The MAGENTA Block Cipher Algorithm

M.J. Jacobson, Jr.[*] and K. Huber[†]
Deutsche Telekom AG
Am Kavalleriesand 3
64295 Darmstadt
GERMANY

June 8, 1998

Contents

[*]jacobs@cdc.informatik.tu-darmstadt.de
[†]huber@tzd.telekom.de

1 Introduction

The development of MAGENTA (*M*ultifunctional *A*lgorithm for *G*eneral-purpose *E*ncryption and *N*etwork *T*elecommunication) began in 1990, with the basic design principles explained in the unpublished paper [7]. The idea was to apply simple and transparent techniques (no magic tables or constants) which can be efficiently implemented in both software and hardware. Originally, this was realized by using a butterfly structure to accomplish diffusion and discrete exponentiation in a finite field for confusion.

In the following years, the idea of hardware realizations was investigated more closely. In cooperation with hardware specialist S. Wolter the butterfly structure of the original proposal was switched to the FHT shuffle structure, which has the advantage of giving identical structures at each stage. Further slight changes to the algorithm occurred when an analysis of the algorithm done by specialists of the company SIT [5] was carried out in 1994.

Plans were to develop a chip which would be capable of operating up to the gigabit/sec range (see [8]). It was envisioned to use such chips for encryption of ATM connections. Unfortunately the hardware realization did not proceed as planned since the need for such encryption is not yet widely appreciated, although investigations have shown that it should be possible to achieve [8].

Currently, the MAGENTA algorithm is used within Deutsche Telekom for securing sensitive management data. In addition, a VHDL design and a FPGA (Field Programmable Gate Array) realization are in progress.

There are two schemes to mention which essentially used structures of the fast Fourier Transform for cryptographic purposes prior to the MAGENTA algorithm. The first is an invention of Jean Pierre Vaseur [19] which was filed on 2nd June 1959. The second is the so-called Comp-128 algorithm which is used by some GSM providers. The Comp-128 algorithm was designed at the end of the eighties, and was recently disclosed through the Internet.

In the next section, we describe the MAGENTA algorithm in detail. As mentioned above, the security of MAGENTA has been analyzed in great detail on behalf of Deutsche Telekom by SIT GmbH for use with 128-bit keys. In Sections 4 to 9, we highlight the details of the internal report based on this analysis [5] and extend the analysis to 192 and 256 bit keys where appropriate.

2 The MAGENTA Algorithm

Let $B = \{0,1\}^8$ be the set of all 8-bit binary vectors (bytes). For $x \in B$, we will write $x = (x_7, x_6, \ldots, x_0)$, and we associate each byte with an integer in $\{0, 1, \ldots, 255\}$ via the formula $(x_7, x_6, \ldots, x_0) \mapsto 2^7 x_7 + 2^6 x_6 + \ldots + x_0$. The operator $\oplus$ will denote bitwise addition modulo 2, i.e., the usual bitwise XOR operation.

The heart of the MAGENTA algorithm is based on the Fast Hadamard Transform (FHT) [10]. However, we replace the addition and subtraction at each node in the shuffle structure by the following non-linear operation. Let α be a primitive element of the field $GF(256)$ with generating polynomial $p(x) = X^8 + X^6 + X^5 + X^2 + 1$ and $p(\alpha) = 0$. For all $x \in B$, define

$$f(x) = \begin{cases} \alpha^x & x \neq 255 \\ 0 & x = 255 \end{cases} . \qquad (1)$$

Then, for all $(x, y) \in B^2$ we define

$$A(x,y) = f\left(x \oplus f(y)\right) \qquad (2)$$

and

$$PE(x,y) = (A(x,y), A(y,x)) = (f(x \oplus f(y)), f(y \oplus f(x))) \quad . \qquad (3)$$

For all $(x_0, \ldots, x_{15}) \in B^{16}$, our modification of the FHT is given by

$$T(x_0, \ldots, x_{15}) = \Pi(\Pi(\Pi(\Pi(x_0, \ldots, x_{15})))), \qquad (4)$$

where $\Pi(x_0, \ldots, x_{15})$ is defined as

$$\Pi(x_0, \ldots, x_{15}) = (PE(x_0, x_8), PE(x_1, x_9), \ldots, PE(x_7, x_{15})) \quad . \qquad (5)$$

The function $T(x_0, \dots, x_{15})$ operates on a single 128-bit parameter and returns a 128-bit output. Clearly, this operation is very quick, since it can be implemented entirely with bit operations.

For all $X = (x_0, \dots, x_{15}) \in B^{16}$, define

$$X_e = (x_0, x_2, \dots, x_{14})$$

and

$$X_o = (x_1, x_3, \dots, x_{15}),$$

i.e., X_e consists of the bytes of X with even index and X_o consists of the bytes of X with odd index. The function C consists of repeated applications of our FHT variant, and is recursively defined for $j \geq 1$ and all $(x_0, \dots, x_{15})$ by

$$C^{(j+1)}(x_0, \dots, x_{15}) = T\left((x_0, \dots, x_7) \oplus C_e^{(j)}, (x_8, \dots, x_{15}) \oplus C_o^{(j)}\right) \tag{6}$$

where the initial value $C^{(1)} = T(x_0, \dots, x_{15})$. For a fixed number of rounds r, we define

$$E^{(r)}(x_0, \dots, x_{15}) = C_e^{(r)} \ . \tag{7}$$

Originally, MAGENTA was designed with $r = 7$. However, during analysis by SIT GmbH [5, Appendix] it was discovered that using $r = 7$ made a chosen plaintext attack possible. It was recommended that the number of rounds be reduced to 3, and the analysis in the following chapters shows that to the best of our knowledge this choice does not result in any significant cryptographic weaknesses of the overall block cipher. Therefore, we fix $r = 3$.

The complete MAGENTA block cipher makes use of the well-known Feistel construction [3] using the function $E^{(3)}$ as the basic cyryptomodule. For $x = (x_0, \dots, x_{15}) \in B^{16}$ and $y = (y_0, \dots, y_7) \in B^8$, one "Feistel-round" is defined as

$$F_y(x) = \Big((x_8, \dots, x_{15}), (x_0, \dots, x_7) \oplus E^{(3)}(x_8, \dots, x_{15}, y_0, \dots, y_7)\Big) \ . \tag{8}$$

Let $M = (x_0, \dots, x_{15}) \in B^{16}$ be one plaintext block (128 bits). The MAGENTA algorithm supports the following three key sizes:

128 bit:	$K = (K_1, K_2)$,
192 bit:	$K = (K_1, K_2, K_3)$,
256 bit:	$K = (K_1, K_2, K_3, K_4)$,

where $K_1 = (y_0, \dots, y_7)$, $K_2 = (y_8, \dots, y_{15})$, $K_3 = (y_{16}, \dots, y_{23})$, and $K_4 = (y_{24}, \dots, y_{31})$. The MAGENTA algorithm makes use of six or eight Feistel rounds, where each round uses a different part of the key. The algorithm is given by

$$Enc_K(M) = \begin{cases} F_{K_1}(F_{K_1}(F_{K_2}(F_{K_2}(F_{K_1}(F_{K_1}(M)))))) & \text{if } K = (K_1K_2) \in B^{16} \\ F_{K_1}(F_{K_2}(F_{K_3}(F_{K_3}(F_{K_2}(F_{K_1}(M)))))) & \text{if } K = (K_1K_2K_3) \in B^{24} \\ F_{K_1}(F_{K_2}(F_{K_3}(F_{K_4}(F_{K_4}(F_{K_3}(F_{K_2}(F_{K_1}(M)))))))) & \text{if } K = (K_1K_2K_3K_4) \in B^{32} \end{cases} . \tag{9}$$

Due to a palindromic property of Equation 9 given in Section 9, the decryption function can easily be expressed in terms of the encryption function by

$$Dec_K(M) = V\left(Enc_K(V(M))\right), \tag{10}$$

where $V(x_0, \dots, x_{15}) = (x_8, x_9, \dots, x_{15}, x_0, x_1, \dots, x_7)$.

3 Computational Efficiency

In this section, we give estimates of the number of clock cycles required to perform five basic operations of the MAGENTA algorithm on two architectures, an Intel Pentium Pro processor running at 200 MHz and the Z80 microprocessor, a typical 8-bit processor. In particular, we consider the following operations:

- set up a key,
- change a key,
- initialize the algorithm,
- encrypt one 128-bit block in ECB mode,
- decrypt one 128-bit block in ECB mode.

We analyze all three key sizes supported by MAGENTA, namely 128, 192, and 256 bits.

In order to obtain efficiency estimates on a Pentium Pro processor, we computed the run times in seconds required to perform each of the five operations 10^6 times on a given random key-plaintext pair using our optimized C implementation. These run times were then normalized to obtain an approximation of the number of clock cycles required for one iteration by dividing by 10^6 (the total number of iterations) and multiplying by the number of cycles per second ($2 \cdot 10^8$). The results of these calculations are in Table 1.

Operation	128-bit	192-bit	256-bit
Key Setup	2044	3094	4308
Change Key	2044	3094	4308
Alg. Init	2320	2320	2320
Encrypt Block	23694	23694	31550
Decrypt Block	23884	23884	31744

Table 1: Efficiency estimates (in clock cycles) for 200MHz Pentium Pro

Note that in MAGENTA, key setup and changing a key are the same operation. Furthermore, the initialization of the algorithm consists of computing a table of values of the function f, i.e., a table containing 256 bytes. This initialization need only be performed once per session, or not at all if an array representing f is implemented as a constant.

For computing estimated clock cycle requirements on a Z80 processor, we use the architecture and instruction set specified in [20]. We have written rough assembler programs for parts of MAGENTA using this instruction set, based on an optimized C implementation. These programs have never been tested or even run on a computer, so our clock cycle counts should be considered only as estimates. Samples of this assembler code can be found in Appendix A. The clock cycle estimates are located in Table 2.

One frequently occurring operation in MAGENTA is copying 8-byte and 16-byte blocks of memory. Z80 assembler code for this can be found in Appendix A. According to the cycle requirements per instruction given in [20], this operation (as we have written it) requires 72 cycles for 8-byte blocks and 136 cycles for 16-byte blocks.

If we assume that the key input is given to us as raw byte data, then setting up the key consists of recording the size of the key and copying either 2, 3, or 4 8-byte data blocks, depending on the key size. Assembler code for this is given in Appendix A. According to the cycles per instruction in [20], this requires 157, 234, or 306 cycles for 128, 192, and 256-bit keys respectively. Again, changing a key is the same operation as setting a key up. It should be noted that the clock cycle requirements we have computed for key setup on a Pentium Pro are higher because the C implementation we used includes overhead for converting the ASCII character representation of the key to raw byte format.

Assembler code for initializing the algorithm (i.e., computing a table of values representing the function f) is given in Appendix A. According to the cycles per instruction in [20], this operation takes 3582 clock cycles. However, as stated above, this table should be implemented as a constant in an actual implementation, so this computation should usually not be necessary.

The function Π (Equation 5) requires 672 clock cycles according to the instruction set in [20] and our assembler code in Appendix A. The function T (Equation 4) consists of four consecutive applications of PI, and hence requires 2688 clock cycles. During one Feistel round (Equation 8), T is executed three times. In addition, based on our optimized C implementation, two 8-byte and two 16-bytes memory transfers are required, as well as two iterations of simple computations requiring 144 and 188 cycles each per iteration (computing X_e and X_o and computing the XOR of two 16-byte blocks) and one iteration of a block requiring 108 cycles (computing the last 8-byte block XOR sum). Hence, one complete Feistel round requires 9249 clock cycles. One encryption of a 128-bit block in ECB mode requires six Feistel rounds for 128 and 192-bit keys and eight Feistel rounds for 256-bit keys, plus one 16-byte memory copy (copying the plaintext block). In total, 55630 clock cycles are required to encrypt one block using 128 and 192-bit keys, and 74128 clock cycles for 256-bit keys. The decrypt operation is almost identical, except we also need to compute the function $V(x_0, \ldots, x_{15}) = (x_8, x_9, \ldots, x_{15}, x_0, x_1, \ldots, x_7)$ twice, which costs an additional four 8-byte memory copies, resulting in 55918 clock cycles for 128 and 192-bit keys, and 74416 clock cycles for 256-bit keys.

Operation	128-bit	192-bit	256-bit
Key Setup	157	234	306
Change Key	157	234	306
Alg. Init	3582	3582	3582
Encrypt Block	55630	55630	74128
Decrypt Block	55918	55918	74416

Table 2: Efficiency estimates (in clock cycles) for 8-bit processor

4 Algebraic Properties

4.1 The Function $f(x)$

During one application of the MAGENTA algorithm, the function $f(x)$ (Equation 1) is executed a total of 2304 times for 128 and 192-bit keys and 3072 times for 256-bit keys. Since this function is the non-linear building block of the algorithm, it has special importance in any analysis of the entire algorithm. The following properties of f are known:

1. It is one-to-one, i.e., it is a permutation over the set B.

2. This permutation can be represented as a product of 6 disjoint cycles of lengths 198, 38, 9, 5, 5, and 1. With regards to combinatorial analysis [17], these values are "normal" — there is no significant deviation from the cycle representations of randomly generated permutations. The single fixed point is the number 161.

3. Considering bytes as integers (via the map $(x_7, x_6, \ldots, x_0) \mapsto 2^7 x_7 + 2^6 x_6 + \ldots + x_0$), for all $x \in B$ such that $f(x) \in \{1, 2, \ldots, 127\}$ we have

$$f(x+1) = 2 \cdot f(x) \pmod{255},$$

where by $\cdot$ we denote multiplication over the integers. For all $(x, y) \in B^2$ such that $f(x) \cdot f(y) \in \{1, 2, \ldots, 255\}$, we can generalize this property by

$$f((x+y) \pmod{255}) = f(x) \cdot f(y) \ .$$

These relatively strong properties do not seem to have any negative cryptographic effects on the overall security of the algorithm.

4. If we consider $f(x)$ as a vector function of the form $f(x) = (f_7(x), f_6(x), \ldots, f_0(x))$, then each of the eight Boolean functions $f_0, \ldots, f_7$ is non-linear with degree 7. Also, all possible non-trivial XOR-combinations of these functions are non-linear. In [5], these functions are explicitly given in algebraic normal form (Shegalkin polynomials). One interesting property is that each function has exactly 128 summands. No other special properties of these functions (symmetry properties, etc.) were found.

4.2 The Function $PE(x, y)$

The function PE (Equation 3) has two striking properties:

1. For $(x, y) \in B^2$, define $v(x, y) = (y, x)$. Then, $PE(x, y)$ has the following symmetry property:
$$v(PE(x, y)) = PE(v(x, y)), \tag{11}$$
which can simplify certain analyses.

2. $PE(x, y)$ is not one-to-one. There are

24235	elements in B^2 with no pre-image
23952	elements in B^2 with exactly one pre-image
12028	elements in B^2 with exactly two pre-images
4079	elements in B^2 with exactly three pre-images
1007	elements in B^2 with exactly four pre-images
172	elements in B^2 with exactly five pre-images
47	elements in B^2 with exactly six pre-images
11	elements in B^2 with exactly seven pre-images
3	elements in B^2 with exactly eight pre-images
1	elements in B^2 with exactly nine pre-images
1	elements in B^2 with exactly eleven pre-images.

The element $(236, 236)$ has 9 pre-images, namely $(18, 18)$, $(38, 141)$, $(67, 205)$, $(141, 38)$, $(146, 146)$, $(205, 67)$, $(213, 224)$, $(224, 213)$, $(236, 236)$. The element $(256, 256)$ is a fixed point of PE. The element $(227, 227)$ has 11 pre-images, namely $(17, 111)$, $(28, 255)$, $(55, 55)$, $(104, 186)$, $(111, 17)$, $(126, 126)$, $(166, 166)$, $(173, 173)$, $(186, 104)$, $(197, 197)$, $(255, 28)$.

The fact that PE is not one-to-one implies that it is potentially easier to construct collisions if MAGENTA is used as a hash function. In addition, it also implies that the images of the function $E^{(3)}$ can occur with unequally distributed frequencies. On the other hand, it is possible that this may increase the difficulty of inverting $E^{(3)}$.

4.3 Two Successive Combinations of Π

The function T consists of 4 successive applications of Π. Two successive combinations of Π, i.e., the function $\Pi(\Pi(x_0, \ldots, x_{15}))$ consists of four independent PE-combinations. Each of these combinations has four input bytes (x_0, x_1, x_2, x_3) and transforms them into four output bytes (y_0, y_1, y_2, y_3) as follows:

$$\begin{aligned}
y_0 &= f\left(f(x_0 \oplus f(x_1)) \oplus f(f(x_2 \oplus f(x_3)))\right) \\
y_1 &= f\left(f(x_2 \oplus f(x_3)) \oplus f(f(x_0 \oplus f(x_1)))\right) \\
y_2 &= f\left(f(x_1 \oplus f(x_0)) \oplus f(f(x_3 \oplus f(x_2)))\right) \\
y_3 &= f\left(f(x_3 \oplus f(x_2)) \oplus f(f(x_1 \oplus f(x_0)))\right)
\end{aligned}$$

As a direct consequence of the properties of PE we have:

1. This function has the following symmetry properties:
   (a) The permutation $\pi : (x_0, x_1, x_2, x_3) \to (x_2, x_3, x_0, x_1)$ on the pre-image implies the permutation $\rho : (y_0, y_1, y_2, y_3) \to (y_1, y_0, y_3, y_2)$ on the image.
   (b) The permutation $\pi' : (x_0, x_1, x_2, x_3) \to (x_1, x_0, x_3, x_2)$ on the pre-image implies the permutation $\rho' : (y_0, y_1, y_2, y_3) \to (y_2, y_3, y_0, y_1)$ on the image.
   (c) The permutation $\pi'' : (x_0, x_1, x_2, x_3) \to (x_3, x_2, x_1, x_0)$ on the pre-image implies the permutation $\rho'' : (y_0, y_1, y_2, y_3) \to (y_3, y_2, y_1, y_0)$ on the image.

2. This function is not one-to-one. At least $2 \cdot 24235 \cdot 2^{16} - 24235^2 = 2589194695$ elements of B^4 (more than 60 percent) have no pre-image. There is at least one element in B^4 with $11^2 = 121$ or more pre-images. In particular, the element $(193, 193, 193, 193)$ has 201 pre-images.

The unequal occurrences of elements in B^4 as images leads one to ask how much this is reflected in the function $E^{(3)}$. The frequencies of the image components y_0 and y_2 were determined with respect to all 2^{32} possible pre-images. The variation of the frequencies around the average 65536 was relatively small. The smallest value was 64393 for $(82, 98)$ and the largest was 67130 for $(41, 41)$.

4.4 The Function T

The shuffle structure of the function T (Equation 4) also exhibits some symmetry. In conjunction with the symmetry properties of PE, one can show that for all $(x_0, \ldots, x_{15}) \in B^{16}$

$$\hat{V}(T(x_0, \ldots, x_{15})) = T(\hat{V}(x_0, \ldots, x_{15}))$$

where

$$\hat{V}(x_0, \ldots, x_{15}) = (x_{15}, \ldots, x_0) \ .$$

Generalizations of this easily lead to further properties of T.

From the fact the PE is not one-to-one, it is easy to see that T is not one-to-one either. Since PE is applied 32 times during one application of T, one expects that the image of T is considerably smaller than B^{16}. In fact, it follows from the results of the previous subsection that at least 97.4 percent of all elements in B^{16} have no pre-image with respect to T. On the other hand, the element $(220, 220, \ldots, 220)$ has at least $201^4 > 1.6 \times 10^9$ pre-images.

The fact that T is so clearly not one-to-one can have a negative effect on the confusion and diffusion properties of the overall cipher. However, the following property helps rectify the situation. For all $(x_0, \ldots, x_{15})$, for all $c \in B \backslash \{0\}$, and for all $i \in \{0, \ldots, 15\}$, if

$$(y_0, \ldots, y_{15}) = T(x_0, \ldots, x_{15})$$

and

$$(y'_0, \ldots, y'_{15}) = T(x_0, \ldots, x_{i-1}, x_i \oplus c, x_{i+1}, \ldots, x_{15})$$

then for all $j \in \{0, \ldots, 15\}$ $y_j \neq y'_j$. This property provides us with an essential fact guaranteeing cryptographic properties of the function $E^{(3)}$ as well as the overall cipher. At the same time, it follows that the function T assumes all 256 possible byte-components if all 256^{16} pre-images are applied.

4.5 The Function $E^{(3)}$

The function $E^{(3)}$ (Equation 7) has the following symmetry properties. Let $x \wedge y$ denote bit-wise AND. Then, for all $(x_0, \ldots, x_{15}) \in B^{16}$

1. If $x_0 = x_1 = \ldots = x_{15}$, then for $E^{(3)}(x_0, \ldots, x_{15}) = (y_0, \ldots, y_7)$ it follows that $y_0 = y_1 = \ldots = y_7$. For example, $E^{(3)}(102, \ldots, 102) = (0, \ldots, 0)$.

2. If $x_0 = x_1 = \ldots = x_7$ and $x_8 = x_9 = \ldots = x_{15}$, then for $E^{(3)}(x_0, \ldots, x_{15}) = (y_0, \ldots, y_7)$ it follows that $y_0 = y_4 \wedge y_1 = y_5 \wedge y_2 = y_6 \wedge y_3 = y_7$.

3. If $x_0 = x_1 = x_2 = x_3 = x_8 = x_9 = x_{10} = x_{11}$ and $x_4 = x_5 = x_6 = x_7 = x_{12} = x_{13} = x_{14} = x_{15}$ then for $E^{(3)}(x_0, \ldots, x_{15}) = (y_0, \ldots, y_7)$ it follows that $y_0 = y_2 \wedge y_1 = y_3 \wedge y_4 = y_6 \wedge y_5 = y_7$.

4. If $x_0 = x_1 = x_4 = x_5 = x_8 = x_9 = x_{12} = x_{13}$ and $x_2 = x_3 = x_6 = x_7 = x_{10} = x_{11} = x_{14} = x_{15}$ then for $E^{(3)}(x_0, \ldots, x_{15}) = (y_0, \ldots, y_7)$ it follows that $y_0 = y_1 \wedge y_2 = y_3 \wedge y_4 = y_5 \wedge y_6 = y_7$.

A possible cryptographic effect of these properties is that for certain keys and highly structured plaintexts (with many identical bytes) the corresponding ciphertext may exhibit structural properties. In this respect, one can designate keys with many identical bytes as "weak keys." However, the probability that such a weak key is generated randomly is sufficiently small.

4.6 The MAGENTA Algorithm

An essential requirement for the suitability of (Equation 9) as a cipher is fulfilled due to the fact that for all $K \in B^{16}, B^{24}$, and B^{32}, and all $M \in B^{16}$, the function $Enc_K(M)$ is a one-to-one function $B^{16} \to B^{16}$. This fact follows from the easily demonstrated fact that $F_y(x)$ (Equation 8) is also one-to-one.

Further algebraic properties of $Enc_K(M)$ are given in Section 9.

5 Avalanche Properties

An important design criterium for any block cipher is the "Avalanche Effect" — by modifying one input bit on average one half of the output bits should change. This should ensure that the relationship of the plaintext to the ciphertext and respectively the key to the ciphertext is as irregular as possible, i.e., the ciphertext should seem random and not exhibit any structural dependencies on the plaintext or the key.

The Avalanche Effect is also important in the context of hash functions. The smallest modification of the text should result in considerable modifications to the hash value with high probability.

In the following we describe the results of tests [5], where several of the functions involved in the MAGENTA algorithm were tested as to whether they fulfill one of the strictest avalanche criteria [4]. This criteria is satisfied if the modification of one input bit causes each output bit to change with probability "close" to 1/2. Significance levels were computed for these probabilities and χ^2-tests were applied in order to have a precise test. A short presentation of the tests can be found in [6].

During the course of each test an $n \times n$ matrix $A = (a_{i,j})$, $0 \leq i, j < n$ was generated, where n is the number of input bits and the number of output bits of the function being tested. We call A the dependency matrix. For specific i and j, $a_{i,j}$ is the number of changes of bit i in the image vector when bit j of m randomly selected pre-image vectors is changed. One can consider each $a_{i,j}$ as the sum of m random variables $\mathcal{A}_{i,j}$ with $0 - 1$ distribution. If each of these is binomially distributed with parameter 0.5, then with high probability the value of each $a_{i,j}$ should lie between $(m - 3\sqrt{m})/2$ and $(m + 3\sqrt{m})/2$, the so-called 3σ range.

The following hypothesis was tested:

Hypothesis 1 **(H_0)** *The function under consideration fulfills the strict avalanche criterium (SAC) — for all $i, j \in \{0, \ldots, n-1\}$*

$$P(\mathcal{A}_{i,j} = 1) \approx \frac{1}{2}$$

holds.

A χ^2-test was constructed with help of the dependency matrix with test size

$$\chi^2 = 2 \sum_{0 \leq i < n, 0 \leq j < n} \frac{\left(\frac{m}{2} - a_{i,j}\right)^2}{\frac{m}{2}} \ .$$

A χ^2-distribution with n^2 degrees of freedom was used. For large n (larger than 10) it is approximately normally distributed with $N(n^2, 2n^2)$.

5.1 The Function $f(x)$

Since this function only has 8 input and output bits, the exact probabilities can be computed relatively easily. All the probabilities lie between 0.453 and 0.532, and are clearly within the previously computed interval bounds 0.406 and 0.594. Although there are probably functions for which the corresponding probabilities are closer to 0.5, the results show that there is no reason to modify f. These results also show that every input bit affects every output bit in a non-linear manner [4].

5.2 The Function $PE(x, y)$

The exact probabilities for PE can also be computed exactly, since it only has 16 input and output bits. All the probabilities also lie between 0.453 and 0.532. However, the interval bounds here are 0.494 and 0.506, and the average deviation from 0.5 seems to be too large here. In the following χ^2-test the test bounds were exceeded. As a result, the analysis of the avalanche properties of $E^{(3)}$ was conducted in more depth.

5.3 The Function $E^{(3)}$

Due to the previously described role of $E^{(3)}$ in MAGENTA, it seems practical to fix half of the input bits and analyze the effects of bit modifications in the other half for each of the avalanche tests. Depending on which half of the input bits is fixed, one can refer to "plaintext-ciphertext avalanche" or "key-ciphertext avalanche" [6].

Plaintext-Ciphertext Avalanche

We fix the bytes $(x_8, \ldots, x_{15})$ and analyze the dependence of the 64 image bits on the 64 bits in $(x_0, \ldots, x_7)$. As a spot check, the number of times that each output bit differed by 1 was counted for each of 25000 input pairs that differ in exactly one bit. In this manner, estimations of the investigated probabilities (up to normalization) were obtained. The $64 \cdot 64$ values were then subjected to a χ^2 test.

As fixed values of $(x_8, \ldots, x_{15})$ the following 12 "keys" were used:

- $(0, 0, 0, 0, 0, 0, 0, 0)$
- $(18, 18, 18, 18, 18, 18, 18, 18)$
- $(255, 255, 255, 255, 255, 255, 255, 255)$
- $(171, 171, 171, 171, 205, 205, 205, 205)$
- $(121, 121, 228, 228, 121, 121, 228, 228)$
- $(1, 35, 1, 35, 1, 35, 1, 35)$
- $(212, 236, 144, 246, 118, 168, 92, 66)$
- $(201, 40, 143, 18, 198, 39, 83, 159)$
- $(1, 35, 69, 103, 137, 171, 205, 239)$
- $(254, 220, 186, 152, 118, 84, 50, 16)$
- $(26, 43, 60, 77, 94, 111, 112, 129)$
- $(146, 163, 180, 197, 214, 231, 248, 9)$

For all keys the χ^2 test values were between 3976 and 4183, hence in the acceptable range for the χ^2-distribution with 4096 degrees of freedom and 5 percent significance level (approximately 4245). The numerical values were almost all between 12250 and 12750. Hence, one can conclude that the strict avalanche criterium [4] is satisfied.

Key-Ciphertext Avalanche

We fix the bytes $(x_0, \ldots, x_7)$ and analyze the effect of the remaining 8 bytes (the key bytes) on the output. Any regular properties exhibited here could be used to simplify attempts at key reconstruction.

As before, the number of times that each output bit differed by 1 was counted for each of 25000 input pairs that differ in exactly one bit. The same 12 vectors as above were also used here as fixed byte vectors.

For each of the 12 fixed byte vectors the χ^2 test values were between 4043 and 4242, hence within the acceptable range. The numerical values were almost all between 12250 and 12750. As a result, this test gives no indications of any weaknesses of $E^{(3)}$ with respect to the strict avalanche criterium.

5.4 The MAGENTA Algorithm

Plaintext-Ciphertext Avalanche

For each of 10000 plaintext pairs that differ in exactly one bit, the number of times that each output bit differed by 1 was counted. In this manner, estimations of the investigated probabilities (up to normalization) were obtained. The $128 \cdot 128$ values were then subjected to a χ^2 test.

The following 9 vectors were used as fixed key values:

- $(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$
- $(18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18)$
- $(255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255)$
- $(171, 171, 171, 171, 205, 205, 205, 205, 205, 205, 205, 205, 171, 171, 171, 171)$
- $(121, 121, 228, 228, 121, 121, 228, 228, 121, 121, 228, 228, 121, 121, 228, 228)$
- $(1, 35, 1, 35, 1, 35, 1, 35, 69, 69, 69, 69, 69, 69, 69, 69)$
- $(212, 236, 144, 236, 118, 168, 92, 66, 201, 40, 143, 18, 198, 39, 83, 159)$
- $(1, 35, 69, 103, 137, 171, 205, 239, 254, 220, 186, 152, 118, 84, 50, 16)$
- $(26, 43, 60, 77, 94, 111, 112, 129, 146, 163, 180, 197, 214, 231, 248, 9)$

For all keys the χ^2 test values were between 16204 and 16668, hence in the acceptable range for the χ^2-distribution with 16384 degrees of freedom and 5 percent significance level (approximately 16683). The numerical values were almost all between 4850 and 5150. Hence, one can conclude that the strict avalanche criterium [4] is satisfied.

Key-Ciphertext Avalanche

We fix the plaintext bytes and analyze the effect of the key components on the resulting ciphertext. As above, the number of times that each output bit differed by 1 was counted for each of 10000 key pairs that differ in exactly one bit. The same 9 vectors used above were used as fixed plaintexts.

For each of the 9 fixed plaintexts the χ^2 test values were between 16144 and 16572, hence within the acceptable range for a 5 percent significance level. The numerical values were almost all between 4850 and 5150. Hence, these tests do not show any significant weaknesses in the MAGENTA algorithm with respect to the strict avalanche criterium.

6 Statistical Tests

In this section we describe a number of statistical tests that were applied to the MAGENTA algorithm, as well as their evaluation [5]. As a starting point the tests in [6] were used, since these tests allow one to compare the results to those of several other block ciphers.

For fixed keys, the following properties were tested:

1. the XOR-sums from 10000 randomly chosen plaintext blocks $(x_0, \ldots, x_{15})$ with the corresponding ciphertext blocks (Test series 1),
2. the ciphertexts for all 16514 plaintext blocks $(x_0, \ldots, x_{15})$ in which at least 126 bits are 0, and respectively at least 126 bits are 1 (Test series 2).

In each test series segments of 128 bits were analyzed according to the following criteria (for a more exact description, see [6]):

1. frequencies of zeros and ones,

2. frequencies of the bigrams 00, 01, 10, and 11,
3. number of subsegments consisting of equal bits in relation to the frequencies of zeros and ones,
4. number of distinct subsegments ("patterns"),
5. frequency of ones in the binary derivative.

Up to and including the fourth test, two different significance levels (up to 5 percent) were applied to each test, resulting in 9 tests for each 128-bit block. The adaptation of the test statistics from 64 to 128 bits causes difficulties in computing exact test boundaries and threshold values. Hence, the analysis is based on test values from randomly generated 128-bit segments.

Test Results

Both test series were conducted with the 9 keys from Section 5.4. The analysis was based on the subset of tested 128-bit blocks that did not fulfill the specified test criterium.

In the first test series, the deviation of the values corresponding to the investigated subset from the previously computed values was insignificant. Hence, we found no statistical dependences between the plaintext bytes and the corresponding ciphertexts.

Key	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9
1	2.67	0.62	5.33	1.14	4.08	0.84	4.66	4.64	1.06
2	2.51	0.54	4.99	1.02	3.91	0.73	4.70	4.75	1.06
3	2.56	0.52	5.04	0.93	4.17	0.69	4.36	4.63	1.17
4	2.53	0.54	5.12	1.00	4.29	0.74	5.08	4.86	1.00
5	2.43	0.50	5.08	1.01	4.07	0.83	4.95	4.64	0.86
6	2.94	0.61	5.23	1.01	4.07	0.66	5.26	4.50	0.93
7	2.79	0.61	4.97	1.00	3.91	0.72	4.72	4.51	0.98
8	2.26	0.49	4.65	0.88	4.02	0.66	4.95	4.46	1.08
9	2.57	0.58	4.77	1.02	4.03	0.73	4.65	4.56	0.96

For the sake of comparison, we give here the evaluation results for 4 equal sized sets of randomly generated 128-bit blocks:

Set	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9
1	2.68	0.61	5.14	1.17	4.28	0.78	4.51	4.53	0.98
2	2.90	0.59	5.31	1.05	3.79	0.70	4.76	4.69	1.10
3	2.82	0.49	5.04	0.82	3.74	0.64	4.90	4.69	1.09
4	2.70	0.61	5.24	1.03	4.03	0.81	4.82	4.73	1.03

The analysis of the second test series resulted in a similar picture to that of the first test series. With respect to the test criteria, the set of computed ciphertext blocks was indistinguishable from equally sized sets of randomly generated blocks, even though the plaintext was very structured.

Key	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9
1	2.75	0.66	5.40	0.93	3.92	0.81	4.56	4.61	0.98
2	2.58	0.56	4.96	0.97	3.90	0.65	5.01	4.63	1.02
3	2.68	0.69	4.93	1.01	3.71	0.67	4.90	4.77	0.95
4	2.73	0.57	5.04	1.02	3.76	0.70	5.08	4.57	1.05
5	2.71	0.59	4.75	0.88	3.74	0.62	4.52	4.56	1.01
6	2.51	0.57	5.23	1.14	4.05	0.78	4.71	4.54	0.91
7	2.80	0.70	4.99	0.95	3.81	0.70	4.71	4.73	1.01
8	2.53	0.54	4.97	1.00	4.06	0.75	4.81	4.80	1.04
9	2.77	0.58	4.97	0.99	3.98	0.66	4.72	4.56	0.95

A visual examination of the data for both test series does not reveal any abnormalities, even for the zero key. Therefore, we conclude that these tests have revealed no statistical weaknesses of the MAGENTA algorithm.

7 Differential Cryptanalysis

Differential cryptanalysis consists of methods and techniques for recovering keys used for rounds in block ciphers. Statements about single round keys (usually the last round) are derived based on ciphertext pairs whose corresponding plaintexts have the same pre-selected difference, i.e., this is a chosen-plaintext attack. In general, these methods are constructed by analyzing the effects of plaintext differences from round to round. In [2], this technique is described in detail with respect to the DES algorithm.

In general, extensive analysis of the round functions is required in order to find the most effective method. In [2], the XOR-tables (the S-Boxes) are used for this purpose. From these tables, statements about the probability of the transition of one difference to the next are derived. These probabilities have considerable influence on how many ciphertext pairs are necessary for a successful attack.

Since then, differential cryptanalysis methods have been applied to several block ciphers and hash functions [4]. It appears that at least part of the MAGENTA algorithm can be analyzed with these methods. In the following, we will consider especially the function $E^{(3)}$.

Similarly to the S-box XOR-tables in [2], the complete XOR-tables for the function f have been computed. In other words, for each pair (pre-image difference, image difference), the number of pre-image pairs from B^2 with the specified difference whose image pair from B^2 has the specified image difference was counted. By "difference" we mean the XOR-sum [2]. This concept of difference is natural, since in the cipher many XOR combinations are applied.

The XOR-table for the function f has 256 rows and 256 columns. The sum of every row and ever column is equal to 256. The table entries are all integers. The first row, which corresponds to the pre-image difference 0, contains exactly one entry of 256 (image difference 0) and all other entries are 0. Similarly, the first column, which corresponds to the image difference 0, contains exactly one entry of 256 (pre-image difference 0, and 0 elsewhere (due to the fact that f is one-to-one). Especially large values (except in the first row) are of special interest in differential cryptanalysis. The largest value in the table was 8, and it occurred for the following difference pairs:

pre-image difference	image difference
51	35, 66, 154, 155, 250
102	111, 114, 232, 233, 244
153	96, 97, 115, 229, 247
204	18, 19, 38, 207, 251

The rest of the table entries are 0, 2, 4, and 6. The maximal "transition probability" for non-zero differences is $8/256 = 2^{-5}$. Therefore, one must fix the transition probability in every level of the map T no higher than 2^{-5}, because every pre-image byte runs through f at least once.

In order to get a better understanding of the possibilities for making combinations of the function PE, the largest entries of the XOR-table of the map PE have also been determined. The largest entry outside the first row was 36 for the pre-image difference $(234, 234)$ and image difference $(0, 0)$. On this basis, one obtains as an upper bound for the transition probability within one level (for non-zero pre-image differences) the value $36/65336 \approx 1/1820$.

The maximum possible transition probability over the entire map T is 2^{-20}. However, the diffusion properties of T (see Section 4.4) and the fact that the 4 given pre-image differences of f do not correspond to the 20 given image differences suggest that smaller probabilities should be expected when more levels are combined.

Consequently, one must assume a transition probability considerably less than 2^{-60} for the map $E^{(3)}$. One exception is the case where the second or third pass through the map T operates on input with difference 0. This could raise the probability, however only to some value under 2^{-40}.

Hence, attempts to apply the theory of [2] to one Feistel round result in transition probabilities considerable less than 2^{-40}. This approximation is rather rough, but the exact values are even smaller. One can assume that the transition probabilities are smaller than those of DES [2].

Since the number of required ciphertext pairs is larger than the reciprocal of the respective transition probability, no practical differential cryptanalytical attack on the MAGENTA cipher is obtainable. This estimation does not preclude the possibility that combining these methods with other currently unknown attacks might result in a more successful attack.

With respect to the current state-of-the-art, the MAGENTA cipher is secure against differential cryptanalytical attacks based on XOR-tables. It is still open whether the use of other differences result in stronger attacks.

8 Linear Cryptanalysis

Linear cryptanalysis deals with attempts to reconstruct the key based on affine linear approximations of the relationships between plaintext bits, ciphertext bits, and key bits. Chosen-plaintext attacks like those used in differential cryptanalysis are in general not required here, provided that a large number of ciphertexts and corresponding plaintexts are known. However, the success of these attacks can depend on how irregular the plaintexts are distributed in the plaintext-space.

Up to now, linear cryptanalysis attacks on DES [13, 12] and FEAL [14, 1, 15] have been published. The suitability of certain partial maps of the round functions for affine linear approximations was utilized for both of these algorithms — within the five S-boxes for DES and within the maps S_0 and S_1 for FEAL. The search for a similar part of the MAGENTA algorithm leads one to the analysis of the non-linearity properties of the map f.

Analogous to [1], the linear approximation table of f was computed. For each of the non-linear Boolean functions $f_0, \dots, f_7$ and every XOR sum $f_0 \oplus f_1, f_0 \oplus f_2, \dots, f_0 \oplus f_1 \oplus \dots \oplus f_7$ as well as all the linear functions, it was determined how many digits of the values in the table agree with the corresponding digits in the linear function under consideration. In total, 255^2 values between 0 and 256 were computed. The table entries were normalized by subtracting 128.

The largest and smallest table entries for each column were determined. All entries were between -24 and 26. These values can be considered "normal," since they roughly correspond to those expected on average from randomly selected Boolean functions. The value 26 occurred with the following linear combinations:

$$\begin{array}{ll} f_1 \oplus f_2 \oplus f_3, & f_1 \oplus f_3 \oplus f_4 \oplus f_6, \\ f_1 \oplus f_3 \oplus f_6, & f_2 \oplus f_3 \oplus f_4 \oplus f_6, \\ f_1 \oplus f_4 \oplus f_7, & f_2 \oplus f_3 \oplus f_5 \oplus f_6, \\ f_2 \oplus f_6 \oplus f_7, & f_3 \oplus f_4 \oplus f_5 \oplus f_6 \quad . \end{array}$$

A linear approximation is described in [1] for part of the five S-boxes which is correct with probability 12/64. This means that the negation of this linear function correctly approximates the Boolean function under consideration with probability $52/64 = 0.8125$. This is a significant deviation from the probability 0.5. In contrast, the best affine linear approximation for parts of the function f are correct with probability $(128 + 26)/256 \approx 0.6$.

In [1, p.353], a formula for evaluating combinations of affine linear approximations is given. The probability that the combination of approximations is correct is $1/2 + 2^{l-1} \prod_{i=1}^{l} p_i$, when l affine linear approximations with probabilities $1/2 + p_i, 1 \leq i \leq l$ are combined. Applying this to the map $E^{(3)}$ using $p_i = 0.6$ ($l = 12$) results in a value approximately $1/2 + 2 \cdot 10^{-9}$. This value is an upper bound for the best affine linear approximation for $E^{(3)}$. The real maximal value is presumably lower, but "optimal" combinations from level to level were not investigated. As in Section 7, it is expected here that the diffusion properties of T (see Section 4.4) considerably hinder the search for useful combinations.

According to [1], approximately p^{-2} plaintext-ciphertext pairs are required in order to use an affine linear approximation, when the approximation is correct with probability $1/2 + p$. This means that an attack on the function $E^{(3)}$ would require at least $2.5 \cdot 10^{17}$ pairs. Hence, no practical attacks using linear cryptanalysis are foreseeable for the MAGENTA algorithm. Furthermore, the entries in the linear approximation table confirm the non-linearity of f mentioned in Section 4.1.

9 Palindrome Properties

In the MAGENTA algorithm, six or eight rounds of the so-called Feistel permutation

$$(X_1, X_2) \mapsto (X_2, X_1 \oplus F_Z(X_2))$$

are applied in the form of the function $F_y(x)$ (Equation 8). The sequence of round-keys y forms a palindrome: K_1, K_1, K_2, K_2, K_1, K_1 for 128-bit keys, K_1, K_2, K_3, K_3, K_2, K_1 for 192-bit keys, and K_1, K_2, K_3, K_4, K_4, K_3, K_2, K_1 for 256-bit keys.

A property of this type was investigated in [18] for DES. Similar cases were considered in [11] and [16] from the point of view of constructing pseudo-random permutations. The most important statement for the MAGENTA algorithm from these sources is that for all keys K (128, 192, or 256 bit) and $(x_0, \ldots, x_{15}) \in B^{16}$ the following holds:

$$V\left((Enc_K(x_0, \ldots, x_{15})\right) = Enc_K^{-1}(V(x_0, \ldots, x_{15})), \qquad (12)$$

where $V(x_0, \ldots, x_{15}) = (x_8, x_9, \ldots, x_{15}, x_0, x_1, \ldots, x_7)$. It is therefore simple to represent the decryption function in terms of the encryption function. This is an advantage when implementing the algorithm (in contrast to other block-ciphers), because the sequence of round-keys is the same for encryption and decryption. On the other hand, it may be desirable to separate users with only encryption and respectively decryption permissions, and the property described in (Equation 12) offers the possibility to circumvent this separation.

It is not possible to meet the concept of "pseudo-randomness" [11, 16] with this property, since randomly generated permutations do not have this property with higher probability. From current knowledge, cryptographic weaknesses can only be found in this context when OFB-mode with the map $V \circ Enc$ is used. The arising addition sequence would have a period of at most 256.

Another useful result from [18] states that for every key $(y_0, \ldots, y_{15}) \in B^{16}$ there are at least 2^{64} plaintext blocks with the property $Enc_{(y_0,\ldots,y_{15})}(x_0, \ldots, x_{15}) = V(x_0, \ldots, x_{15})$. It is expected that this set of plaintext blocks depends on the key and is difficult to describe. There are also no clues which indicate that these sets contain considerably more than 2^{64} plaintext blocks. Hence, there is no reason to believe that this property causes any cryptographic weaknesses in the MAGENTA algorithm. The analogous property for DES shows that the cipher has at least 2^{32} fixed points for certain special keys. As a result, only statements about cycles were derived in [18].

This property implies that the cycle representation of the MAGENTA algorithm is composed of at least 2^{63} cycles. The average cycle length, which plays a role if OFB-mode is used, is expected to be in the same range as the maximal cycle length of DES [9] (somewhere between 2^{63} and 2^{64}). That is less than what is possible for 128-bits, due to the fact that a randomly generated permutation over $\{0,1\}^{128}$ has no more than 120 cycles with high probability [17], but it should be enough for cryptographic purposes.

The special cases $K_i = K_j$ for all sizes of keys must be given extra attention. Since the same permutation is conducted six times, it is expected that the average cycle length is shorter than in the general case. If a cycle length of a Feistel round is divisible by 6, then this cycle splits into 6 shorter cycles when six rounds of this type are applied.

In conclusion, the palindromic properties of the MAGENTA algorithm do give rise to certain special properties. However, by paying attention to these properties (for example, excluding 128-bit keys with $K_1 = K_2$), it seems that they cause no significant cryptographic weaknesses.

10 Conclusions

To date, no significant security weaknesses are known for the MAGENTA block cipher algorithm. It seems to have sufficient avalanche properties, no statistical weaknesses have been found, and attacks based on differential and linear cryptanalysis do not offer any significant improvement over a brute force attack. More analysis is certainly required, especially for 192 and 256 bit keys, but since the building blocks of the algorithm appear to be cryptographically sound we don't expect these additional key sizes to introduce any weaknesses. Based on our current knowledge we can hypothesize that the work factor required to break a

MAGENTA system is of the order of the key space, i.e., 2^{127} for 128-bit keys, 2^{191} for 192-bit keys, and 2^{255} for 256-bit keys.

There are some algebraic and palindromic properties which yield weak keys, namely those keys $K = (K_1, \ldots, K_j)$, $(j \in \{2,3,4\}$ with two or more identical sub-keys K_i, and keys with one or more sub-keys having all 8 bytes identical. However, such weak keys are rare and can easily be avoided in an implementation.

Although designed as a block cipher, there are other applications for which MAGENTA is suitable. For example, it can easily be used for a stream cipher algorithm by following the standard procedures for any general block cipher algorithm, for example, by using it in 1-bit cipher feedback mode. MAGENTA is also suitable for use as a pseudo-random number generator. One can set the key and plaintext at a fixed random bit pattern and take some convenient subset of these input bits as counter bits. Then, using the algorithm reproducible pseudo-random numbers can be generated. However, some of the algebraic properties of MAGENTA, especially those of the function PE (Equation 3), simplify the construction of collisions when MAGENTA is applied as a hash function or MAC generator, so MAGENTA is not suitable for either of these applications [5].

MAGENTA is very well-suited for implementation in a wide variety of environments. The entire algorithm can be implemented completely using bit operations, which ensures efficient software implementations (in computer languages with the bit operation capability) and ease of implementation in hardware. Since the basic data unit is the 8-bit byte, MAGENTA is also particularly suitable for small smart-card processors. The algorithm can also be optimized for small storage space. The size of the table of 256 bytes representing the function f (Equation 1) can be traded against execution speed, for example, by storing only every l-th value of the complete 256 byte table used and computing any required value by additional shifts. The whole algorithm also fits into the fast-address area of most processors which leads to fast implementations. Due to the convenient data format, the small storage space necessary, and the fast encryption speed, the algorithm is also very suitable for applications in ATM, HDTV, B-ISDN, voice and satellite applications.

A further advantage of MAGENTA is that it contains no obvious hiding places for trap doors. As mentioned in the introduction, the algorithm was built on simple and transparent design techniques, and there are no magic tables or constants. The only possible bone of contention is the choice of the primitive polynomial $p(x) = x^8 + x^6 + x^5 + x^2 + 1$ used for the function f (Equation 1). In fact any one of the 16 primitive polynomials of $GF(256)$ could have been applied.

References

[1] E. Biham. On Matsui's linear cryptanalysis. In *Proc. EUROCRYPT '94*, volume 658 of *Lecture Notes in Computer Science*, pages 81–91, 1995.

[2] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

[3] H. Feistel, W. Notz, and J.L. Smith. Some cryptographic techniques for machine-to-machine data communications. *Proceedings of the IEEE*, 63(11):1545–1553, November 1975.

[4] W. Fumy and H.P. Rieß. *Kryptographie.* R. Oldenbourg Verlag, München Wien, second edition, 1994.

[5] SIT GmbH. Abschlußbericht - Untersuchung eines universellen Kryptoalgorithmus. Technical report, SIT GmbH, 1994.

[6] H. Gustafson, E. Dawson, and B. Caelli. Comparison of block ciphers. In *Proc. AUSCRYPT '90*, volume 453 of *Lecture Notes in Computer Science*, pages 208–220, 1990.

[7] K. Huber. Neue Kryptographische Verfahren durch Kombination von Operationen in endlichen Kp̈rpern mit der schnellen Walshtransformation. Unpublished manuscript, 1990.

[8] K. Huber and S. Wolter. Telekom's MAGENTA algorithm for en-/decryption in the gigabit/sec range. In *ICASSP 1996 Conference Proceedings*, volume 6, pages 3233–3235, 1996.

[9] R.R. Juenemann. Analysis of certain aspects of output feedback mode. In *Proc. CRYPTO '92*, pages 99–127. Plenum Press, 1983.

[10] S.Y. Kung. *VLSI Array Processors.* Prentice Hall, 1988.

[11] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Computing*, 17(2):373–386, April 1988.

[12] M. Matsui. The first experimental cryptanalysis of the Data Encryption Standard. In *Proc. CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 1–11, 1994.

[13] M. Matsui. Linear cryptanalysis method for DES cipher. In *Proc. EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397, 1994.

[14] M. Matsui and A. Yamagishi. A new method for known plaintext attack of FEAL cipher. In *Proc. EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 81–91, 1993.

[15] K. Ohta and K. Aoki. Linear cryptanalysis of the Fast Data Encipherment Algorithm. In *Proc. CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 12–16, 1994.

[16] J. Pieprzyk and B. Sadeghiyan. *Design of Hashing Algorithms*, volume 756 of *Lecture Notes in Computer Science.* Springer, 1993.

[17] W.N. Satschkow. *Wahrscheinlichkeitsmethoden in der kombinatorischen Analysis (russ.).* Verlag Nauka,, Moscow, 1978.

[18] G.J. Simmons and J.H. Moore. Cycle structure of the DES for keys having palindromic (or antipalindromic) sequences of round keys. *IEEE Transactions on Software Engineering*, SE-13(2):262–273, 1987.

[19] J.P Vaseur. Verschlüsselungsanordnung mit Mischverdrahtung. Deutsches Patentamt Auslegeschrift 1148397, Anmeldetag: 2.Juni 1959, Auslegeschrift: 9.Mai 1963, Anmelder: Compagnie Generale de Telegraphie sans Fil, Paris.

[20] Rodnay Zaks. *Programming the Z80.* SYBEX Inc., Berkeley, USA, 1980.

A Assembler Code

The first function is an assembler version of the C function memcpy. This function copies a number of consecutive bytes from one location in memory to another. In MAGENTA, this function is used to copy blocks of either 8 or 16 bytes. The assembler code is essentially that in [20, p.441], with the exception that we have unrolled the for-loop.

```
MEMCPY: LD  IX,FROM
        LD  IY,TO
NEXT:   LD  A,(IX)
        LD  (IY),A
        INC IX
END:    INC IY
        repeat NEXT to END 7 or 15 more times
```

The second function implements setting up a key. It records the size of the key and uses the memcpy function to copy the raw key data to two, three, or four sub-key locations required for the encryption and decryption operations, depending on the key size.

```
KEY:    LD  A,(KSIZEIN)
        LD  (KSIZE),A
        memcpy first 8 byte block of key to location K0
        memcpy second 8 byte block of key to location K1
        SUB A,128
        JP  Z,DONE
        memcpy third 8 byte block of key to location K2
        SUB A,64
        JP  Z,DONE
        memcpy fourth 8 byte block of key to location K3
DONE:
```

The third function initializes an array of 256 bytes starting at FTABLE which represents the function f (Equation 1). This code is based on an optimized C implementation. The register B is initialized to 1 and left-shifted until the carry bit is set, at which point the value in B is reduced by XOR-ing it with the constant $Q = 101$. This part of the code is executed exactly 128 times.

```
INIT:   LD  HL,FTABLE
        LD  B,1
        LD  C,255
FOR:    LD  (HL),B
        SLA B
        JP  NC,ENDIF
        LD  A,B
        XOR Q
        LD  B,A
ENDIF:  INC HL
        DEC C
        JP  NZ,FOR
END:    LD  (HL),0
```

The last function implements Π (Equation 5). The input is the 16-byte block starting at the address DATA and the output is written back to the same location at the end of the function. TEMP is assumed to be a 16-byte memory block used as temporary storage by the function. A table of 256 bytes representing the function f (Equation 1) is assumed to have been loaded starting at FTABLE.

```
PI:     LD  IX,DATA
        LD  IY,TEMP
NEXT:   LD  E,(IX+8)
        LD  D,0
        LD  HL,FTABLE
        ADD HL,DE
        LD  A,(HL)
        XOR (IX)
        LD  HL,FTABLE
        LD  E,A
        LD  D,0
        ADD HL,DE
        LD  B,(HL)
        LD  (IY),B
        INC IY
        LD  E,(IX)
        LD  D,0
        LD  HL,FTABLE
        ADD HL,DE
        LD  A,(HL)
        XOR (IX+8)
        LD  HL,FTABLE
        LD  E,A
        LD  D,0
        ADD HL,DE
        LD  B,(HL)
        LD  (IY),B
        INC IX
END:    INC IY
        repeat NEXT to END 7 more times
        memcpy $16$ bytes from TEMP to DATA
```