a Boolean map (or S-box) $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^q$ are true with certain frequencies (or probabilities). We collect these frequencies in a matrix of size $2^n \times 2^q$, called the **approximation table** of f. This table gives, for each pair (α, β) of linear forms, the number of arguments x where $\beta \circ f(x) = \alpha(x)$. Table 5.5 shows the approximation table of the S-box S_0 of LUCIFER. The entry 16 in the upper left corner says that the relation $0 = 0$ is true in all 16 possible cases. At the same time 16 is the common denominator by which we have to divide all other entries to get the probabilities. In the general case the upper left corner would be 2^n. The remaining entries of the first column (corresponding to $\beta = 0$) are 8 because each non-zero linear form α takes the value 0 in exactly half of all cases, that is 8 times. (In the language of linear algebra we express this fact as: The kernel of a linear form $\neq 0$ is a subspace of dimension $n - 1$.) For the first row an analogous argument is true—provided that f is bijective (or balanced).

The **correlation matrix** and the **linear spectrum** (also called linear profile or linearity profile—not to be confused with the linear complexity profile of a bit sequence that is defined by linear feedback shift registers and

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	$-\frac{1}{4}$	$-\frac{1}{4}$	0	0	$-\frac{1}{4}$	$-\frac{1}{4}$	0	0	$-\frac{1}{4}$	$-\frac{1}{4}$	0	0	$\frac{3}{4}$	$-\frac{1}{4}$	0
2	0	$\frac{1}{4}$	0	$-\frac{1}{4}$	$-\frac{1}{2}$	$-\frac{1}{4}$	0	$-\frac{1}{4}$	$-\frac{1}{4}$	$\frac{1}{2}$	$-\frac{1}{4}$	0	$\frac{1}{4}$	0	$-\frac{1}{4}$	0
3	0	$\frac{1}{2}$	$\frac{1}{4}$	$-\frac{1}{4}$	$\frac{1}{2}$	0	$\frac{1}{4}$	$-\frac{1}{4}$	$-\frac{1}{4}$	$-\frac{1}{4}$	0	0	$\frac{1}{4}$	$\frac{1}{4}$	0	0
4	0	0	$-\frac{1}{2}$	0	0	0	0	$-\frac{1}{2}$	$\frac{1}{4}$	$-\frac{1}{4}$	$-\frac{1}{4}$	$-\frac{1}{4}$	$\frac{1}{4}$	$-\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$
5	0	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{2}$	0	$\frac{1}{4}$	$-\frac{1}{4}$	0	$\frac{1}{4}$	0	$-\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	$-\frac{1}{4}$
6	0	$\frac{1}{4}$	0	$\frac{1}{4}$	0	$\frac{1}{4}$	0	$\frac{1}{4}$	0	$\frac{1}{4}$	0	2	0	$\frac{1}{4}$	0	$\frac{1}{4}$
7	0	0	$\frac{1}{4}$	$-\frac{1}{4}$	0	0	2	$-\frac{1}{4}$	0	0	$\frac{1}{4}$	$-\frac{1}{4}$	0	0	$\frac{1}{4}$	$-\frac{1}{4}$
8	0	0	$-\frac{1}{4}$	$\frac{1}{4}$	$-\frac{1}{4}$	$\frac{1}{4}$	0	0	$-\frac{1}{2}$	0	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{2}$	0
9	0	$\frac{1}{4}$	0	$\frac{1}{4}$	$-\frac{1}{4}$	$-\frac{1}{2}$	$\frac{1}{4}$	0	0	$-\frac{1}{4}$	0	$-\frac{1}{4}$	$-\frac{1}{4}$	0	$\frac{1}{4}$	$-\frac{1}{2}$
10	0	$-\frac{1}{4}$	$\frac{1}{4}$	0	$-\frac{1}{4}$	0	0	$\frac{1}{4}$	$-\frac{1}{4}$	$-\frac{1}{2}$	0	$-\frac{1}{4}$	$\frac{1}{2}$	$-\frac{1}{4}$	$-\frac{1}{4}$	0
11	0	$\frac{1}{2}$	0	0	$-\frac{1}{4}$	$-\frac{1}{4}$	$-\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$-\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	$\frac{1}{2}$
12	0	0	$\frac{1}{4}$	$\frac{1}{4}$	$-\frac{1}{4}$	$\frac{1}{4}$	0	$-\frac{1}{2}$	$-\frac{1}{4}$	$-\frac{1}{4}$	0	0	$-\frac{1}{2}$	0	$-\frac{1}{4}$	$\frac{1}{4}$
13	0	$-\frac{1}{4}$	$\frac{1}{2}$	$-\frac{1}{4}$	$-\frac{1}{4}$	0	$\frac{1}{4}$	0	$\frac{1}{4}$	0	$-\frac{1}{4}$	0	0	$\frac{1}{4}$	$\frac{1}{2}$	0
14	0	$-\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$	$-\frac{1}{2}$	0	$-\frac{1}{4}$	0	$\frac{1}{4}$	$\frac{1}{4}$	0	$\frac{1}{4}$	0	0	$\frac{1}{4}$
15	0	0	0	0	$\frac{1}{4}$	$-\frac{1}{4}$	$-\frac{1}{4}$	$\frac{1}{4}$	$-\frac{1}{2}$	0	$-\frac{1}{2}$	0	$-\frac{1}{4}$	$-\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$

Table 5.6: Correlation matrix of the S-box S_0 of LUCIFER. Row and column indices are linear forms represented by integers.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{9}{16}$	$\frac{1}{16}$	0
2	0	$\frac{1}{16}$	0	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	0	$\frac{1}{16}$	0	$\frac{1}{16}$	0
3	0	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{16}$	0	0
4	0	0	$\frac{1}{4}$	0	0	0	0	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$
5	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	0	$\frac{1}{16}$	$\frac{1}{16}$	0	$\frac{1}{16}$	0	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$
6	0	$\frac{1}{16}$	0	$\frac{1}{16}$	0	$\frac{1}{16}$	0	$\frac{1}{16}$	0	$\frac{1}{16}$	0	$\frac{9}{16}$	0	$\frac{1}{16}$	0	$\frac{1}{16}$
7	0	0	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{9}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{16}$
8	0	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{4}$	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	0
9	0	$\frac{1}{16}$	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$	0	$\frac{1}{16}$	$\frac{1}{16}$	0	$\frac{1}{16}$	$\frac{1}{4}$
10	0	$\frac{1}{16}$	$\frac{1}{16}$	0	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	0	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	0
11	0	$\frac{1}{4}$	0	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	0	0	0	$\frac{1}{4}$
12	0	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	0	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{4}$	0	$\frac{1}{16}$	$\frac{1}{16}$
13	0	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	0	$\frac{1}{16}$	0	$\frac{1}{16}$	0	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$
14	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{4}$	0	$\frac{1}{16}$	0	$\frac{1}{16}$	$\frac{1}{16}$	0	$\frac{1}{16}$	0	0	$\frac{1}{16}$
15	0	0	0	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	0	$\frac{1}{4}$	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$

Table 5.7: Linear spectrum of the S-box S_0 of LUCIFER. Row and column indices are linear forms represented by integers.

sometimes also called linearity profile) are the analogous matrices whose entries are the I/O-correlations or the potentials of the corresponding linear relations. The correlation matrix arises from the approximation table by first dividing the entries by 2^n (getting the probabilities p) and then transforming the probabilities to I/O-correlations by the formula $\tau = 2p - 1$. To get the linear spectrum we have to square the single entries of the correlation matrix.

For S_0 Table 5.6 shows the correlation matrix, and Table 5.7, the linear spectrum. Here again the first rows and columns hit the eye: The zeroes tell that a linear relation involving the linear form 0 has potential 0, hence is useless. The 1 in the upper left corner says that the relation $0 = 0$ holds for any arguments, but is useless too. In the previous subsection we picked the pair (α, β) where $\alpha(x) = x_4$ (represented by `0001` $\hat{=} 1$) and $\beta(y) = y_1 + y_2 + y_4$ (represented `1101` $\hat{=} 13$) in row 1, column 13. It assumes the maximum value $\frac{9}{16}$ for the potential that moreover also occurs at the coordinates $(6, 11)$ and $(7, 6)$. (We ignore the true, but useless, maximum value 1 in the upper left corner.)

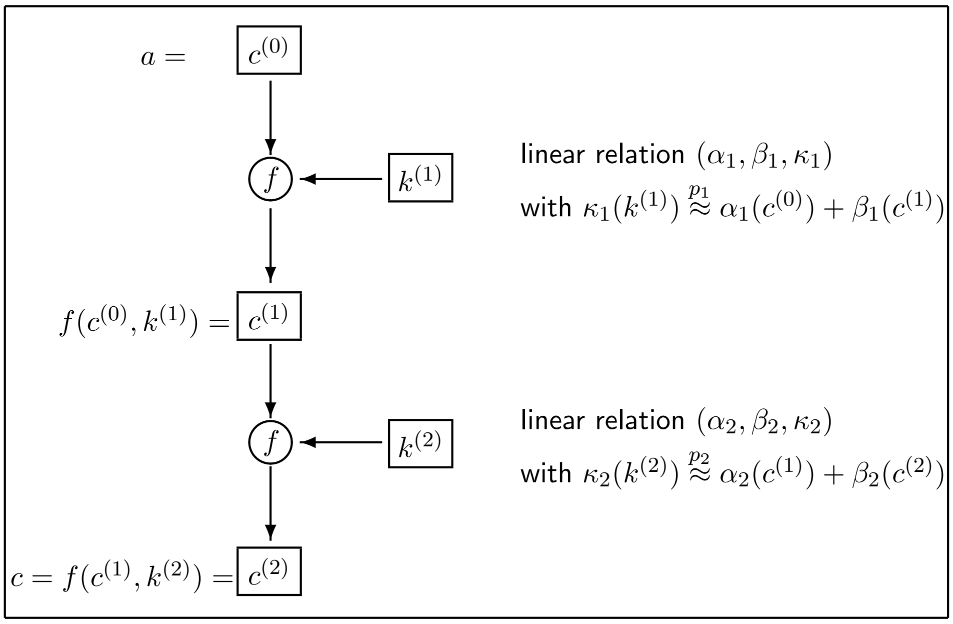

$a = \boxed{c^{(0)}}$
$\boxed{k^{(1)}}$
linear relation $(\alpha_1, \beta_1, \kappa_1)$
with $\kappa_1(k^{(1)}) \overset{p_1}{\approx} \alpha_1(c^{(0)}) + \beta_1(c^{(1)})$
$f(c^{(0)}, k^{(1)}) = \boxed{c^{(1)}}$
$\boxed{k^{(2)}}$
linear relation $(\alpha_2, \beta_2, \kappa_2)$
with $\kappa_2(k^{(2)}) \overset{p_2}{\approx} \alpha_2(c^{(1)}) + \beta_2(c^{(2)})$
$c = f(c^{(1)}, k^{(2)}) = \boxed{c^{(2)}}$

Figure 5.5: General two-round cipher

5.4 Example B: A Two-Round Cipher

As a next step we iterate the round map

$$f \colon \mathbb{F}_2^n \times \mathbb{F}_2^q \longrightarrow \mathbb{F}_2^n$$

of a bitblock cipher over two rounds using round keys $k^{(i)} \in \mathbb{F}_2^q$ as illustrated in Figure 5.5.

Remark In a certain sense example A already was a two-round cipher since we used two partial keys. Adding one more S-box at the right side of Figure 5.3 would be cryptologically irrelevant, because this non-secret part of the algorithm would be known to the cryptanalyst who simply could "strip it off" (that is, apply its inverse to the cipher text) and be left with example A. In a similar way we could interpret example B as a three-round cipher. However this would be a not so common counting of rounds.

We consider linear relations

$$\kappa_1(k^{(1)}) \overset{p_1}{\approx} \alpha_1(c^{(0)}) + \beta_1(c^{(1)})$$

with probability p_1, I/O-correlation $\tau_1 = 2p_1 - 1$, and potential $\lambda_1 = \tau_1^2$, and

$$\kappa_2(k^{(2)}) \overset{p_2}{\approx} \alpha_2(c^{(1)}) + \beta_2(c^{(2)})$$

with probability p_2, I/O-correlation $\tau_2 = 2p_2 - 1$, and potential $\lambda_2 = \tau_2^2$. We can combine these two linear relations if $\alpha_2 = \beta_1$, thereby getting a linear relation for some key bits expressed by the (known) plaintext $c^{(0)} = a$ and the ciphertext $c^{(2)} = c$,

$$\kappa_1(k^{(1)}) + \kappa_2(k^{(2)}) \overset{p}{\approx} \alpha_1(c^{(0)}) + \beta_2(c^{(2)}),$$

that holds with a certain probability p, and has I/O-correlation τ and potential λ, that in general depend on $k = (k^{(1)}, k^{(2)})$ and are difficult to determine. Therefore we again consider a simplified example B, see Figure 5.6. Encryption is done step by step by the formulas

$$b^{(0)} = a+k^{(0)}, a^{(1)} = f_1(b^{(0)}), b^{(1)} = a^{(1)}+k^{(1)}, a^{(2)} = f_2(b^{(1)}), c = a^{(2)}+k^{(2)}.$$

(Here f_1 is given by the S-box S_0, and f_2, by the S-box S_1 that could be identical with S_0. Note that we allow that the round functions of the different rounds differ. The reason is that usually the round function consists of several parallel S-boxes and the permutations direct an input bit through different S-boxes on its way through the rounds, see Section 5.7.)

As for example A adding some key bits after the last round prevents the "stripping off" of f_2. Comparing example B with the general settings in Chapter 2 we have:

- key length $l = 3n$, key space $\mathbb{F}_2^{3n}$, and keys of the form $k = (k^{(0)}, k^{(1)}, k^{(2)})$ with $k^{(0)}, k^{(1)}, k^{(2)} \in \mathbb{F}_2^n$.
- Encryption is defined by the map
$$\begin{array}{rcl} F\colon \mathbb{F}_2^n \times \mathbb{F}_2^n \times \mathbb{F}_2^n \times \mathbb{F}_2^n & \longrightarrow & \mathbb{F}_2^n, \\ (a, k^{(0)}, k^{(1)}, k^{(2)}) & \mapsto & f_2(f_1(a + k^{(0)}) + k^{(1)}) + k^{(2)}. \end{array}$$
- The linear form $\kappa\colon \mathbb{F}_2^n \times \mathbb{F}_2^n \times \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ is given by
$$\kappa(k^{(0)}, k^{(1)}, k^{(2)}) = \alpha(k^{(0)}) + \beta(k^{(1)}) + \gamma(k^{(2)}).$$

Here (α, β) is a linear relation for f_1 with probability p_1, I/O-correlation τ_1, and potential λ_1, and (β, γ), a linear relation for f_2 with probability p_2, I/O-correlation τ_2, and potential λ_2 (the same β since we want to combine the linear relations), where

$$\begin{array}{rcl} p_1 & = & \dfrac{1}{2^n} \cdot \#\{x \in \mathbb{F}_2^n \mid \beta \circ f_1(x) = \alpha(x)\} \\ p_2 & = & \dfrac{1}{2^n} \cdot \#\{y \in \mathbb{F}_2^n \mid \gamma \circ f_2(y) = \beta(y)\} \end{array}$$

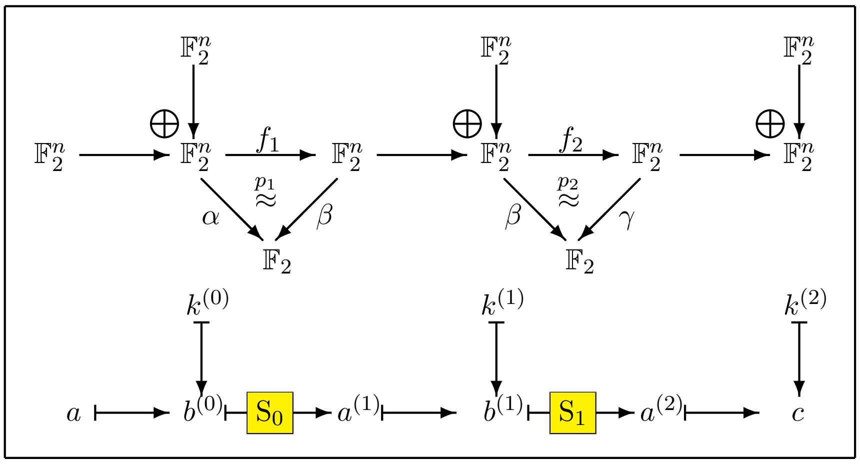


Figure 5.6: Example B

With the notations of Figure 5.6 we have

$$\begin{aligned}\gamma(c) = \gamma(a^{(2)}) + \gamma(k^{(2)}) &\overset{p_2}{\approx} \beta(b^{(1)}) + \gamma(k^{(2)}) = \beta(a^{(1)}) + \beta(k^{(1)}) + \gamma(k^{(2)})\\ &\overset{p_1}{\approx} \alpha(b^{(0)}) + \beta(k^{(1)}) + \gamma(k^{(2)}) = \alpha(a) + \alpha(k^{(0)}) + \beta(k^{(1)}) + \gamma(k^{(2)})\end{aligned}$$

Hence we get a linear relation for the key bits as a special case of Equation (1) in the form

$$\alpha(k^{(0)}) + \beta(k^{(1)}) + \gamma(k^{(2)}) \overset{p}{\approx} \alpha(a) + \gamma(c)$$

with a certain probability p that is defined by the formula

$$\begin{aligned}p &= p_{F,\alpha,\beta,\gamma}(k)\\ &= \frac{1}{2^n} \cdot \#\{a \in \mathbb{F}_2^n \mid \alpha(k^{(0)}) + \beta(k^{(1)}) + \gamma(k^{(2)}) = \alpha(a) + \gamma(F(a,k))\}.\end{aligned}$$

We try to explicitly determine p. As for the one-round case we first ask how p depends on k. Insert the definition of $F(a,k)$ into the defining equation inside the set brackets. Then $\gamma(k^{(2)})$ cancels out and we are left with

$$p_{F,\alpha,\beta,\gamma}(k) = \frac{1}{2^n} \cdot \#\{a \in \mathbb{F}_2^n \mid \alpha(k^{(0)}+a) + \beta(k^{(1)}) = \gamma(f_2(k^{(1)} + f_1(k^{(0)}+a)))\}.$$

This is independent of $k^{(2)}$, and for all $k^{(0)}$ assumes the same value

$$p_{F,\alpha,\beta,\gamma}(k) = \frac{1}{2^n} \cdot \#\{x \in \mathbb{F}_2^n \mid \alpha(x) = \beta(k^{(1)}) + \gamma(f_2(k^{(1)} + f_1(x)))\}$$

because $x = k^{(0)} + a$ runs through $\mathbb{F}_2^n$ along with a. This value indeed depends on k, but only on the middle component $k^{(1)}$. Now form the mean value $\bar{p} := p_{F,\alpha,\beta,\gamma}$ over all keys:

$$\bar{p} = \frac{1}{2^{2n}} \cdot \#\{(x, k^{(1)}) \in \mathbb{F}_2^{2n} \mid \alpha(x) = \beta(k^{(1)}) + \gamma(f_2(k^{(1)} + f_1(x)))\}.$$

Inside the brackets we see the expression $\gamma(f_2(k^{(1)}+f_1(x)))$, and we know:

$$\gamma(f_2(k^{(1)}+f_1(x))) = \begin{cases} \beta(k^{(1)}+f_1(x)) & \text{with probability } p_2, \\ 1+\beta(k^{(1)}+f_1(x)) & \text{with probability } 1-p_2. \end{cases}$$

Here "probability p_2" means that the statement is true for $p_2 \cdot 2^{2n}$ of all possible $(x, k^{(1)}) \in \mathbb{F}_2^{2n}$. Substituting this we get

$$\bar{p} = \frac{1}{2^{2n}} \cdot \Big[p_2 \cdot \#\{(x,k^{(1)}) \in \mathbb{F}_2^{2n} \mid \alpha(x) = \beta(f_1(x))\}$$

$$+(1-p_2) \cdot \#\{(x,k^{(1)}) \in \mathbb{F}_2^{2n} \mid \alpha(x) \neq \beta(f_1(x))\}\Big]$$

where now the defining equations of both sets are also independent of $k^{(1)}$. We recognize the definition of p_1 and substitute it getting

$$\bar{p} = p_1p_2 + (1-p_1)(1-p_2) = 2p_1p_2 - p_1 - p_2 + 1.$$

This formula looks much more beautiful if expressed in terms of the I/O-correlations $\bar{\tau} = 2\bar{p} - 1$ and $\tau_i = 2p_i - 1$ for $i = 1$ and 2:

$$\bar{\tau} = 2\bar{p} - 1 = 4p_1p_2 - 2p_1 - 2p_2 + 1 = (2p_1 - 1)(2p_2 - 1) = \tau_1\tau_2.$$

In summary we have proved:

Proposition 6 *For example B we have:*

(i) *The probability $p_{F,\alpha,\beta,\gamma}(k)$ depends only on the middle component $k^{(1)}$ of the key $k = (k^{(0)}, k^{(1)}, k^{(2)}) \in \mathbb{F}_2^n \times \mathbb{F}_2^n \times \mathbb{F}_2^n$.*

(ii) *The mean value of these probabilities over all keys k is $p_{F,\alpha,\beta,\gamma} = \bar{p} = 2p_1p_2 - p_1 - p_2 + 1$.*

(iii) *The I/O-correlations and the potentials are multiplicative:*

$$\tau_{F,\alpha,\beta,\gamma} = \tau_1\tau_2 \quad \textit{and} \quad \lambda_{F,\alpha,\beta,\gamma} = \lambda_1\lambda_2.$$

In Matsui's test we face the decision whether to use the linear relation or its negation for estimating a bit. We can't do better than use the mean value $p_{F,\alpha,\beta,\gamma}$ since the key and the true probability $p_{F,\alpha,\beta,\gamma}(k)$ are unknown. This could be an error since these two probabilities might lie on different sides of $\frac{1}{2}$.

Example

Let $n = 4$, S_0 as in example A, and S_1 as given in Table 5.8 (in different order) as transition from column $b^{(1)}$ to column $a^{(2)}$. (By the way this is the second S-box of LUCIFER.) Consider the linear forms $\alpha \hat{=}$ `0001` and $\beta \hat{=}$ `1101` as before with $p_1 = \frac{7}{8}$, $\tau_1 = \frac{3}{4}$, $\lambda_1 = \frac{9}{16}$. Furthermore let $\gamma \hat{=}$ `1100`. Then the linear relation for f_2 defined by (β, γ) (see Table 5.9, row index

a	$b^{(0)}$	$a^{(1)}$	$b^{(1)}$	$a^{(2)}$	c	$\beta(b^{(1)})$	$\gamma(a^{(2)})$	$\alpha(a)+\gamma(c)$
0000	1000	0010	0011	1001	1111	1	1	0
0001	1001	0110	0111	0100	0010	0	1	1
0010	1010	0011	0010	1110	1000	0	0	1
0011	1011	0001	0000	0111	0001	0	1	1
0100	1100	1001	1000	1100	1010	1	0	1
0101	1101	0100	0101	1011	1101	0	1	1
0110	1110	0101	0100	0011	0101	1	0	1
0111	1111	1000	1001	1101	1011	0	0	0
1000	0000	1100	1101	1111	1001	1	0	1
1001	0001	1111	1110	1000	1110	0	1	1
1010	0010	0111	0110	0000	0110	1	0	1
1011	0011	1010	1011	1010	1100	0	1	1
1100	0100	1110	1111	0101	0011	1	1	0
1101	0101	1101	1100	0110	0000	0	1	1
1110	0110	1011	1010	0001	0111	1	0	1
1111	0111	0000	0001	0010	0100	1	0	0

Table 5.8: The data flow in the concrete example for B, and some linear forms

13, column index 12) has probability $p_2 = \frac{1}{4}$, I/O-correlation $\tau_2 = -\frac{1}{2}$, and potential $\lambda_2 = \frac{1}{4}$, the maximum possible value by Table 5.10. (Note that the linear profile of S_1 is more uniform than that of S_0.)

As concrete round keys take $k^{(0)} =$ `1000`, $k^{(1)} =$ `0001`—as before—, and $k^{(2)} =$ `0110`. We want to gain the bit $\alpha(k^{(0)})+\beta(k^{(1)})+\gamma(k^{(2)})$ (that in cheat mode we know is 0). Since $\tau_1\tau_2 < 0$ in the majority of cases $\alpha(a)+\gamma(c)$ should give the complementary bit 1. Table 5.8 shows that in 12 of 16 cases this prediction is correct. Therefore $1-p = \frac{3}{4}$, $p = \frac{1}{4}$, $\tau = -\frac{1}{2}$, $\lambda = \frac{1}{4}$. Remember that this value depends on the key component $k^{(1)}$. In fact it slightly deviates from the mean value

$$\bar{p} = 2\cdot\frac{7}{8}\cdot\frac{1}{4} - \frac{7}{8} - \frac{1}{4} + 1 = \frac{7}{16} - \frac{14}{16} - \frac{4}{16} + \frac{16}{16} = \frac{5}{16}.$$

Calculating the variation of the probability as function of the partial key $k^{(1)}$ we get the values $\frac{1}{4}$ and $\frac{3}{8}$ each 8 times, all lying on the "correct side" of $\frac{1}{2}$ and having the correct mean value $\frac{5}{16}$.

There are other "paths" from α to γ—we could insert any β in between. Calculating the mean probabilities yields—besides the already known $\frac{5}{16}$—three times $\frac{15}{32}$, eleven times exactly $\frac{1}{2}$, and even a single $\frac{17}{32}$ that lies on the "wrong" side of $\frac{1}{2}$. Thus only the one case we explicitly considered is really good.

As an alternative concrete example take $\beta \hat{=}$ `0001`. Here $\lambda_1 = \frac{1}{16}$, $p_1 = \frac{3}{8}$,

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	16	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
1	8	10	8	10	8	6	12	10	10	4	6	8	10	8	10	8
2	8	6	4	10	6	8	6	8	8	10	4	6	10	8	10	8
3	8	8	8	8	6	6	6	6	10	6	6	10	4	8	8	12
4	8	8	8	4	8	8	8	4	6	6	6	10	10	10	10	6
5	8	6	8	10	4	6	8	6	8	6	12	6	8	10	8	6
6	8	10	12	10	6	12	6	8	10	8	6	8	8	10	8	6
7	8	8	8	12	10	10	10	6	4	8	8	8	6	10	10	10
8	8	8	6	10	10	6	8	8	10	10	8	12	8	12	6	6
9	8	6	6	8	6	12	8	10	8	6	10	12	10	8	8	10
10	8	6	6	8	12	10	6	8	10	4	8	6	6	8	8	6
11	8	4	10	10	8	8	10	6	8	8	6	10	8	4	6	6
12	8	8	6	6	6	10	12	8	8	8	6	6	6	10	4	8
13	8	10	6	8	6	8	8	10	6	8	8	10	4	6	10	4
14	8	10	6	8	8	10	10	4	12	10	10	8	8	6	10	8
15	8	4	10	6	8	8	10	10	10	10	8	8	6	10	12	8

Table 5.9: Approximation table of the S-box S_1 of LUCIFER. Row and column indices are linear forms represented by integers. For the probabilities divide by 16.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	$\frac{1}{16}$	0	$\frac{1}{16}$	0	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	0	$\frac{1}{16}$	0	$\frac{1}{16}$	0
2	0	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	0	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	0	$\frac{1}{16}$	0
3	0	0	0	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	0	0	$\frac{1}{4}$
4	0	0	0	$\frac{1}{4}$	0	0	0	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$
5	0	$\frac{1}{16}$	0	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	0	$\frac{1}{16}$	0	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	0	$\frac{1}{16}$	0	$\frac{1}{16}$
6	0	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	0	$\frac{1}{16}$	0	$\frac{1}{16}$	0	0	$\frac{1}{16}$	0	$\frac{1}{16}$
7	0	0	0	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	0	0	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$
8	0	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{16}$	0	$\frac{1}{4}$	0	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$
9	0	$\frac{1}{16}$	$\frac{1}{16}$	0	$\frac{1}{16}$	$\frac{1}{4}$	0	$\frac{1}{16}$	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$
10	0	$\frac{1}{16}$	$\frac{1}{16}$	0	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	0	$\frac{1}{16}$	$\frac{1}{4}$	0	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$
11	0	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{16}$	0	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$
12	0	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	0	0	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	0
13	0	$\frac{1}{16}$	$\frac{1}{16}$	0	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$
14	0	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{16}$	0
15	0	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	0	0	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	0

Table 5.10: Linear profile of the S-box S_1 of LUCIFER. Row and column indices are linear forms represented by integers.

$\tau_1 = -\frac{1}{4}$, and $\lambda_2 = \frac{1}{16}$, $p_2 = \frac{5}{8}$, $\tau_2 = \frac{1}{4}$. Hence $\tau = -\frac{1}{16}$ and $p = \frac{15}{32}$. The target bit is $\alpha(k^{(0)}) + \beta(k^{(1)}) + \gamma(k^{(2)}) + 1 = 1$, and the success probability is $1 - p = \frac{17}{32}$. The mean value of p over all keys is $\frac{15}{32}$ for this β in coincidence with the key-specific value.

$$\begin{tikzcd}
 & \mathbb{F}_2^n \arrow[d, "\oplus"'] & & & & \mathbb{F}_2^n \arrow[d, "\oplus"'] & & & \mathbb{F}_2^n \arrow[d, "\oplus"'] \\
\mathbb{F}_2^n \arrow[r] & \mathbb{F}_2^n \arrow[rr, "f_1"] \arrow[dr, "\beta_0"'] & \overset{p_1}{\approx} & \mathbb{F}_2^n \arrow[r] \arrow[dl, "\beta_1"] & \cdots \arrow[r] & \mathbb{F}_2^n \arrow[rr, "f_r"] \arrow[dr, "\beta_{r-1}"'] & \overset{p_r}{\approx} & \mathbb{F}_2^n \arrow[r] \arrow[dl, "\beta_r"] & \mathbb{F}_2^n \\
 & & \mathbb{F}_2 & & & & \mathbb{F}_2 & &
\end{tikzcd}$$

Figure 5.7: Example C

5.5 Linear Paths

Consider the general case where the round map $f\colon \mathbb{F}_2^n \times \mathbb{F}_2^q \longrightarrow \mathbb{F}_2^n$ is iterated for r rounds with round keys $k^{(i)} \in \mathbb{F}_2^q$, in analogy with Figure 5.5. Let $(\alpha_i, \beta_i, \kappa_i)$ be a linear relation for round i. Let $\alpha_i = \beta_{i-1}$ for $i = 2, \ldots, r$. Set $\beta_0 := \alpha_1$. Then the chain $\beta = (\beta_0, \ldots, \beta_r)$ is called a **linear path** for the cipher.

For a simplified scenario, let's call it example C as a generalization of example B, again we'll derive a useful result on the probabilities. So we consider the special but relevant case where the round keys enter the algorithm in an additive way, see Figure 5.7.

Given a key $k = (k^{(0)}, \ldots, k^{(r)}) \in \mathbb{F}_2^{n\cdot(r+1)}$ we compose the encryption function F successively with the intermediate results

$$a^{(0)} = a \mid b^{(0)} = a^{(0)} + k^{(0)} \mid a^{(1)} = f_1(b^{(0)}) \mid b^{(1)} = a^{(1)} + k^{(1)} \mid \ldots$$

$$b^{(r-1)} = a^{(r-1)} + k^{(r-1)} \mid a^{(r)} = f_r(b^{(r-1)}) \mid b^{(r)} = a^{(r)} + k^{(r)} = c =: F(a,k)$$

The general formula is

$$b^{(i)} = a^{(i)} + k^{(i)} \text{ for } i = 0, \ldots, r,$$

$$a^{(0)} = a \text{ and } a^{(i)} = f_i(b^{(i-1)}) \text{ for } i = 1, \ldots, r.$$

We consider a linear relation

$$\kappa(k) \overset{p}{\approx} \beta_0(a) + \beta_r(c),$$

where

$$\kappa(k) = \beta_0(k^{(0)}) + \cdots + \beta_r(k^{(r)}),$$

and p is the probability

$$p_{F,\beta}(k) = \frac{1}{2^n} \cdot \#\{a \in \mathbb{F}_2^n \mid \sum_{i=0}^{r} \beta_i(k^{(i)}) = \beta_0(a) + \beta_r(F(a,k))\}$$

that depends on the key k. Denote the mean value of these probabilities over all k by q_r. It depends on $(f_1, \ldots, f_r)$ and on the linear path β:

$$q_r := \frac{1}{2^{n\cdot(r+2)}} \cdot \#\{a, k^{(0)}, \ldots, k^{(r)} \in \mathbb{F}_2^n \mid \sum_{i=0}^{r} \beta_i(k^{(i)}) = \beta_0(a) + \beta_r(F(a,k))\}.$$

Substitute $F(a,k) = a^{(r)} + k^{(r)} = f_r(b^{(r-1)}) + k^{(r)}$ into the defining equation of this set. Then $\beta_r(k^{(r)})$ cancels out, and we see that the count is independent of $k^{(r)}$. The remaining formula is

$$q_r = \frac{1}{2^{n\cdot(r+1)}} \cdot \#\{a, k^{(0)}, \ldots, k^{(r-1)} \in \mathbb{F}_2^n \mid \sum_{i=0}^{r-1} \beta_i(k^{(i)}) = \beta_0(a) + \beta_r(f_r(b^{(r-1)}))\}.$$

In this formula the probability p_r is hidden: We have

$$\beta_r(f_r(b^{(r-1)})) = \begin{cases} \beta_{r-1}(b^{(r-1)}) & \text{with probability } p_r, \\ 1 + \beta_{r-1}(b^{(r-1)}) & \text{with probability } 1 - p_r, \end{cases}$$

where "with probability p_r" means: in $p_r \cdot 2^{n\cdot(r+1)}$ of the $2^{n\cdot(r+1)}$ possible cases. Hence

$$\begin{aligned} q_r &= \frac{1}{2^{n\cdot(r+1)}} \cdot \Bigg[p_r \cdot \#\{a, k^{(0)}, \ldots, k^{(r-1)} \mid \sum_{i=0}^{r-1} \beta_i(k^{(i)}) = \beta_0(a) + \beta_{r-1}(b^{(r-1)})\} \\ &\quad + (1-p_r) \cdot \#\{a, k^{(0)}, \ldots, k^{(r-1)} \mid \sum_{i=0}^{r-1} \beta_i(k^{(i)}) = 1 + \beta_0(a) + \beta_{r-1}(b^{(r-1)})\} \Bigg] \\ &= p_r \cdot q_{r-1} + (1-p_r) \cdot (1 - q_{r-1}), \end{aligned}$$

for the final counts exactly correspond to the probabilities for $r-1$ rounds.

This is the perfect entry to a proof by induction, showing:

Proposition 7 (Matsuis Piling-Up Theorem) *In example C the mean value $p_{F,\beta}$ of the probabilities $p_{F,\beta}(k)$ over all keys $k \in \mathbb{F}_2^{n(r+1)}$ fulfills*

$$2p_{F,\beta} - 1 = \prod_{i=1}^{r} (2p_i - 1).$$

In particular the I/O-correlations and the potentials are multiplicative.

Proof. The induction starts with the trivial case $r = 1$ (or with the case $r = 2$ that we proved in Proposition 6).

From the previous consideration we conclude

$$2q_r - 1 = 4p_r q_{r-1} - 2p_r - 2q_{r-1} + 1 = (2p_r - 1)(2q_{r-1} - 1),$$

and the assertion follows by induction on r. $\diamond$

For real ciphers in general the round keys are *not* independent but derive from a "master key" by a specific key schedule. In practice however this effect is negligeable. The method of linear cryptanalysis follows the rule of thumb:

Along a linear path the potentials are multiplicative.

Proposition 7, although valid only in a special situation and somewhat imprecise for real life ciphers, gives a good impression of how the cryptanalytic advantage (represented by the potential) of linear approximations decreases with an increasing number of rounds; note that the product of numbers smaller than 1 (and greater than 0) decreases with the number of factors. This means that the security of a cipher against linear cryptanalysis is the better, the more rounds it involves.

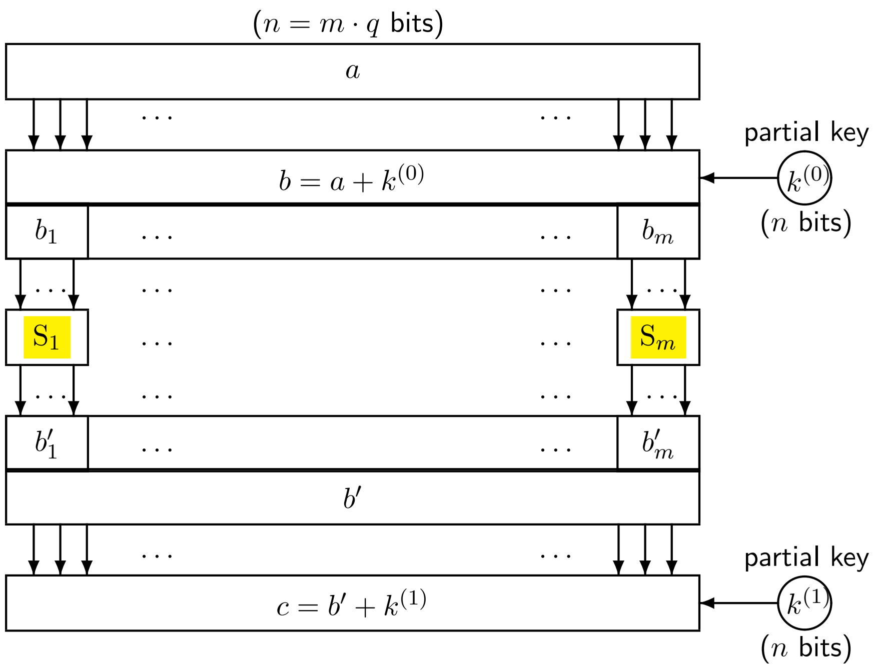


Figure 5.8: Example D, parallel arrangement of m S-boxes $\mathrm{S}_1, \ldots, \mathrm{S}_m$ of width q

5.6 Parallel Arrangement of S-Boxes

The round map of an SP-network usually involves several "small" S-boxes in a parallel arrangement. On order to analyze the effect of this construction we again consider a simple example D, see Figure 5.8.

Proposition 8 *Let $S_1, \ldots, S_m \colon \mathbb{F}_2^q \longrightarrow \mathbb{F}_2^q$ be Boolean maps, $n = m \cdot q$, and f, the Boolean map*

$$f \colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n, \; f(x_1, \ldots, x_m) = (S_1(x_1), \ldots, S_m(x_m)) \text{ for } x_1, \ldots, x_m \in \mathbb{F}_2^q.$$

Let (α_i, β_i) for $i = 1, \ldots, m$ be linear relations for S_i with probabilities p_i. Let

$$\begin{aligned} \alpha(x_1, \ldots, x_m) &= \alpha_1(x_1) + \cdots + \alpha_m(x_m) \\ \beta(y_1, \ldots, y_m) &= \beta_1(y_1) + \cdots + \beta_m(y_m) \end{aligned}$$

Then (α, β) is a linear relation for f with probability p given by

$$2p - 1 = (2p_1 - 1) \cdots (2p_m - 1).$$

Proof. We consider the case $m = 2$ only; the general case follows by a simple induction as for Proposition 7.

In the case $m = 2$ we have $\beta \circ f(x_1, x_2) = \alpha(x_1, x_2)$ if and only if

- *either* $\beta_1 \circ \mathrm{S}_1(x_1) = \alpha_1(x_1)$ and $\beta_2 \circ \mathrm{S}_2(x_2) = \alpha_2(x_2)$
- *or* $\beta_1 \circ \mathrm{S}_1(x_1) = 1 + \alpha_1(x_1)$ and $\beta_2 \circ \mathrm{S}_2(x_2) = 1 + \alpha_2(x_2)$.

Hence $p = p_1p_2 + (1-p_1)(1-p_2)$, and the assertion follows as for Proposition 6. $\diamond$

As a consequence the I/O-correlations and the potentials are multiplicative also for a parallel arrangement. At first view this might seem a strengthening of the security, but this appearance is deceiving! We cannot detain the attacker from choosing all linear forms as zeroes except the "best" one. And the zero forms have probabilities $p_i = 1$ and potentials 1. Hence the attacker picks a pair (α_j, β_j) with maximum potential, and then sets $\alpha(x_1, \ldots, x_m) = \alpha_j(x_j)$ and $\beta(y_1, \ldots, y_m) = \beta_j(y_j)$. In a certain sense she turns the other S-boxes, except S_j, "inactive". Then the complete linear relation inherits exactly the probability and the potential of the "active" S-box S_j.

Example

Once again we consider a concrete example with $m = 2$ and $q = 4$, hence $n = 8$. As S-boxes we take the ones from LUCIFER, S_0 at the left, and S_1 at the right, see Figure 5.8. For the left S-box S_0 we take the linear relation with $\alpha \mathrel{\hat{=}} \texttt{0001}$ and $\beta \mathrel{\hat{=}} \texttt{1101}$, that we know has probability $p_1 = \frac{7}{8}$, for the right S-Box S_1 we take the relation $(0, 0)$ with probability 1. The combined linear relation for $f = (\mathrm{S}_0, \mathrm{S}_1)$ then also has probability $p = \frac{7}{8}$ and potential $\lambda = \frac{9}{16}$, and we know that linear cryptanalysis with $N = 5$ pairs of plaintext and ciphertext has 95% success probability. We decompose all relevant bitblocks into bits:

plaintext: $a = (a_0, \ldots, a_7) \in \mathbb{F}_2^8$,

ciphertext: $c = (c_0, \ldots, c_7) \in \mathbb{F}_2^8$,

key: $k = (k_0, \ldots, k_{15}) \in \mathbb{F}_2^{16}$ where $(k_0, \ldots, k_7)$ serves as "initial key" (corresponding to $k^{(0)}$ in Figure 5.8), and $(k_8, \ldots, k_{15})$ as "final key" (corresponding to $k^{(1)}$).

Then $\alpha(a) = a_3$, $\beta(c) = c_0 + c_1 + c_3$, and $\kappa(k) = \alpha(k_0, \ldots, k_7) + \beta(k_8, \ldots, k_{15}) = k_3 + k_8 + k_9 + k_{11}$. Hence the target relation is

$$k_3 + k_8 + k_9 + k_{11} = a_3 + c_0 + c_1 + c_3.$$

We use the key $k = \texttt{1001011000101110}$ whose relevant bit is $k_3 + k_8 + k_9 + k_{11} = 1$, and generate five random pairs of plaintext and ciphertext, see Table 5.11. We see that for this example Matsui's algorithm guesses the relevant key bit correctly with no dissentient.

a	a_3	c	$c_0 + c_1 + c_3$	estimate
00011110	1	00000010	0	1
00101100	0	00111111	1	1
10110010	1	01011101	0	1
10110100	1	01010000	0	1
10110101	1	01010111	0	1

Table 5.11: Calculations for example D

index i	0	1	2	3	4	5	6	7
$\mathrm{P}(i)$	2	5	4	0	3	1	7	6

Table 5.12: Lucifer's permutation P

5.7 Mini-Lucifer

As a slightly more complex example we define a toy cipher "Mini-Lucifer" that employs the S-boxes and a permutation of the true Lucifer. Here is the construction, see Figure 5.9:

- Before and after each round map we add a partial key. We use two keys $k^{(0)}$ and $k^{(1)}$ in alternating order. They consist of the first or last 8 bits of the 16 bit master key. In particular for $r \geq 3$ the round keys are not independent.
- The round function consists of a parallel arrangement of the two S-boxes, as in the example of Section 5.6, followed by the permutation P.
- The permutation P maps a single byte (octet) to itself as defined in Table 5.12. As usual for SP-networks we omit it in the last round.

Up to now we ignored permutations in linear cryptanalysis. How do they influence the analysis?

Well, let f be a Boolean map, (α, β), a linear relation for f with probability p, and P, a permutation of the range of f. Then we set $\beta' = \beta \circ \mathrm{P}^{-1}$, a linear form, and immediately see that (α, β') is a linear relation for $\mathrm{P} \circ f$ with the same probability p:

$$\begin{aligned} p &= \frac{1}{2^n} \cdot \#\{x \in \mathbb{F}_2^n \mid \beta(f(x)) = \alpha(x)\} \\ &= \frac{1}{2^n} \cdot \#\{x \in \mathbb{F}_2^n \mid (\beta \circ \mathrm{P}^{-1})(\mathrm{P} \circ f(x)) = \alpha(x)\}. \end{aligned}$$

The assignment $\beta \mapsto \beta'$ simply permutes the linear forms β. In other words: appending a permutation to f permutes the columns of the approximation table, of the correlation matrix, and of the linear profile.

> *Inserting a permutation into the round function of an SP-network affects linear cryptanalysis in a marginal way only.*

We'll verify this assertion for a concrete example, and see how "marginal" the effect really is. By the way the same argument holds if we replace the permutation by a more general bijective linear map L: also $\beta \mapsto \beta \circ \mathrm{L}^{-1}$ permutes the linear forms.

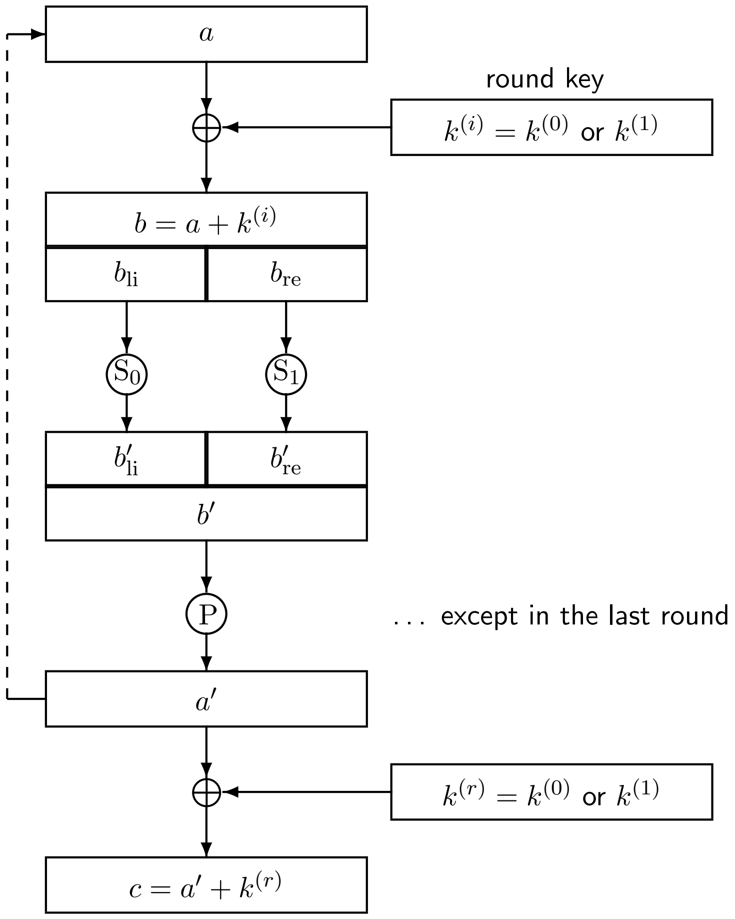


Figure 5.9: Mini-Lucifer

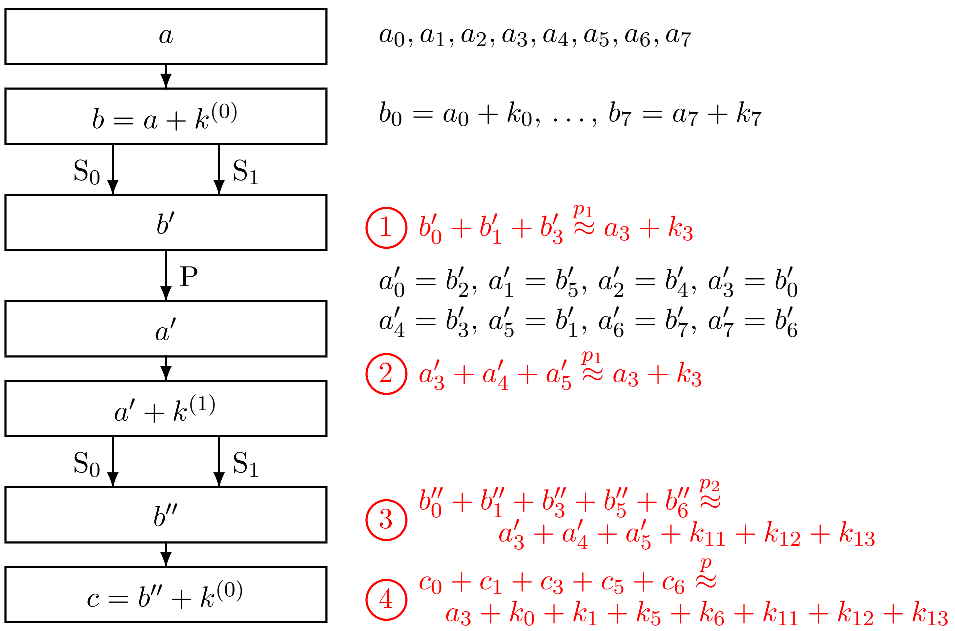

a
$b = a + k^{(0)}$
S_0
S_1
b'
P
a'
$a' + k^{(1)}$
S_0
S_1
b''
$c = b'' + k^{(0)}$
$a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7$
$b_0 = a_0 + k_0, \ldots, b_7 = a_7 + k_7$
$\textcircled{1}\ b'_0 + b'_1 + b'_3 \overset{p_1}{\approx} a_3 + k_3$
$a'_0 = b'_2,\ a'_1 = b'_5,\ a'_2 = b'_4,\ a'_3 = b'_0$
$a'_4 = b'_3,\ a'_5 = b'_1,\ a'_6 = b'_7,\ a'_7 = b'_6$
$\textcircled{2}\ a'_3 + a'_4 + a'_5 \overset{p_1}{\approx} a_3 + k_3$
$\textcircled{3}\ b''_0 + b''_1 + b''_3 + b''_5 + b''_6 \overset{p_2}{\approx} a'_3 + a'_4 + a'_5 + k_{11} + k_{12} + k_{13}$
$\textcircled{4}\ c_0 + c_1 + c_3 + c_5 + c_6 \overset{p}{\approx} a_3 + k_0 + k_1 + k_5 + k_6 + k_{11} + k_{12} + k_{13}$

Figure 5.10: Mini-Lucifer with 2 rounds

Example

The concrete example is specified in Figure 5.10. The relation 1, namely

$$\beta(b') \overset{p_1}{\approx} \alpha(a + k^{(0)})$$

holds with probability $p_1 = \frac{7}{8}$ between $\alpha \mathrel{\hat{=}} \texttt{0001}$ and $\beta \mathrel{\hat{=}} \texttt{1101}$. The permutation P transforms it to the relation 2, namely

$$\beta \circ \mathrm{P}^{-1}(a') \overset{p_1}{\approx} \alpha(a + k^{(0)}) = \alpha(a) + \alpha(k^{(0)}).$$

But P also distributes the bits from the left-hand side of the relation over the two S-boxes of the next round. So the cryptanalytic trick of letting only one S-box per round become active works only for the first round.

> *Inserting a permutation into the round function of an SP-network has the effect that linear cryptanalysis has to deal with more than one parallel S-box becoming active in later rounds.*

We'll soon see in the example that this effect reduces the potential. The relevant bits a'_3, a'_4, a'_5, or, after adding the key, $a'_3 + k_{11}$, $a'_4 + k_{12}$, $a'_5 + k_{13}$, split as input to the left S-box S_0 of the second round (namely $a'_3 + k_{11}$), and to the right one, S_1 (namely $a'_4 + k_{12}$ and $a'_5 + k_{13}$).

On the left-hand side, for S_0, the linear form for the input is $\beta'_1 \mathrel{\hat{=}} \texttt{0001} \mathrel{\hat{=}} 1$, on the right-hand side, for S_1, we have $\beta'_2 \mathrel{\hat{=}} \texttt{1100} \mathrel{\hat{=}} 12$. From the linear profile of S_0 we see that the maximum possible potential for β'_1 is $\lambda'_2 = \frac{9}{16}$ with $p'_2 = \frac{7}{8}$, assumed for $\gamma_1 \mathrel{\hat{=}} 13 \mathrel{\hat{=}} \texttt{1101}$.

For β'_2 the maximum potential is $\lambda''_2 = \frac{1}{4}$. Having two choices we choose $\gamma_2 \mathrel{\hat{=}} 6 \mathrel{\hat{=}} \texttt{0110}$ with probability $p''_2 = \frac{3}{4}$. The combined linear relation with $\beta'(x) = \beta'_1(x_0, \ldots, x_3) + \beta'_2(x_4, \ldots, x_7)$ and, on the output side, $\gamma(y) = \gamma_1(y_0, \ldots, y_3) + \gamma_2(y_4, \ldots, y_7)$ has I/O-correlation

$$2p_2 - 1 = (2p'_2 - 1)(2p''_2 - 1) = \frac{3}{8}$$

by Proposition 8, hence $p_2 = \frac{11}{16}$, $\lambda_2 = \frac{9}{64}$.

The relation between $\beta'(a'+k^{(1)})$ and $\gamma(b'')$ is labelled by 3 in Figure 5.10, namely

$$\gamma(b'') \overset{p_2}{\approx} \beta'(a' + k^{(1)}) = \beta'(a') + \beta'(k^{(1)}).$$

Combining 2 and 3 (and cancelling k_3) yields the relation

$$\gamma(c) + \gamma(k^{(0)}) = \gamma(c + k^{(0)}) = \gamma(b'') \overset{p}{\approx} \alpha(a) + \alpha(k^{(0)}) + \beta'(k^{(1)}),$$

labelled by 4 in the figure, whose probability p is given by Proposition 7 since the two round keys are independent. We get

$$2p - 1 = (2p_1 - 1)(2p_2 - 1) = \frac{3}{4} \cdot \frac{3}{8} = \frac{9}{32},$$

whence $p = \frac{41}{64}$. The corresponding potential is $\lambda = \frac{81}{1024}$.

The number N of needed plaintexts for a 95% success probability follows from the approximation in Table 5.4:

$$N = \frac{3}{\lambda} = \frac{1024}{27} \approx 38.$$

Note that there are only 256 possible plaintexts at all.

In the example the success probability derived from the product of the I/O-correlations (or of the potentials) of all active S-boxes. We had luck since in this example the involved partial keys were independent. In the general case this is not granted. Nevertheless the cryptanalyst relies on the empirical evidence and ignores the dependencies, trusting the rule of thumb:

> *The success probability of linear cryptanalysis is (approximately) determined by the multiplicativity of the I/O-correlations (or of the potentials) of all the active S-boxes along the considered path (including all of its ramifications).*

The restriction in this rule of thumb concerns the *success probability* of linear cryptanalysis but not the *course of action.* The cryptanalyst is right if and only if she succeeds, no matter whether her method had an exact mathematical foundation for all details.

Now we obtained a single bit. So what?

Of course we may find more relations, and detect more key bits. However we have to deal with smaller and smaller potentials, and face an increasing danger of hitting a case where the probability for the concrete (target) key lies on the “wrong” side of $\frac{1}{2}$. Moreover we run into a multiple test situation reusing the same known plaintexts several times. This enforces an unpleasant adjustment of the success probabilities.

5.8 Systematic Search for Linear Relations

The search for useful linear relations over several rounds has no general elegant solution. The published examples often use linear paths that more or less appear from nowhere, and it is not evident that they are the best ones.

Let n be the block length of the cipher, and r, the number of rounds. Then for each round the choice is between 2^n linear formes, making a total of $2^{n(r+1)}$ choices. This number also specifies the cost of determining the best relation by complete search. There are some simplifications that however don't reduce the order of magnitude of the cost:

- In the first round consider only linear forms that activate only one S-box.
- Then choose the next linear form such that it activates the least possible number of S-boxes of the next round (with high, but not necessarily maximum potential).
- If one of the relations in a linear path has probability $\frac{1}{2}$, or I/O-correlation 0, then the total I/O-correlation is 0 by multiplicativity, and this path may be neglected. The same is true componentwise if the linear forms split among the S-boxes of the respective round. However this negligence could introduce errors since we deal with average probabilities not knowing the key-dependent ones.

For our 2-round example with Mini-Lucifer the systematic search is feasible by pencil and paper or by a Sage or Python script. Our example has the following characteristics:

- $\alpha = (\alpha_1, \alpha_2)$ with $\alpha_1 \mathrel{\hat{=}} 1 \mathrel{\hat{=}} \texttt{0001}$ and $\alpha_2 \mathrel{\hat{=}} 0 \mathrel{\hat{=}} \texttt{0000}$ (α_1 was formerly denoted α. Now for uniformity we make both components of all linear forms explicit and index them by 1 and 2.)
- $\beta = (\beta_1, \beta_2)$ with $\beta_1 \mathrel{\hat{=}} 13 \mathrel{\hat{=}} \texttt{1101}$ and $\beta_2 \mathrel{\hat{=}} 0 \mathrel{\hat{=}} \texttt{0000}$
- $\beta' = (\beta'_1, \beta'_2)$ with $\beta'_1 \mathrel{\hat{=}} 1 \mathrel{\hat{=}} \texttt{0001}$, $\beta'_2 \mathrel{\hat{=}} 12 \mathrel{\hat{=}} \texttt{1100}$
- $\gamma = (\gamma_1, \gamma_2)$ with $\gamma_1 \mathrel{\hat{=}} 13 \mathrel{\hat{=}} \texttt{1101}$, $\gamma_2 \mathrel{\hat{=}} 6 \mathrel{\hat{=}} \texttt{0110}$
- $\tau_1 = \frac{3}{4}$, $\tau'_2 = \frac{3}{4}$, $\tau''_2 = \frac{1}{2}$, $\tau_2 = \frac{3}{8}$, $\tau = \frac{9}{32}$, $p = \frac{41}{64} = 0,640625$
- $c_0 + c_1 + c_3 + c_5 + c_6 \overset{p}{\approx} a_3 + k_0 + k_1 + k_5 + k_6 + k_{11} + k_{12} + k_{13}$

An alternative choice of γ_2 is $\gamma_2 \mathrel{\hat{=}} 14 \mathrel{\hat{=}} \texttt{1110}$; this yields a linear path with the characteristics

- $\alpha \mathrel{\hat{=}} (1, 0)$, $\beta \mathrel{\hat{=}} (13, 0)$, $\beta' \mathrel{\hat{=}} (1, 12)$, $\gamma \mathrel{\hat{=}} (13, 14)$

  - $\tau = -\frac{9}{32}$, $p = \frac{23}{64} = 0,359375$
  - $c_0+c_1+c_3+c_4+c_5+c_6 \overset{p}{\approx} a_3+k_0+k_1+k_4+k_5+k_6+k_{11}+k_{12}+k_{13}$

The systematic search finds two even "better" linear paths, characterized by

- $\alpha \mathrel{\hat{=}} (8,0)$, $\beta \mathrel{\hat{=}} (8,0)$, $\beta' \mathrel{\hat{=}} (1,0)$, $\gamma \mathrel{\hat{=}} (13,0)$
  - $\tau = -\frac{3}{8}$, $p = \frac{5}{16} = 0,3125$
  - $c_0 + c_1 + c_3 \overset{p}{\approx} a_0 + k_1 + k_3 + k_{11}$
- $\alpha \mathrel{\hat{=}} (15,0)$, $\beta \mathrel{\hat{=}} (8,0)$, $\beta' \mathrel{\hat{=}} (1,0)$, $\gamma \mathrel{\hat{=}} (13,0)$
  - $\tau = -\frac{3}{8}$, $p = \frac{5}{16} = 0,3125$
  - $c_0 + c_1 + c_3 \overset{p}{\approx} a_0 + a_1 + a_2 + a_3 + k_2 + k_{11}$

that do not completely exhaust the potential of the single S-boxes but on the other hand activate only one S-box of the second round, and thereby show the larger potential $\lambda = \frac{9}{64}$. Thus we get a 95% success probability with only

$$N = \frac{3}{\lambda} = \frac{64}{3} \approx 21$$

known plaintexts for determining one bit.

The designer of a cipher should take care that in each round the active bits fan out over as many S-boxes as possible. The inventors of AES, Daemen and Rijmen call this design approach "wide-trail strategy". The design of AES strengthens this effect by involving a linear map instead of a mere permutation, thereby replacing the "P" of an SP-network by an "L".

Figure 5.11 shows an example of a linear path with all its ramifications.

Example (Continued)

For an illustration of the procedure we generate 25 pairs of known plaintexts and corresponding ciphertexts using the key $k \mathrel{\hat{=}}$ `1001011000101110`. The target key bits are

$$\begin{aligned} b_0 &= k_0 + k_1 + k_5 + k_6 + k_{11} + k_{12} + k_{13} \\ b_1 &= k_0 + k_1 + k_4 + k_5 + k_6 + k_{11} + k_{12} + k_{13} \\ b_2 &= k_1 + k_3 + k_{11} \\ b_3 &= k_2 + k_{11} \end{aligned}$$

that we know in cheat mode are $b_0 = 1$, $b_1 = 1$, $b_2 = 1$, $b_3 = 0$. We use all four good relations at the same time without fearing the possible reduction of the success probability. All of these relations assert the probable equality of the bits

$$\begin{aligned}
b_0 &\overset{p}{\approx} c_0 + c_1 + c_3 + c_5 + c_6 + a_3 \\
b_1 &\overset{p}{\approx} 1 + c_0 + c_1 + c_3 + c_4 + c_5 + c_6 + a_3 \\
b_2 &\overset{p}{\approx} 1 + c_0 + c_1 + c_3 + a_0 \\
b_3 &\overset{p}{\approx} 1 + c_0 + c_1 + c_3 + a_0 + a_1 + a_2 + a_3
\end{aligned}$$

each with its individual corresponding probability p. For the last three of these sums we have to take the complementary bits since the corresponding I/O-correlations are negative (the probabilities are $< \frac{1}{2}$). This is done by adding the bit `1`.

Table 5.13 shows the results for these plaintext-ciphertext pairs. As we see our guess is correct for all four bits.

As a consequence of our analysis we get a system of four linear equations for the 16 unknown key bits:

$$\begin{aligned}
1 &= k_0 + k_1 + k_5 + k_6 + k_{11} + k_{12} + k_{13} \\
1 &= k_0 + k_1 + k_4 + k_5 + k_6 + k_{11} + k_{12} + k_{13} \\
1 &= k_1 + k_3 + k_{11} \\
0 &= k_2 + k_{11}
\end{aligned}$$

that allow us to reduce the number of keys for an exhaustion from $2^{16} = 65536$ to $2^{12} = 4096$. Note the immediate simplifications of the system: $k_{11} = k_2$ from the last equation, and $k_4 = 0$ from the first two.

As a cross-check we run some more simulations. The next four yield

- $15, 16, 19, 16$
- $15, 16, 13, 17$
- $15, 20, 19, 17$
- $19, 19, 20, 18$

correct guesses, and so on. Only run number 10 produced a wrong bit (the second one):

- $17, 12, 14, 17$

then again run number 25. Thus empirical evidence suggests a success probability of at least 90% in this scenario.

nr	plaintext	ciphertext	b_0	b_1	b_2	b_3
1	`00001111`	`00001010`	1	1	1	1
2	`00010001`	`11001110`	1	1	1	0
3	`00010110`	`11001001`	1	1	1	0
4	`00111101`	`10110010`	0	1	1	1
5	`01000000`	`11100111`	0	1	1	0
6	`01001000`	`01010111`	0	1	1	0
7	`01001100`	`11101010`	1	1	1	0
8	`01001101`	`01011100`	1	1	1	0
9	`01001111`	`01111010`	1	1	1	0
10	`01100111`	`00110011`	0	1	0	0
11	`10000011`	`11110100`	0	1	1	1
12	`10010011`	`01101011`	1	1	1	0
13	`10011000`	`01100111`	0	1	1	0
14	`10101011`	`11011001`	1	1	1	0
15	`10110001`	`11001000`	1	1	0	0
16	`10110010`	`10100100`	1	0	1	1
17	`10110110`	`11000100`	0	1	0	0
18	`10111001`	`11000001`	1	0	0	0
19	`10111101`	`10111111`	1	1	0	0
20	`11000100`	`01001111`	1	1	1	0
21	`11000111`	`00111111`	1	1	1	0
22	`11011111`	`11011010`	1	1	1	1
23	`11100000`	`11101110`	0	0	0	0
24	`11100100`	`01110011`	1	0	0	0
25	`11110101`	`11110101`	1	0	1	0
		true bit:	1	1	1	0
		correct guesses:	17	20	18	20

Table 5.13: Plaintext/ciphertext pairs for Mini-Lucifer

Analysis over Four Rounds

Now let's explore how an increasing number of rounds impedes linear cryptanalysis.

Consider the toy cipher Mini-Lucifer over four rounds. Searching an optimal linear path over four rounds is somewhat expensive, so we content ourselves with extending the best example from the two round case, the third one, over two additional rounds. Slightly adapting the notation we get:

- for the first round $\beta_0 = \alpha \hat{=} (8,0)$ and $\beta_1 \hat{=} (8,0)$ (the "old" β) with $\tau_1 = -\frac{1}{2}$,
- for the second round (applying the permutation P to β_1) $\beta_1' \hat{=} (1,0)$ and $\beta_2 \hat{=} (13,0)$ (the "old" γ) with $\tau_2 = \frac{3}{4}$,
- for the third round $\beta_2' \hat{=} (1,12)$ and $\beta_3 \hat{=} (13,6)$ with $\tau_3 = \frac{3}{8}$,
- for the fourth round $\beta_3' \hat{=} (5,13)$ and $\beta = \beta_4 \hat{=} (3,12)$ (the "new" β) with $\tau_4 = -\frac{1}{4}$.

Figure 5.11 shows this linear path with its ramifications.

The repeated round keys we used are not independent. Therefore multiplicativity of I/O-correlations is justified by the rule of thumb only yielding an approximate value for the I/O-correlation of the linear relation (α, β) over all of the four rounds:

$$\tau \approx \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{3}{8} \cdot \frac{1}{4} = \frac{9}{256} \approx 0,035.$$

The other characteristics are

$$p \approx \frac{265}{512} \approx 0,518, \quad \lambda \approx \frac{81}{65536} \approx 0,0012, \quad N \approx \frac{65536}{27} \approx 2427,$$

the last one being the number of needed known plaintexts for a 95% success probability.

Comparing this with the cost of exhaustion over all 65536 possible keys we seem to have gained an advantage. However there are only 256 different possible plaintexts all together. So linear cryptanalysis completely lost its sense by the increased number of rounds.

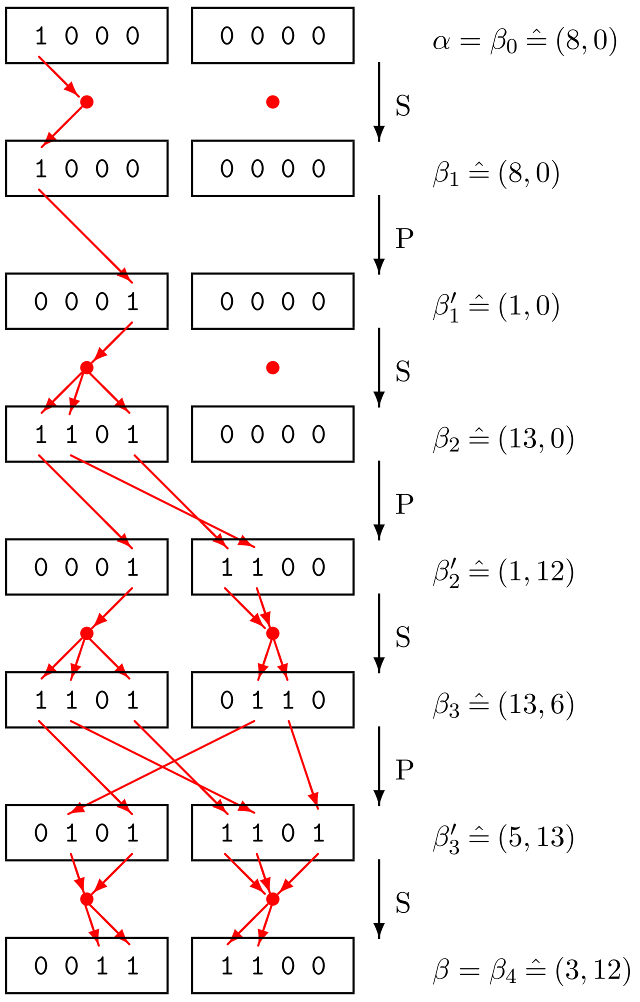

1 0 0 0
0 0 0 0
$\alpha = \beta_0 \mathrel{\hat{=}} (8,0)$
S
1 0 0 0
0 0 0 0
$\beta_1 \mathrel{\hat{=}} (8,0)$
P
0 0 0 1
0 0 0 0
$\beta_1' \mathrel{\hat{=}} (1,0)$
S
1 1 0 1
0 0 0 0
$\beta_2 \mathrel{\hat{=}} (13,0)$
P
0 0 0 1
1 1 0 0
$\beta_2' \mathrel{\hat{=}} (1,12)$
S
1 1 0 1
0 1 1 0
$\beta_3 \mathrel{\hat{=}} (13,6)$
P
0 1 0 1
1 1 0 1
$\beta_3' \mathrel{\hat{=}} (5,13)$
S
0 0 1 1
1 1 0 0
$\beta = \beta_4 \mathrel{\hat{=}} (3,12)$

Figure 5.11: A linear path with ramifications ("trail"). For S the linear form in the range is *chosen* (for high potential), indicated by a red dot. For P the linear form in the range results by applying the permutation.

5.9 The Idea of Differential Cryptanalysis

Differential cryptanalysis has some similarities with linear cryptanalysis but instead of linear relations it uses approximations of Boolean maps by linear structures (see Appendix C). The idea is to consider a difference vector before applying a round map, and its possible values thereafter. Sequences of difference vectors that fit together over all the rounds of an iterated bitblock cipher are called a **differential path** or a **characteristic** [Biham/Shamir 1990]. The potential of a differential path is approximated by the product of the potentials of the single steps. A **differential hull** or a **differential** [Lai/Massey/Murphy 1991] is the collection of all paths between a given input difference (of the entire cipher) and a given output difference. The success of differential cryptanalysis relies on an analoguous rule of thumb:

> *Along a differential path the differential potentials are multiplicative. The potential of a differential hull is approximated by the potential of a dominant differential path.*

This potential reflects the probability for getting an equation for some key bits.

Chapter 6

AES

The cipher AES ("Advanced Encryption Standard") is the successor of the obsolete DES. It was adopted after a thorough competitive selection procedure in 2001. The winner of the competiton was the Belgian algorithm Rijndael, henceforth called AES, sparing English speaking people the plight of correct pronunciation, and neglecting a small difference in the specifications: Rijndael contains some extended parameter options that are not standardized for AES.

AES is a multiple cipher with several rounds but not a Feistel cipher, not even an SP-network in the proper sense. The kernel map is based on an S-box that essentially is the multiplicative inversion in the finite field $\mathbb{F}_{256}$. For a comprehensive analysis of the nonlinear properties of this S-box see Appendix D.

The inventors Joan Daemen and Vincent Rijmen themselves published a book that provides a very comprehensive und comprehensible description of the method:

> Joan Daemen, Vincent Rijmen, The Design of Rijndael. AES – The Advanced Encryption Standard. Springer-Verlag, Berlin 2002. ISBN 3-540-42580-2.

(Note that Joan is a Flemish version of John.)

In this text we only give an introduction into the overall scheme and the kernel map.

6.1 The Structure of AES

Overview

Figures 6.1 and 6.2 give an overall impression of the scheme of AES and illustrate how the construction principles derived in former sections show up in AES.

- The block length is $n = 128$, the key length, $l = 128$, 192, or 256, the number of rounds, $r = 10$, 12 oder 14.
- As a first step of each round, and as a last step after the last round, a partial key is added to the current bitblock, making a total of $r + 1$ partial keys.
- The 128-bit "partial keys" $k^{(i)}$ are not partial keys in the proper sense but are extracted from the real key k by a somewhat involved procedure called "key expansion". In particular they are not stochastically independent.
- At the begin of each round, after adding the round key, the current 128-bit block is decomposed into 16 partial blocks each with 8 bits. The S-box $\mathrm{S} : \mathbb{F}_2^8 \longrightarrow \mathbb{F}_2^8$ is applied to each of these 16 blocks. The linear potential of the S-box is $\frac{1}{64}$, see Appendix D.
- The "diffusion" step consists of a permutation followed by a linear map. This step is slightly more complex than the standard for a pure SP-network as in Section 2.4.

A further remark on the key expansion: The selection of the "round keys" $k^{(i)}$ conceal the "real" key. Cryptanalytic approaches attack the "effective" key consisting of the collection of the round keys $k^{(i)}$. This is adequate for breaking the cipher—the real key is not needed. However the complexity of the key expansion might prevent exploiting a dependency of the different round keys, for instance in the case where the round keys are overlapping partial blocks of the "real" key.

Representation of the Bitblocks

AES operates on octets (8-bit bytes), that is, on the $\mathbb{F}_2$ vector space $\mathbb{F}_2^8$. Since the structure of this vector space as a field with 256 elements plays a crucial role in several steps of the algorithm identifying this vector space with the field $\mathbb{F}_{256}$ suggests itself. The exact identification map is given in Section 6.2.

AES operates on bitblocks of lengths that are multiples of 32. Plaintexts, ciphertexts, an intermediate results have 128 bits, the key length is 128, 192, or 256.

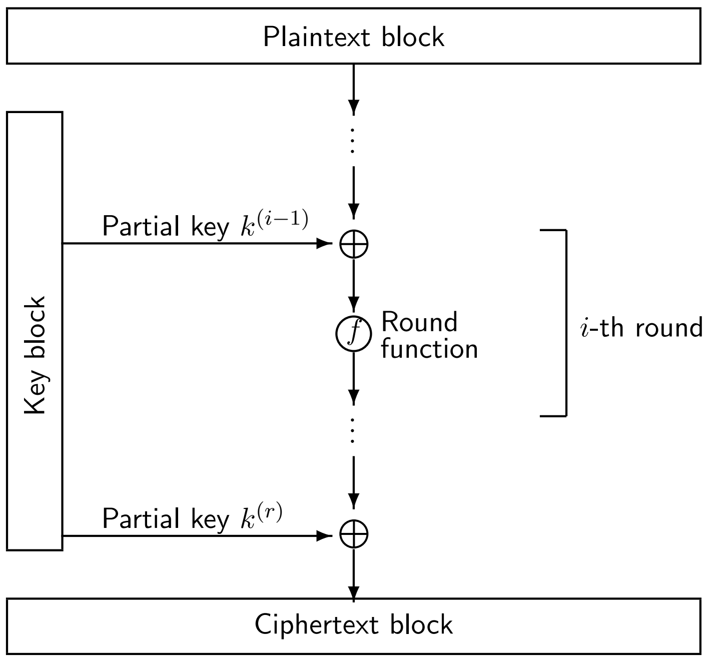


Figure 6.1: The overall scheme of AES

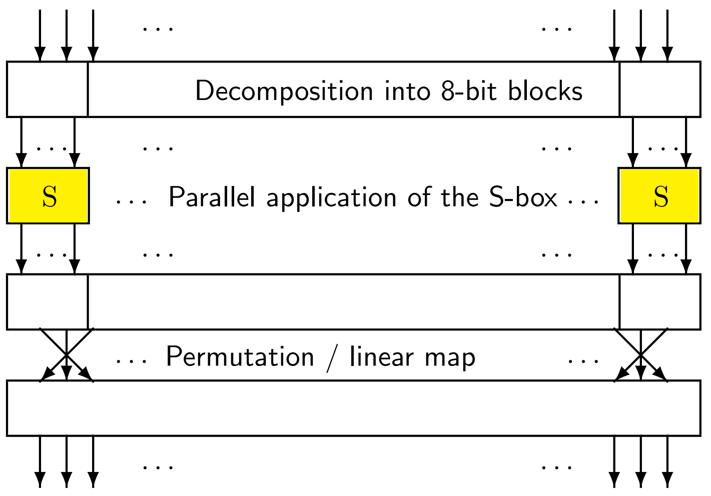

Decomposition into 8-bit blocks
S
. . . Parallel application of the S-box . . .
S
. . . Permutation / linear map . . .

Figure 6.2: The round function f of AES

Remark The only difference between Rijndael and AES is that the specification of Rijndael allows block and key lengths of 128, 160, 192, 224, 256 each. The AES standardization procedure makes no assertions about the security of Rijndael for the additional sizes of these parameters.

Interpret the 32-bit blocks as columns with 4 octets, that is, as elements of the 4-dimensional $\mathbb{F}_{256}$ vector space $\mathbb{F}_{256}^4$. So each block of length $n = 32 \cdot N_b$ is an N_b-tuple of such columns, or a $4 \times N_b$-matrix

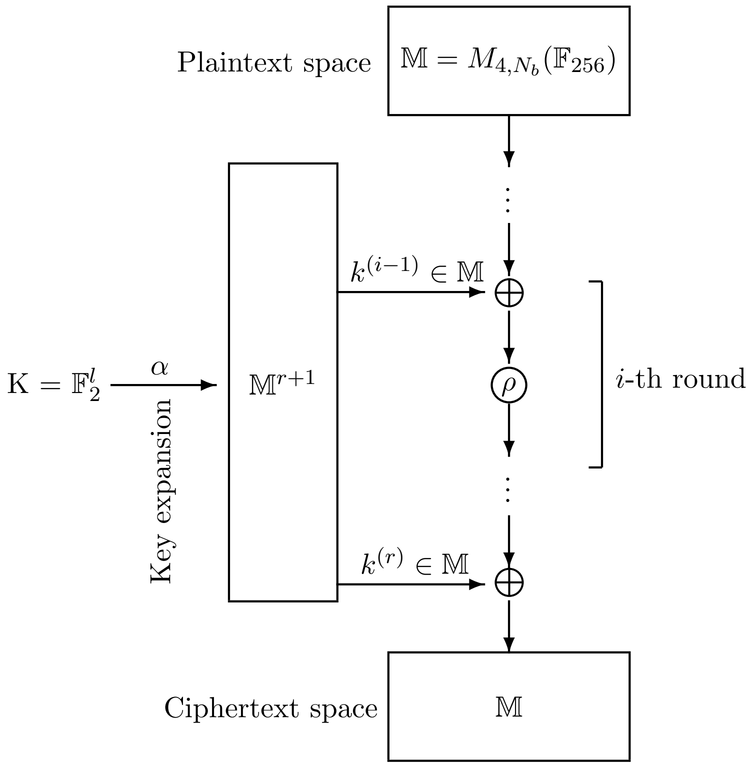
$$a = \begin{pmatrix} a_{0,0} & \ldots & a_{0,N_b-1} \\ \vdots & & \vdots \\ a_{3,0} & \ldots & a_{3,N_b-1} \end{pmatrix} \in M_{4,N_b}(\mathbb{F}_{256}) =: \mathbb{M}.$$

Analogously let the key length be $l = 32 \cdot N_k$; however we don't use the corresponding matrices but expand the keys and then decompose them into round keys $k^{(i)}$.

The specification of Rijndael uses $N_b, N_k \in \{4, 5, 6, 7, 8\}$, the specification of AES, $N_b = 4$, $N_k \in \{4, 6, 8\}$.

The Iteration Scheme

An AES encryption runs through r rounds. Each round consists of the (binary) addition of a round key and the kernel map ρ. In the last round the kernel map is slightly abbreviated—more on this later on. After the last round one more partial key is added. Therefore the key expansion has to produce $r + 1$ partial keys, each consisting of $32N_b$ bits, from the total of l key bits. The kernel map is invertible, and is the same for all rounds except the last one.



The number of rounds for Rijndael is specified as follows:

	N_b				
N_k	4	5	6	7	8
4	10	11	12	13	14
5	11	11	12	13	14
6	12	12	12	13	14
7	13	13	13	13	14
8	14	14	14	14	14

For AES with key lengths 128, 192, or 256 the number of rounds is 10, 12, or 14, tagged green in the table.

The Kernel Map

Each round of AES (or Rijndael) consists of four steps

1. `AddRoundKey`, the addition of the round key, see Figure 6.3.
2. `SubBytes`, a substitution that consists of parallel application of the S-box on each octet, see Figure 6.4,
3. `ShiftRows`, a permutation of each matrix row, see Figure 6.5,

4. `MixColums`, a linear map of each matrix column to itself, see Figure 6.6,

The kernel map ρ consists of steps 2 to 4. In the last round step 4, `MixColums`, is omitted. In this way the inverse map—decryption—has the same structure. The cryptographic strength is not affected by this omission.

Figures 6.3 to 6.6 are taken from the Wikipedia article on AES (for the case $N_b = N_k = 4$).

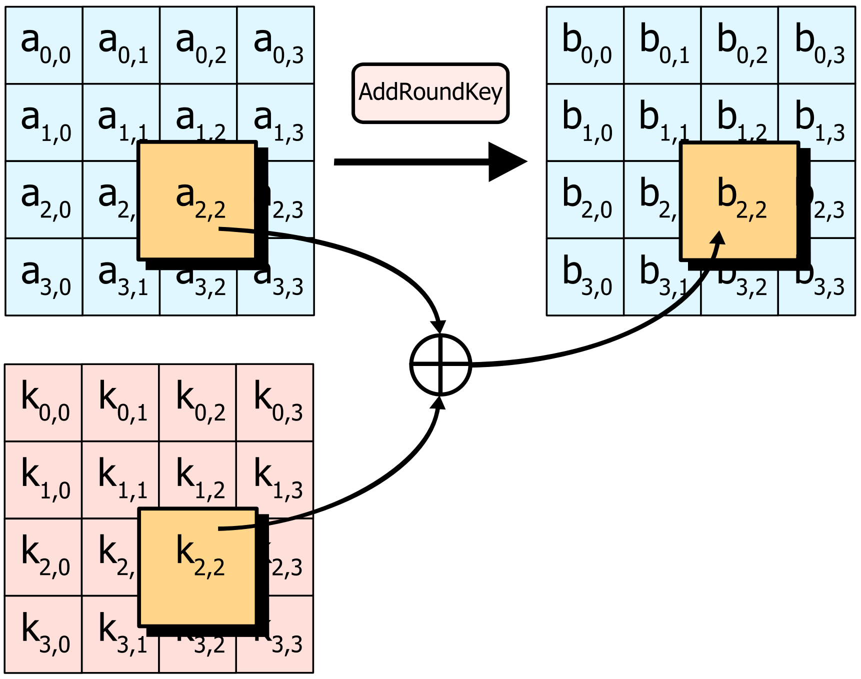

$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
AddRoundKey
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
$k_{0,0}$ $k_{0,1}$ $k_{0,2}$ $k_{0,3}$
$k_{1,0}$ $k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$k_{2,0}$ $k_{2,1}$ $k_{2,2}$ $k_{2,3}$
$k_{3,0}$ $k_{3,1}$ $k_{3,2}$ $k_{3,3}$

Figure 6.3: The operation of `AddRoundKey`

The Substitution Step `SubBytes`

Apply the S-box S_{RD} separately in parallel to all octets of the current state, that is, to all entries of the current matrix $b \in \mathbb{M}$. This S-box is the composition $S_{RD} = f \circ g$ of two maps, the inversion g in $\mathbb{F}_{256}$,

$$g\colon \mathbb{F}_{256} \longrightarrow \mathbb{F}_{256}, \quad g(x) = \begin{cases} x^{-1} & \text{for } x \neq 0, \\ 0 & \text{for } x = 0, \end{cases}$$

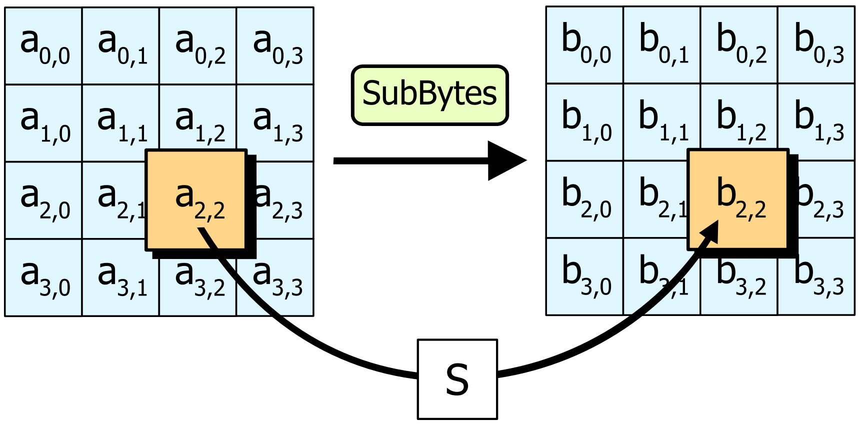


Figure 6.4: The operation of `SubBytes`

(whose nonlinearity properties we know already well), and the affine map

$$f\colon \mathbb{F}_2^8 \longrightarrow \mathbb{F}_2^8, \qquad \begin{pmatrix} x_7\\ x_6\\ x_5\\ x_4\\ x_3\\ x_2\\ x_1\\ x_0 \end{pmatrix} \mapsto \begin{pmatrix} 1&1&1&1&1&0&0&0\\ 0&1&1&1&1&1&0&0\\ 0&0&1&1&1&1&1&0\\ 0&0&0&1&1&1&1&1\\ 1&0&0&0&1&1&1&1\\ 1&1&0&0&0&1&1&1\\ 1&1&1&0&0&0&1&1\\ 1&1&1&1&0&0&0&1 \end{pmatrix} \begin{pmatrix} x_7\\ x_6\\ x_5\\ x_4\\ x_3\\ x_2\\ x_1\\ x_0 \end{pmatrix} + \begin{pmatrix} 0\\ 1\\ 1\\ 0\\ 0\\ 0\\ 1\\ 1 \end{pmatrix}.$$

This construction ensures that

- S_{RD} has no fixed point, that is, $\mathrm{S}_{\mathrm{RD}}(b) \neq b$ for all $b \in \mathbb{F}_{256}$,
- S_{RD} has no “anti fixed point”, that is, $\mathrm{S}_{\mathrm{RD}}(b) \neq \bar{b}$ for all $b \in \mathbb{F}_{256}$;

where $\bar{b} = (1-b_7, \ldots, 1-b_0)$ is the Boolean complement, that is, the bitwise logical negation.

The drawback of this modification of the inversion map is that the involutory property of g gets lost; therefore the decryption algorithm needs the implementation of another S-box $\mathrm{S}_{\mathrm{RD}}^{-1}$.

The Row Permutation `ShiftRows`

Each of the four rows of the current state—a $4 \times N_b$-matrix—gets cyclically shifted to the left by an individual amount; row i by C_i positions. The C_i depend on the block length in the following way:

N_b	C_0	C_1	C_2	C_3
4	0	1	2	3
5	0	1	2	3
6	0	1	2	3
7	0	1	2	4
8	0	1	3	4

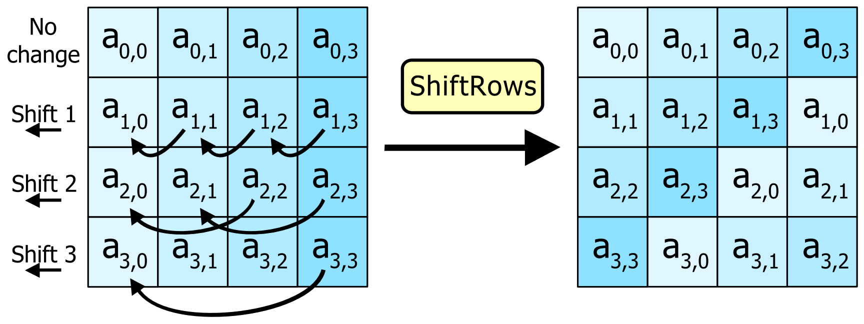


Figure 6.5: The operation of `ShiftRows`

The Linear Transformation of the Columns, `MixColumns`

The current state is multipled with a fixed 4×4-matrix column by column from the left; in other words, the state matrix $\in \mathbb{M}$ is multiplied from the left to generate some diffusion. This is the $\mathbb{F}_{256}$-linear map

$$\mu\colon \mathbb{F}_{256}^4 \longrightarrow \mathbb{F}_{256}^4,$$

specified by the matrix

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$

whose entries must be interpreted as octets in hexadecimal representation, for example $03 = (00000011) \in \mathbb{F}_2^8$.

Decryption

The complete AES (or Rijndael) encryption is the composition

$$F_k = \kappa_r \circ \underbrace{\tau \circ \sigma \circ \kappa_{r-1}}_{i=r} \circ \ldots \circ \underbrace{[\mu \circ \tau \circ \sigma \circ \kappa_{i-1}]}_{i=1,\ldots,r-1} \circ \ldots \circ \kappa_0$$

of maps $\mathbb{M} \longrightarrow \mathbb{M}$. Here κ_i is the addition of the i-th round key, τ = `ShiftRows`, σ = `SubBytes`, and μ = `MixColumns`, and the kernel map is $\rho = \mu \circ \tau \circ \sigma$.

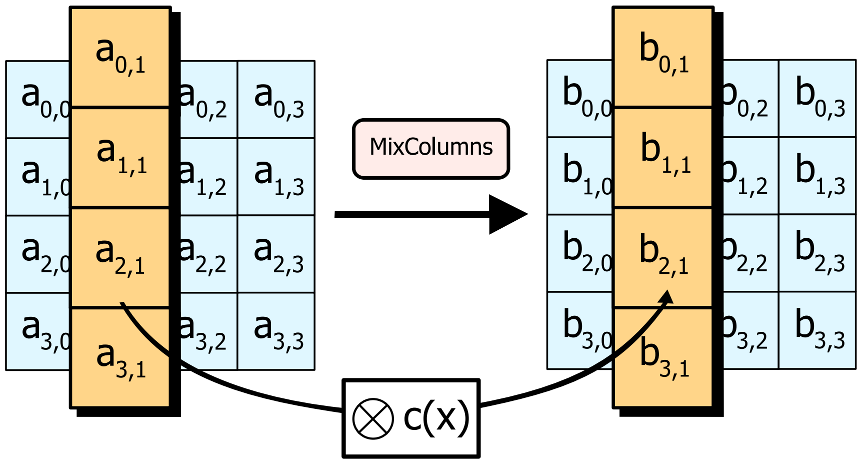

$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
MixColumns
$\otimes$ c(x)
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

Figure 6.6: The operation of `MixColumns`

For decryption we need the inverse map that looks as follows:

$$F_k^{-1} = \kappa_0^{-1} \circ \ldots \circ [\kappa_{i-1}^{-1} \circ \sigma^{-1} \circ \tau^{-1} \circ \mu^{-1}] \circ \ldots \circ \kappa_{r-1}^{-1} \circ \sigma^{-1} \circ \tau^{-1} \circ \kappa_r^{-1}.$$

The promised structural analogy with F_k follows after some transformations:

- We have $\kappa_i^{-1} = \kappa_i$ for all i.
- Because σ acts on the matrix entries, in the same way on each entry, we have $\tau \circ \sigma = \sigma \circ \tau$.
- Finally $\kappa_i \circ \mu(x) = \mu(x) + k^{(i)} = \mu(x + \mu^{-1}(k^{(i)})) = \mu \circ \tilde{\kappa}_i(x)$ since μ is linear. Hence $\mu^{-1} \circ \kappa_i = \tilde{\kappa}_i \circ \mu^{-1}$, where $\tilde{\kappa}_i$ is the binary addition of $\mu^{-1}(k^{(i)})$. Use this for $i = 1, \ldots, r$.

This gives

$$F_k^{-1} = \kappa_0 \circ \tau^{-1} \circ \sigma^{-1} \circ \tilde{\kappa}_1 \circ \ldots \circ [\mu^{-1} \circ \tau^{-1} \circ \sigma^{-1} \circ \tilde{\kappa}_i] \circ \ldots \circ \kappa_r.$$

Hence the decryption algorithm is composed in the same way as the encryption algorithm with the following modifications:

- a modified key expansion: $\tilde{\kappa}_i$ instead of κ_i,
- the reverse order of the partial keys,
- `MixColumns`: μ replaced by the inverse linear map μ^{-1},
- `Shiftrows`: τ replaced by a right shift,
- S-box S_{RD} replaced by the inverse map $\mathrm{S}_{\mathrm{RD}}^{-1}$.

Key Expansion

We wont describe the key expansion in detail, but only mention that it involves cyclic shifts of bytes inside blocks of lengths 4, the S-box S_{RD}, and the addition of fixed constants.

6.2 The Arithmetic of the Base Field

For the description of AES we identify the 8-dimensional $\mathbb{F}_2$ vector space $\mathbb{F}_2^8$ and the field $\mathbb{F}_{256}$. We specify the exact mapping in the following subsections.

Algebraic Representation of the Base Field

The simplest construction of a finite field, see Appendix A, is as a factor ring of the polynomial ring $\mathbb{F}_p[X]$ over its prime field $\mathbb{F}_p$ by a principal ideal that is generated by an irreducible polynomial $h \in \mathbb{F}_p[X]$. The ideal $h\mathbb{F}_p[X]$ is prime, hence

$$K := \mathbb{F}_p[X]/h\mathbb{F}_p[X]$$

is a finite field and has degree (= dimension) $n = \deg h$ over $\mathbb{F}_p$. For the identification of K with the vector space $\mathbb{F}_p^n$ we identify the residue classes of the powers of X with the n unit vectors. So setting $x = X \bmod h$ we identify:

$$x^0 = 1 = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \quad x^1 = x = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{pmatrix}, \quad \ldots, \quad x^{n-1} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}.$$

If $h = X^n + a_1X^{n-1} + \cdots + a_{n-1}X + a_n$ (monic without loss of generality), then from $h \bmod h = 0$ we get

$$x^n = -a_1x^{n-1} - \cdots - a_{n-1}x - a_n$$

in K. Moreover this equation shows how to express the residue class of an arbitrary polynomial f by the canonical basis $1, x, \ldots, x^{n-1}$. Algorithmically this amounts to the remainder of a polynomial division "f divided by h".

For AES we use the polynomial

$$h = X^8 + X^4 + X^3 + X + 1 \in \mathbb{F}_2[X].$$

Multiplication Table

The multiplication table for the basis $(1, x, \ldots, x^{n-1})$ follows from the relation defined by h. In $\mathbb{F}_{256}$ (for AES) we have

$$x^2 \cdot x^7 = x^9 = x \cdot x^8 = x \cdot (x^4 + x^3 + x + 1) = x^5 + x^4 + x^2 + x.$$

Efficient inversion

The implementation of AES uses a complete value table of the S-box. This is efficient for we have to specify only 256 values.

Appendix A

Finite Fields

As a corollary of the Euclidean algorithm we saw that the integers modulo a prime number p form a field, $\mathbb{Z}/p\mathbb{Z} = \mathbb{F}_p$. (For a simple direct proof observe that multiplying by a nonzero element is injective.) The fields $\mathbb{F}_p$ play an important role in the theory of finite fields.

The purpose of this appendix is to determine all finite fields.

A.1 Prime Fields

For an arbitrary ring R (with 1) and an integer n the product $n \cdot 1 \in R$ has a natural definition as sum $1 + \cdots + 1$ of n exemplars of 1, if $n > 0$, as 0, if $n = 0$, and as $-|n| \cdot 1$, if $n < 0$. This makes R an algebra over $\mathbb{Z}$ and defines a canonical ring homomorphism

$$\alpha\colon \mathbb{Z} \longrightarrow R, \;\; \alpha(n) = n \cdot 1.$$

The kernel of α is an ideal $m\mathbb{Z}$ with $m \geq 0$. If $m = rs$, then $\alpha(r)\alpha(s) = 0$ in R. Thus if R is an integral domain (say a field), then $m = p$ is a prime number or 0, and is called the **characteristic** of R. If K is a finite field, then $p > 0$ (else α would be injective), and the Homomorphy Theorem yields a natural embedding $\mathbb{F}_p \hookrightarrow K$. Usually one identifies the field $\mathbb{F}_p$ with its image in K and calls it the **prime field** of K.

Remarks

1. If K is a field of characteristic $p > 0$, then $pa = 0$ for all $a \in K$, since $pa = (p \cdot 1) \cdot a = 0 \cdot a$.

2. With the same assumptions $(a + b)^p = a^p + b^p$ for all $a, b \in K$. For by the Binomial Theorem

$$(a + b)^p = a^p + \sum_{i=1}^{p-1} \binom{p}{i} a^{p-i} b^i + b^p.$$

Since p divides all binomial coefficients $\binom{p}{i}$ for $0 < i < p$ the sum is 0. In particular the map $a \mapsto a^p$ is a ring homomorphism of K into itself with kernel 0, hence injective. If K is finite, it is an automorphism.

Now let K be a finite field of characteristic p with $q = \#K$ elements. Then K is a finite dimensional vector space over $\mathbb{F}_p$. If $e = \dim K$, then K as a vector space is isomorphic with $\mathbb{F}_p^e$. Hence $q = p^e$.

We have proved:

Theorem 4 *Let K be a finite field, and q the number of its elements. Then there is a prime number p and an exponent e such that $q = p^e$. Furthermore K has characteristic p and contains the prime field $\mathbb{F}_p$ (up to isomorphism).*

A.2 The Multiplicative Group of a Finite Field

This is a standard result of Algebra:

Proposition 9 *Let K be a field, and $G \leq K^\times$ a finite subgroup with $\#G = n$ elements. Then G is cyclic and consists of the n-th roots of unity in K.*

Proof. For $a \in G$ always $a^n = 1$. Hence G is contained in the set of roots of the polynomial $T^n - 1 \in K[T]$. Hence K has exactly n different n-th roots of unity, and G consists exactly of these. Now let m be the exponent of G, in particular $m \leq n$. The following Lemma 2 yields: All $a \in G$ are m-th roots of unity whose number—as roots of the polynomial $T^m - 1$—is at most m. Therefore also $n \leq m$, hence $n = m$, and G has an element of order n. $\diamond$

Lemma 2 *Sei G be an abelian group.*

(i) *Let $a, b \in G$, $\operatorname{ord} a = m$, $\operatorname{ord} b = n$, where m, n are finite and coprime. Then $\operatorname{ord} ab = mn$.*

(ii) *Let $a, b \in G$, $\operatorname{ord} a$, $\operatorname{ord} b$ finite, $q = \operatorname{lcm}(\operatorname{ord} a, \operatorname{ord} b)$. Then there is a $c \in G$ with $\operatorname{ord} c = q$.*

(iii) *Let $m = \max\{\operatorname{ord} a | a \in G\}$, the exponent of G, be finite. Then $\operatorname{ord} b \mid m$ for all $b \in G$.*

Proof. (i) Let $k := \operatorname{ord}(ab)$. From $(ab)^{mn} = (a^m)^n \cdot (b^n)^m = 1$ it follows that $k|mn$. Since $a^{kn} = a^{kn} \cdot (b^n)^k = (ab)^{kn} = 1$ also $m|kn$, hence $m|k$, and likewise $n|k$, hence $mn|k$.

(ii) Let p^e be a prime power with $p^e|q$, say $p^e|m := \operatorname{ord} a$. Then a^{m/p^e} has order p^e. If $q = p_1^{e_1} \cdots p_r^{e_r}$ is the prime decomposition with different primes p_i, then there are $c_i \in G$ with $\operatorname{ord} c_i = p_i^{e_i}$. By (i) $c = c_1 \cdots c_r$ has order q.

(iii) Let $\operatorname{ord} b = n$. Then there is a $c \in G$ with $\operatorname{ord} c = \operatorname{lcm}(m,n)$. Thus $\operatorname{lcm}(m,n) \leq m$, hence $= m$, hence $n|m$. $\diamond$

Theorem 5 *Let K be a finite field, $\#K = q$. Then the multiplicative group $K^\times$ is cyclic of order $q-1$, and $a^{q-1} = 1$ for all $a \in K^\times$. Moreover $a^q = a$ for all $a \in K$. In particular K consists exactly of the roots of the polynomial $T^q - T \in \mathbb{F}_p[T]$.*

An element $a \in K$, K finite, is called **primitive** if it generates the multiplicative group $K^\times$.

A.3 Irreducible Polynomials and Field Extensions

Given two fields $L \supseteq K$ with $n = \operatorname{Dim}_K L < \infty$ we call L a finite field extension of K, and n its degree.

There is a common way of constructing field extensions: Let $f \in K[T]$ be an irreducible polynomial of degree n.

> The definition of "irreducible" is: f is not constant, and if $f = gh$ for $g, h \in K[T]$, then g or h is constant.

We'll show that $L = K[T]/fK[T]$ is a field extension of degree n.

First $K \subseteq K[T]$ as the set of constant polynomials, and $K \cap fK[T] = 0$. Therefore the natural homomorphism $K[T] \to L$ induces an injection $K \hookrightarrow L$, that allows us to identify K as a subfield of L.

Next we want to show that L is a field. We start with the division algorithm of polynomials. For a convenient handling of the zero polynomial in this context we assign it the degree $-\infty$. Thus $\deg r < 0$ is equivalent with $r = 0$.

Proposition 10 *Let K be a field, and let $f, g \in K[T]$, $g \neq 0$. Then there are uniquely determined polynomials $q, r \in K[T]$ such that $f = q \cdot g + r$ and $\deg r < \deg g$.*

Proof. Uniqueness: If $f = \tilde{q} \cdot g + \tilde{r}$ with $\deg \tilde{r} < \deg g$, then

$$0 = (\tilde{q} - q) \cdot g + \tilde{r} - r,$$

$$(q - \tilde{q}) \cdot g = \tilde{r} - r.$$

The degree of the right-hand side is $< \deg g$. If we assume that $q \neq \tilde{q}$, then the left-hand side has degree $\geq \deg g$ because the degree of a product is the sum of the degrees, contradiction. Hence $q = \tilde{q}$, and consequently also $r = \tilde{r}$.

Existence: We use the following Lemma 3 to conclude that we get a correct algorithm by the instructions:

Initialization: Put $r := f$, $q := 0$. (Then $f = qg + r$.)

Division loop: While $\deg r \geq \deg g$, replace q by $q + s$ and r by $r - sg$ with $\deg(r - sg) < \deg r$. (Then $\deg r$ decreases while the condition $f = qg + r$ is preserved.)

At the exit of the loop we have the sought-after polynomials. $\diamond$

Lemma 3 *Let $n \geq m$ and $f = a_nT^n + \cdots a_0$, $g = b_mT^m + \cdots + b_0$ with leading coefficients $a_n, b_m \neq 0$. Then* $\deg(f - qg) < \deg f$ *for*

$$q = \frac{a_n}{b_n} \cdot T^{n-m}.$$

Proof. The leading term of f cancels out. $\diamond$

As for integers this algorithm leads to an Euclidean algorithm. Here we only need a theoretical consequence. Define a **principal ring** to be a ring R all of whose ideals are principal, that is of the form aR (we consider commutative rings only). We already know a principal ring: $\mathbb{Z}$.

Proposition 11 *The polynomial ring $K[T]$ over a field K is principal.*

Proof. Let $\mathfrak{a} \trianglelefteq K[T]$ be an ideal. We may assume $\mathfrak{a} \neq 0$. Choose $g \in \mathfrak{a}$ of minimal degree ≥ 0, and $f \in \mathfrak{a}$ arbitrary. Division yields $r = f - qg \in \mathfrak{a}$ with a smaller gegree. This is possible only if $r = 0$, hence $f = qg \in gK[T]$. Therefore $\mathfrak{a} = gK[T]$. $\diamond$

An ideal $\mathfrak{m} \trianglelefteq R$ of a ring R is called maximal if it is maximal in the ordered set of *proper* ideals $\mathfrak{a} \neq R$. An ideal $\mathfrak{m}$ is maximal if and only if the residue class ring $R/\mathfrak{m}$ has only two ideals: the zero ideal $\mathfrak{m}/\mathfrak{m}$, and the unit ideal $R/\mathfrak{m}$, that is if and only if it is a field.

Proposition 12 *Let $f \in K[T]$ be irreducible and have degree n. Then $L = K[T]/fK[T]$ is a field extension of K of degree n.*

Proof. First L is a field since $fK[T]$ is a maximal ideal: If $fK[T] \subseteq \mathfrak{a} \triangleleft K[T]$, then the ideal $\mathfrak{a}$ also is principal $= gK[T]$. As a member of this ideal $f = gh$, and the irreducibility forces $h \in K$. Hence $fK[T] = gK[T] = \mathfrak{a}$.

Furthermore L as a vector space is spanned by the residue classes $t_i = T^i \bmod f$. The equation $f \bmod f = 0$ displays t^n as a linear combination of $t_0, \ldots, t_{n-1}$. By induction all t_i $(i \geq n)$ are linear combinations. Hence the dimension is $\leq n$. A linear combination $= 0$ of $t_0, \ldots, t_{n-1}$ would define a polynomial $g \equiv 0 \pmod{f}$ of degree $\leq n - 1$. Hence all its coefficients must be 0. Thus the dimension is $= n$. $\diamond$

An isomorphism of field extensions of K is an isomorphism of fields that fixes all elements of K. By $K[a]$ for $a \in L \supseteq K$ we denote the smallest subring of L that contains K and a. It consists of the polynomial expressions in a with coefficients in K. Note that in general these are not all different as elements of L.

Corollary 3 *Let $f \in K[T]$ be irreducible. Then in the field $L = K[T]/fK[T]$ the polynomial f has the root $t = T \bmod f$.*

If $M \supseteq K$ is a field extension containing a root a of f, then $K[a] \cong L$.

Proof. The natural homomorphism $K[T] \to L$ coincides with the substitution map $g \mapsto g(t)$. It maps f to 0, and that means that $f(t) = 0$.

The substitution map $K[T] \to M$, $g \mapsto g(a)$, is a homomorphism whose kernel contains $fK[T]$. By the Homomorphy Theorem it induces a homomorphism $\varphi\colon L \to M$. Since L is a field φ is injective, and the image of φ is $K[a]$. $\diamond$

This construction of field extensions generalizes one of the usual constructions of the complex numbers as $\mathbb{C} = \mathbb{R}[T]/(T^2 + 1)\mathbb{R}[T]$.

A.4 Splitting Fields

Continuing the considerations of the last section we are going to construct a field extension where a given polynomial f, not necessarily irreducible, splits into linear factors.

If f is reducible (i. e. not irreducible), then we split off a factor of smaller degree and successively arrive at a decomposition into irreducible polynomials. (Showing the uniqueness is easy but not needed here.) Therefore there is a field extension $L \supseteq K$ such that f has a root in L, hence a linear factor in $L[T] \supseteq K[T]$. Split this factor off and process the remaining polynomial in the same way until there remain only linear factors. A field extension $L \supseteq K$ where $f \in K[T]$ decomposes into linear factors is called **splitting field** of f. We just have shown the existence:

Proposition 13 *Every polynomial $f \in K[T]$ has a splitting field.*

Now let $L \supseteq K$ be an arbitrary field extension, and $a \in L$. Then

$$\mathfrak{a} = \{g \in K[T] \mid g(a) = 0\}$$

is an ideal of $K[T]$, hence a principal ideal $fK[T]$, where f has minimal degree in $\mathfrak{a} - \{0\}$ and is irreducible. (Otherwise a would be a root of a proper factor of f that also would belong to $\mathfrak{a}$.) Assume without restriction that the leading coefficient of f is 1. Then f is called **minimal polynomial** of a. Clearly its degree is $\dim_K K[a]$.

This said we return to finite fields. Let K be one of them with $q = p^e$ elements, p a prime number. Choose a primitive element $a \in K$. Then each element $\neq 0$ of K is a power of a, whence a forteriori a polynomial in a. Hence $K = \mathbb{F}_p[a]$. The minimal polynomial $f \in \mathbb{F}_p[T]$ of a divides $T^q - T$, and $K \cong \mathbb{F}_p[T]/f\mathbb{F}_p[T]$.

Consider an arbitrary field L of q elements. Then $L \supseteq \mathbb{F}_p$, and L is a splitting field of $T^q - T \in \mathbb{F}_p[T]$. In particular f has a root b in L. Hence $\mathbb{F}_p[b] \cong \mathbb{F}_p[T]/f\mathbb{F}_p[T] \cong K$, and because $\mathbb{F}_p[b]$ has q elements it must be the whole of L. Hence L is isomorphic with K: Up to isomorphism there is at most one field with q elements.

To show the existence we start with a splitting field K of $h = T^q - T \in \mathbb{F}_p[T]$. (We know there is one.) The derivative $h' = -1$ is constant $\neq 0$. Hence all roots of h in K are different. In particular there are q of them. They constitute a subfield of L: The sum of two roots a, b is again a root, $(a + b)^p = a^p + b^p = a + b$, likewise the product, and for $a \neq 0$ also $1/a$. We proved:

Theorem 6 (Galois **1830**/E. H. Moore **1893**) *For each prime power q there is up to isomorphism exactly one field with q elements.*

This result allows us to think of *the* field of q elements. We denote it by $\mathbb{F}_q$.

Appendix B

Polynomials and Polynomial Functions

Consider an arbitrary (commutative) field K. The functions from K^n to K form a K-algebra $A := \mathrm{Map}(K^n, K)$. Let $K[T]$ be the polynomial algebra in the n-tuple $T = (T_1, \ldots, T_n)$ of indeterminates. Then

$$\begin{array}{rcll} \alpha\colon K[T] & \longrightarrow & A, & \\ \varphi & \mapsto & \alpha(\varphi) & \text{with } \alpha(\varphi)(x_1, \ldots, x_n) := \varphi(x_1, \ldots, x_n) \end{array}$$

is a K-algebra homomorphism, called the "substitution homomorphism". Its image, $\alpha(K[T]) \subseteq A$, is the algebra of polynomial functions on K^n. We distinguish two fundamentally different cases—K is infinite, or K is finite.

B.1 Polynomial Functions over Infinite Fields

Let K be infinite. Then α is

- injective, i. e., different polynomials define different functions—the proof is the uniqueness proof of interpolation formulas, and is given below,
- not surjective, because $K[T]$ has the same cardinality as K, but A is strictly larger—the proof is elementary set theory.

The proof of injectivity relies on the following lemma:

Lemma 4 *Let K be a field with at least $d + 1$ elements, and let $\varphi \in K[T]$ be a polynomial of degree $\leq d$ with $\varphi(x) = 0$ for all $x \in K^n$. Then $\varphi = 0$.*

Proof. We prove this by induction on the dimension n. In the case $n = 1$ the polynomial φ has more than d roots, whence $\varphi = 0$ by elementary algebra.

Now let $n \geq 2$. Split the indeterminates into $X = (T_1, \ldots, T_{n-1})$ and $Y = T_n$. Then

$$\varphi = \sum_{i=0}^{d} \psi_i(X) \cdot Y^i \quad \text{where } \deg \psi_i \leq d - i \leq d.$$

Fix an arbitrary $x \in K^{n-1}$. Then $\varphi(x, y) = \sum_i \psi_i(x) \cdot y^i = 0$ for all $y \in K$. The assertion in the case $n = 1$ gives $\psi_0(x) = \ldots = \psi_d(x) = 0$. Since this holds for all x, induction gives $\psi_0 = \ldots = \psi_d = 0$. Hence $\varphi = 0$. $\diamond$

From this lemma we immediately get the following theorem:

Theorem 7 *Let K be an infinite field. Then the substitution homomorphism $\alpha \colon K[T] \longrightarrow A$ is injective.*

Now let $x_1, \ldots, x_d \in K^n$ be d distinct points, $x_i = (x_{i1}, \ldots, x_{in})$. We want to construct a polynomial that takes given (not necessarily distinct) values $a_1, \ldots, a_d$ at these points. To this end consider the polynomials

$$\psi_k := \prod_{i \in \{1,\ldots,d\} \setminus \{k\}} \; \prod_{j \in \{1,\ldots,n \mid x_{ij} \neq x_{kj}\}} (T_j - x_{ij}).$$

For $i \neq k$ at least one coordinate $x_{ij} \neq x_{kj}$, therefore $\psi_k(x_i) = 0$. On the other hand $\psi_k(x_k) \neq 0$. Hence for $\varphi_k := \psi_k / \psi_k(x_k)$ we conclude:

Lemma 5 *For each $k = 1, \ldots d$ there is a polynomial $\varphi_k \in K[T]$ with all partial degrees $\leq d - 1$ and*

$$\varphi_k(x_i) = \begin{cases} 1 & \textit{for } i = k, \\ 0 & \textit{for } i \textit{ otherwise.} \end{cases}$$

Taking the linear combination $\varphi = \sum a_k \varphi_k$ we get:

Theorem 8 *Let $x_1, \ldots, x_d \in K^n$ be d distinct points, and $a_1, \ldots, a_d \in K$. Then there is a polynomial $\varphi \in K[T_1, \ldots, T_n]$ of partial degree $\leq d - 1$ in each T_i such that $\varphi(x_k) = a_k$ for $k = 1, \ldots d$.*

Note that the proof was constructive but didn't care about the most efficient algorithm.

B.2 Polynomial Functions over Finite Fields

Let K be finite with $\#K = q$ elements. Then α is

- not injective, because $K[T]$ is infinite, but $\#A = q^{q^n}$.

- surjective, because $F \in A$ is completely determined by the q^n pairs $(x, F(x))$, $x \in K^n$, that is by the graph of F; interpolation gives a polynomial $\varphi \in K[T]$ with $\varphi(x) = F(x)$ for all $x \in K^n$, i. e., $\alpha(\varphi) = F$. A proof follows directly from Theorem 8, however in the following we give an independent proof.

The polynomial

$$\varphi = \prod_{i=1}^{n} \left(-T_i^{q-1} + 1 \right) \in K[T]$$

has partial degree $q - 1$ in each T_i.

Lemma 6 *The function $\alpha(\varphi)$ is the indicator function*

$$\varphi(x) = \begin{cases} 1 & \text{for } x = 0, \\ 0 & \text{for } x \in K^n \text{ otherwise.} \end{cases}$$

Proof. This is immediate from $a^{q-1} = 1$ for $a \in K^\times$. $\diamond$

Corollary 1 *For each $a \in K$ there is a polynomial $\varphi_a \in K[T]$ with all partial degrees $q - 1$ and*

$$\varphi_a(x) = \begin{cases} 1 & \text{for } x = a, \\ 0 & \text{for } x \in K^n \text{ otherwise.} \end{cases}$$

Proof. Take $\varphi_a = \varphi(T_1 - a_1, \ldots, T_n - a_n)$. $\diamond$

Now let $F\colon K^n \longrightarrow K$ be given. Then the polynomial

$$\varphi = \sum_{a \in K^n} F(a)\varphi_a \in K[T]$$

has all partial degrees $\leq q - 1$, and $\varphi(x) = F(x)$ for all $x \in K^n$. This proves the following theorem:

Theorem 9 *Let K be a finite field with q elements, and $n \in \mathbb{N}$. Then each function $F\colon K^n \longrightarrow K$ is given by a polynomial $\varphi \in K[T_1, \ldots, T_n]$ of partial degree $\leq q - 1$ in each T_i.*

Corollary 2 *Each function $F : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ is given by a polynomial $\varphi \in \mathbb{F}_2[T_1, \ldots, T_n]$ that is linear in each T_i.*

Corollary 3 *The kernel of the substitution homomorphism α is the ideal $\mathfrak{a} = (T_1^q - T_1, \ldots, T_n^q - T_n) \trianglelefteq K[T]$.*

Proof. Clearly $\mathfrak{a} \subseteq \ker \alpha$. Because $\dim K[T]/\mathfrak{a} = q^n = \dim A$, and α is surjective, we have $\mathfrak{a} = \ker \alpha$. $\diamond$

Corollary 4 *Let $m, n \in \mathbb{N}$. Then each map $F : K^n \longrightarrow K^m$ is given by an m-tuple $(\varphi_1, \ldots, \varphi_m)$ of polynomials $\varphi_i \in K[T_1, \ldots, T_n]$ of partial degree $\leq q - 1$ in each T_i.*

Corollary 5 *Each map $F : V \longrightarrow W$ between finite dimensional K-vectorspaces V and W is polynomial with partial degrees each $\leq q - 1$.*

Appendix C

Boolean Functions, Boolean Maps, and Boolean Circuits

This Chapter is in the separate document `Boole.pdf`

Appendix D

Fourier Analysis of Boolean Maps

This Chapter is in the separate document `Fourier.pdf`

Appendix E

The Hypergeometric Distribution

The urn problem underlying the hypergeometric distribution is "drawing without replacement". Assume the urn contains n balls s of which are black, and $t = n - s$ are white. Let

$$p := \frac{s}{n}$$

be the proportion of black balls, and assume without loss of generality that $p > \frac{1}{2}$. (The case $p = \frac{1}{2}$ is not interesting, the case $p < \frac{1}{2}$ is symmetric to the considered case.)

Draw r balls ($r \leq n$) by random. The probability that exactly ν of the balls are white is

$$q_r^{(s)}(\nu) = \frac{\binom{s}{r-\nu}\binom{t}{\nu}}{\binom{n}{r}}.$$

The function

$$q_r^{(s)} \colon \mathbb{Z} \longrightarrow \mathbb{R}$$

is called the **hypergeometric distribution** (with parameters n, s, and r). We have $q_r^{(s)}(\nu) = 0$ for $\nu < 0$ as well as for $\nu > r$. The probability of drawing more blacks balls than white ones is

$$p_r^{(s)} = \begin{cases} \sum_{\nu=0}^{\frac{r-1}{2}} q_r^{(s)}(\nu) & \text{if } r \text{ is odd,} \\ \sum_{\nu=0}^{\frac{r}{2}-1} q_r^{(s)}(\nu) + \frac{1}{2} q_r^{(s)}(\frac{r}{2}) & \text{if } r \text{ is even,} \end{cases}$$

in case of a tie we randomly decide between black and white with probability $\frac{1}{2}$.

In the uninteresting case $p = \frac{1}{2}$ obviously all $p_r^{(s)} = \frac{1}{2}$.

Lemma 7
(i) $p_1^{(s)} = p$.

(ii) $p_2^{(s)} = p_1^{(s)}$ (if $t \geq 1$).
(iii) $p_3^{(s)} = \frac{s(s-1)}{n(n-1)} \cdot \left[3 - 2 \cdot \frac{s-2}{n-2}\right]$ (if $t \geq 2$).
(iv) $p_4^{(s)} = p_3^{(s)}$ (if $t \geq 2$).
(v) $p_r^{(s)} = 1$ for $r > 2t$.

Proof. (i) Trivial.

(ii) We draw two balls, and break the tie (in the case where we draw one ball of each type) by a random decision. Therefore the numerator is

$$\binom{s}{2} + \frac{1}{2}\binom{s}{1}\binom{t}{1} = \frac{s(s-1)}{2} + \frac{s(n-s)}{2} = \frac{s(n-1)}{2}.$$

The denominator is $\frac{n(n-1)}{2}$, and the quotient is

$$p_2^{(s)} = \frac{s(n-1)}{n(n-1)} = p.$$

(iii) Here the numerator is

$$\begin{aligned}\binom{s}{3} + \binom{s}{2} \cdot (n-s) &= \frac{s(s-1)(s-2) + 3s(s-1)(n-s)}{6} \\ &= \frac{s(s-1)}{6} \cdot [s - 2 + 3 \cdot (n-s)] \\ &= \frac{s(s-1)}{6} \cdot [3 \cdot (n-2) - 2 \cdot (s-2)].\end{aligned}$$

The denominator is $\frac{1}{6} \cdot n(n-1)(n-2)$, hence the asserted value of $p_3^{(s)}$.

(iv) We omit the calculation since the next lemma contains a more general statement.

(v) In this case we necessarily draw a majority of black balls. ◇

Lemma 8 *If r is even and $2 \leq r \leq 2t$, then*

$$p_{r+1}^{(s)} > p_r^{(s)} = p_{r-1}^{(s)}.$$

Proof. Let $A_r^{(s)}(\nu) = \binom{n}{r} \cdot q_r^{(s)}(\nu)$ be the numerator of $q_r^{(s)}(\nu)$, and $B_r^{(s)} = \binom{n}{r} \cdot p_r^{(s)}$, the numerator of $p_r^{(s)}$.

After $r+1$ drawings we have a black majority in $B_{r+1}^{(s)}$ cases. Considering the change from r to $r+1$ we have:

- $\sum_{\nu=0}^{\frac{r}{2}-1} A_r^{(s)}(\nu)$ cases where the number of black balls is at least $\frac{r}{2}+1$ after r drawings. We have $n-r$ possibilities for the $(r+1)$-th ball, but all of these cannot change the majority. So we get

$$X_1 = (n-r) \cdot \sum_{\nu=0}^{\frac{r}{2}-1} A_r^{(s)}(\nu)$$

cases with a black majority.

- $A_r^{(s)}(\frac{r}{2})$ cases where after r drawings we have exactly $\frac{r}{2}$ black balls. From the $n-r$ possibilities for the $(r+1)$-th ball
  - $s-\frac{r}{2}$ are black and give a black majority,
  - $t-\frac{r}{2}$ are white and give a white majority.

  Thus we get another
  $$X_2 = (s-\frac{r}{2}) \cdot A_r^{(s)}(\frac{r}{2})$$
  cases with a black majority.
- In the remaining cases after r drawings we have at most $\frac{r}{2}-1$ black balls. Therefore the $(r+1)$-th ball cannot change the white majority.

This count contains each resulting set exactly $r+1$ times. Therefore
$$B_{r+1}^{(s)} = \frac{1}{r+1} \cdot (X_1+X_2) = \frac{n-r}{r+1} \cdot \left[\sum_{\nu=0}^{\frac{r}{2}-1} A_r^{(s)}(\nu) + \frac{s-\frac{r}{2}}{n-r} \cdot A_r^{(s)}(\frac{r}{2})\right].$$
For the coefficient of the last term we have
$$\frac{s-\frac{r}{2}}{n-r} > \frac{1}{2} \Longleftrightarrow 2s-r > n-r \Longleftrightarrow s > \frac{n}{2}.$$
(Since $r \leq 2t$ also $r < n$.) Therefore
$$B_{r+1}^{(s)} > \frac{n-r}{r+1} \cdot B_r^{(s)},$$
and the first part of the assertion, $p_{r+1}^{(s)} > p_r^{(s)}$, follows.

Analyzing the change from $r-1$ to r is somewhat more complicated. After r drawings we have a black majority in $B_r^{(s)}$ cases. Among these are:

- $\sum_{\nu=0}^{\frac{r}{2}-2} A_{r-1}^{(s)}$ cases where after $r-1$ drawings we have at least $\frac{r}{2}+1$ black balls. The $n-r+1$ possibilities for the r-th ball can't change the decision. Hence we get
  $$Y_1 = (n-r+1) \cdot \sum_{\nu=0}^{\frac{r}{2}-2} A_{r-1}^{(s)}$$
  cases with black majority.
- $A_{r-1}^{(s)}(\frac{r}{2}-1)$ cases where after $r-1$ drawings we have exactly $\frac{r}{2}$ black balls. The $n-r+1$ possibilities for the r-th ball dissociate into

  - $s - \frac{r}{2}$ black ones that result in a black majority. This makes

    $$Y_2 = (s - \frac{r}{2}) \cdot A^{(s)}_{r-1}(\frac{r}{2} - 1)$$

    additional cases.
  - $t + 1 - \frac{r}{2}$ white ones where we randomly decide with probability $\frac{1}{2}$. This adds another

    $$Y_3 = \frac{1}{2} \cdot (t + 1 - \frac{r}{2}) \cdot A^{(s)}_{r-1}(\frac{r}{2} - 1)$$

    cases to our collection.
- $A^{(s)}_{r-1}(\frac{r}{2})$ cases where after $r-1$ drawings we have exactly $\frac{r}{2} - 1$ black balls. The $n - r + 1$ possibilities for the r-th ball dissociate into
  - $s + 1 - \frac{r}{2}$ black ones where we randomly decide with probability $\frac{1}{2}$. This gives another

    $$Y_4 = \frac{1}{2} \cdot (s + 1 - \frac{r}{2}) \cdot A^{(s)}_{r-1}(\frac{r}{2})$$

    cases.
  - $t - \frac{r}{2}$ white ones that don't disturb the white majority.
- In the remaining cases after $r-1$ drawings we have at most $\frac{r}{2} - 2$ black balls. The white majority is unchanged.

Each set of drawn balls is counted exactly r times. Therefore

$$\begin{aligned}
B^{(s)}_r &= \frac{1}{r} \cdot (Y_1 + Y_2 + Y_3 + Y_4) \\
&= \frac{n - r + 1}{r} \cdot \sum_{\nu=0}^{\frac{r}{2}-2} A^{(s)}_{r-1} + \frac{1}{r} \cdot (s - \frac{r}{2} + \frac{t}{2} + \frac{1}{2} - \frac{r}{4}) \cdot A^{(s)}_{r-1}(\frac{r}{2} - 1) \\
&\quad + \frac{1}{2r} \cdot (s - \frac{r}{2} + 1) \cdot A^{(s)}_{r-1}(\frac{r}{2})
\end{aligned}$$

Since $s + \frac{t}{2} = n - \frac{t}{2}$ the coefficient of the middle term equals

$$s - \frac{r}{2} + \frac{t}{2} - \frac{r}{4} + \frac{1}{2} = n - \frac{t}{2} - r + \frac{r}{4} + 1 - \frac{1}{2} = (n - r + 1) - \frac{1}{2} \cdot (t - \frac{r}{2} + 1).$$

Hence

$$\begin{aligned}
B^{(s)}_r &= \frac{n - r + 1}{r} \cdot \sum_{\nu=0}^{\frac{r}{2}-1} A^{(s)}_{r-1} \\
&\quad - \frac{1}{2r}(t - \frac{r}{2} + 1)\binom{s}{\frac{r}{2}}\binom{t}{\frac{r}{2} - 1} + \frac{1}{2r}(s - \frac{r}{2} + 1)\binom{s}{\frac{r}{2} - 1}\binom{t}{\frac{r}{2}}.
\end{aligned}$$

The two last terms cancel. What remains is

$$B_r^{(s)} = \frac{n-r+1}{r} \cdot B_{r-1}^{(s)}.$$

This proves the second part of the assertion. $\diamond$

We conclude:

Proposition 14 *The probability $p_r^{(s)}$ grows monotonically with r from $p_1^{(s)} = p$ to $p_{2t+1}^{(s)} = 1$.*

If the quotients

$$\frac{rs}{n}, \frac{rt}{n}, \frac{(n-r)s}{n}, \frac{(n-r)t}{n}$$

are sufficiently large (by FISHER's rule of thumb: ≥ 5), the normal distribution approximates the hypergeometric distribution well. In particular

$$\sum_{\nu=0}^{x} q_r^{(s)}(\nu) \approx \Phi(\frac{x-\mu}{\sigma}) = \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{\frac{x-\mu}{\sigma}} e^{-t^2/2}\,dt \qquad (1)$$

where μ is the mean value and σ^2 is the variance of the hypergeometric distribution (with parameters n, s, and r), and Φ is the distribution function of the normal distribution. For mean value and variance we have:

Lemma 9

$$\begin{aligned} \mu &= \frac{rt}{n}, \\ \sigma^2 &= \frac{r(n-r)\cdot t(n-t)}{n^2(n-1)}. \end{aligned}$$

Proof. Take a random sample of r balls. Let $X_k : \Omega \longrightarrow \mathbb{R}$ be a random variable that assumes the value 0 if the k-th ball is black, and 1 if it is white. Then $S = X_1 + \cdots + X_r : \Omega \longrightarrow \mathbb{R}$ is a random variable that counts the number of white balls in our sample. Then $\mu = \mathrm{E}(S)$ is the expectation and $\sigma^2 = \mathrm{Var}(S)$ is the variance of this random variable.

Since $\mathrm{E}(X_k) = \frac{t}{n}$ we have $\mathrm{E}(S) = r \cdot \frac{t}{n}$.

We note that $X_k^2 = X_k$ and derive

$$\mathrm{Var}(X_k) = \mathrm{E}(X_k^2) - \mathrm{E}(X_k)^2 = \frac{t}{n} - \frac{t^2}{n^2} = \frac{t(n-t)}{n^2}.$$

Since $X_jX_k(\omega) = 1 \iff X_j(\omega) = 1$ *and* $X_k(\omega) = 1$ the probability of this event is $\frac{t(t-1)}{n(n-1)}$. This gives the expectation $\mathrm{E}(X_jX_k) = \frac{t(t-1)}{n(n-1)}$. Thus the covariance is

$$\begin{aligned} \mathrm{Cov}(X_j, X_k) &= \mathrm{E}(X_jX_k) - \mathrm{E}(X_j)\mathrm{E}(X_k) = \frac{t(t-1)}{n(n-1)} - \frac{t^2}{n^2} \\ &= \frac{t(n(t-1) - t(n-1))}{n^2(n-1)} = \frac{t(t-n)}{n^2(n-1)}. \end{aligned}$$

We deduce the variance of S:

$$\begin{aligned}
\mathrm{Var}(S) &= \sum_{k=1}^{r} \mathrm{Var}(X_k) + 2 \cdot \sum_{1 \le j < k \le r} \mathrm{Cov}(X_j, X_k) \\
&= \frac{rt(n-t)}{n^2} + r(r-1) \cdot \frac{t(t-n)}{n^2(n-1)} = \frac{rt(n-t)}{n^2} \cdot \left[1 - \frac{r-1}{n-1}\right] \\
&= \frac{rt(n-t)}{n^2(n-1)} \cdot [n-r],
\end{aligned}$$

as claimed. $\diamond$

Proposition 15 (Asymptotic distribution) *The probability of a majority of black balls is*

$$p_r^{(s)} \approx \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{\sqrt{r\lambda}} e^{-t^2/2} dt$$

with $\lambda = (2p-1)^2$, under the assumption that $p \approx \frac{1}{2}$, $r \ll n$, and r not too small.

[By FISHER's rule of thumb $10 \le r \le n - 10$ suffices if $p \approx \frac{1}{2}$.
Note that this “proposition” lacks mathematical precision.]
Proof. We look at the upper boundary of the integral (1) for $x = \frac{r}{2}$:

$$\begin{aligned}
\frac{x-\mu}{\sigma} &= \frac{(\frac{r}{2} - \frac{rt}{n}) \cdot n \cdot \sqrt{n-1}}{\sqrt{r(n-r)t(n-t)}} = \frac{(rn - 2rt)\sqrt{n-1}}{2 \cdot \sqrt{r(n-r)t(n-t)}} \\
&= \frac{\sqrt{r}\sqrt{n-1}}{\sqrt{n-r}} \cdot \frac{s-t}{2\sqrt{st}} = \frac{\sqrt{n-1}}{\sqrt{n-r}} \cdot \sqrt{r} \cdot \frac{2p-1}{2\sqrt{p(1-p)}} \\
&\approx 1 \cdot \sqrt{r} \cdot \frac{2p-1}{2 \cdot \sqrt{\frac{1}{4}}} = \sqrt{r\lambda},
\end{aligned}$$

as claimed. $\diamond$

Bibliography

[1] Joan Daemen, Vincent Rijmen, *The Design of Rijndael. AES – The Advanced Encryption Standard.* Springer, Berlin 2002.

[2] Michael R. Garey, David S. Johnson, *Computers and Intractability.* Freeman, New York 1979.