Supporting Document of MISTY1

Version 1.10

Mitsuru Matsui

Mitsubishi Electric Corporation

September 25, 2000

1. Introduction

The encryption algorithm MISTY1 is a 64-bit block cipher with 128-bit key, whose detailed specification was first published in Japan in 1996 [1] and then presented at the international workshop of Fast Software Encryption in 1997 [2]. Its design principles and security assessments by the designer can be also found in these papers. Needless to say, the designer did not insert any hidden weakness in MISTY1. MISTY1 has been, since then, publicly evaluated from both academic and practical viewpoints; no security flaws have been found. The second paper [2] is attached to this document for reference. MISTY1 has a variable number of rounds; we recommend an 8-round version, which is most commonly used in real applications.

MISTY1 is one of the earliest block ciphers that were aimed at being suitable for software and hardware systems as well, particularly for low cost applications. It was designed so that practical encryption performance could be achieved even in heavy resource constraints. For instance, MISTY1 can be implemented with a very small RAM size (e.g. within 100 bytes) in software on cheap processors, and with a small number of gate counts and low power consumption (e.g. within 10 K gates) in hardware.

MISTY1 is now widely used in many commercial and governmental applications. In software, its target platforms range from 8-bit microprocessors for smart cards and 16-bit digital signal processors for wireless communication systems to high-end 32-bit and 64-bit processors for PKI applications. Also, encryption LSI's with MISTY1 are being used in public transportation systems and high-speed network applications (e.g. hubs and routers). Recently the 3rd generation partnership project (3GPP) adopted a variant of MISTY1, which is called KASUMI, as a mandatory algorithm in the confidentiality and integrity mechanisms of the forthcoming W-CDMA systems.

It is undoubted that 64-bit block ciphers will be eventually shifted to 128-bit block ciphers. However, it is also true that for the time being 64-bit block ciphers will coexist with 128-bit block ciphers because of market requirements of low cost implementation and message format compatibility. We believe that 64-bit block ciphers still are worthy of being standardized internationally and MISTY1 is worthwhile.

MISTY1 is a patented algorithm. However, the owner of its essential patent, Mitsubishi Electric Corporation, has declared that it will give a license without any loyalty fee. For more details, see http://www.mitsubishi.com/ghp_japan/misty/licensee.htm.

2. Security Analysis of MISTY1

MISTY1 was designed on the basis of the theory of “provable security” against differential and linear cryptanalysis [3][4][5]. It was also carefully designed to withstand various cryptanalytic attacks that were known at the time of designing. The detailed design principles and methodologies of MISTY1 to balance its security and performance in various platforms can be found in [2].

Since the publication of MISTY1, many research efforts have been done for evaluating its security level; no security flaws of MISTY1 have been so far found for the recommended 8-round version. In this section we describe, among them, several results about security level of round-reduced versions of MISTY1 that were not fully covered by the designer’s papers.

2.1 Higher order differential cryptanalysis

Higher order differential cryptanalysis was proposed by Lai [6] and developed by Knudsen [7], which is effectively applicable to a block cipher whose algebraic degree is small. MISTY1 contains two look-up tables S7 and S9, both of which have a low algebraic degree, specifically 3 and 2, respectively. Although these look-up tables with the low algebraic degrees contribute to realizing small and fast hardware, it is also natural that many papers concentrated on higher order differential cryptanalysis of MISTY1. The best-known result among them is a successful attack of MISTY1 reduced to five rounds without FL-functions [8]. However the algebraic structure of MISTY1 with FL-functions significantly varies according to the key value due to bit-wise AND/OR operations used in the FL-functions, and hence we believe that it is unlikely that higher order differential cryptanalysis can be applied to the full 8-round MISTY1 with FL-functions.

2.2 Slide attack

The slide attack, which was invented by Biryukov and Wagner [9], is applicable to a block cipher where the same subkey value is supplied to each round (or every n-th round). This attack was strongly motivated by related-key cryptanalysis introduce by Biham [10]. Since MISTY1 has a simple key scheduler for hardware reasons, it is also natural to pay an attention to related key cryptanalysis. The only known observation about related key attacks of MISTY1 is that, if all FL-functions are removed from MISTY1, the slide attack works when one of 65536 keys is used [11]. However, this attack is weaker than a trivial exhaustive key search to find one of the 65536 keys and does not work for other keys. Also MISTY1 with FL-functions withstand this attack. Note that the slide attack is newer than MISTY1, but protection from related-key type cryptanalysis was one of the design principles of the key-scheduling algorithm of MISTY1.

2.3 Impossible differential cryptanalysis

Impossible differential cryptanalysis is a recent attack, which was proposed by Biham and Shamir [12] after the motivation given by Knudsen in his paper about an analysis of DEAL cipher [13]. Impossible differential cryptanalysis pays attention to characteristic paths that never appear, and narrows possible key candidates using this information. This attack, revealed that any Feistel cipher with a bijective round function has impossible characteristic paths in five rounds, which enables us to distinguish the five round cipher from a randomly chosen permutation. This result is a good contrast to the Luby-Rackoff analysis that showed that a four round Feistel cipher with a pseudo-random round function is a pseudo-random permutation and hence is impossible to distinguish from a randomly given permutation. No results about impossible differential cryptanalysis specific to MISTY1 are known; just a general theory tells us that MISTY1 reduced to five rounds without FL-functions can distinguish from a randomly chosen permutation. However again due to the existence of the FL-functions, impossible characteristic paths become heavily dependent on the key value. We hence believe that it is extremely unlikely that impossible differential cryptanalysis is applicable to the full MISTY1.

2.4 Luby-Rackoff style evaluation

The Luby-Rackoff style evaluation of a block cipher is to look into (complexity theoretic) pseudo-randomness of the entire structure of the cipher assuming pseudo-randomness of its smallest components [14]. Several works are known about the Luby-Rackoff style security assessment of the “MISTY1-like” structure. Note that this type of analysis does not direct lead to an attack of a block cipher itself, since the security is asymptotically evaluated in terms of its block size and the attacker’s computational power is limited to polynomial time of its block size. On the other hand, meeting the pseudo-randomness criteria might be considered as an evidence of (sort of) soundness of the basic structure. It is known that purified and idealized “MISTY1-like” structure is a pseudo-random permutation when the number of rounds is five or more, if the smallest components, which correspond to the two loop-up tables S7 and S9, are assumed to be pseudo-random permutations. For more details, see [15].

2.5 Implementation attacks

Implementational attacks, such as timing attacks and power analysis, are often powerful and realistic threats in smart card applications. Since MISTY1 is composed of logical operations and lookup tables only, it is easy to apply standard methods to avoid implementation weakness of block ciphers, for instance removing “jumps” or “variable-cycle instructions”, to MISTY1. However, differential power analysis is a much deeper attack using subtle hardware characteristics. We believe that it is too early to say any conclusive statements about resistance to differential power attack.

3. Implementation Analysis of MISTY1

3.1 Software implementations

The sample C source code attached to the designer's paper [1] shows an example of straightforward implementation based on the description of MISTY1. There are however many more techniques for implementing MISTY1 in software balancing speed and memory requirements. The designer and developers of MISTY1 have published several results about how to implement MISTY1 on various processors [16][17][18]. Here we describe further updated software performance information on MISTY1.

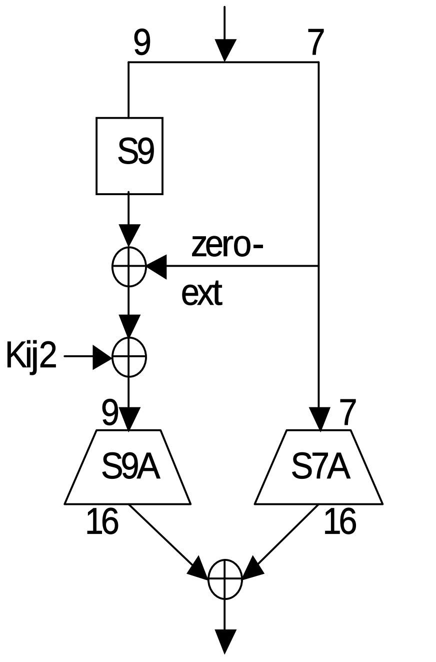

Fig 1: An Alternative Form of FI

For instance, it is not difficult to see that the FI-function can be also written as shown in fig.1, where S9A and S9B are newly introduced look-up tables that transform 9 and 7 input bits into 16 output bits, respectively. Note that the subkey Kij1 can be "embedded" into other subkeys. Though implementing this form requires more memory than the straightforward method, but faster speed is expected. Moreover, noting that the number of possible varieties of Kij2 is only eight, we can even remove the subkey by introducing eight different S9A tables if a target processor has on-chip cache of 16K bytes or more.

Table 1 summarizes implementation results of MISTY1 on Pentium II processor.

Implementation Method	Language	Key Setup Time	Encryption Time
Straightforward	C	170 cycles	450 cycles/block = 56.25 cycles/byte
Fig.1 with Kij2	C	300 cycles	300 cycles/block = 37.50 cycles/byte
Fig.1 without Kij2 (self modified code)	Assembly	8700 cycles	190 cycles/block = 23.75 cycles/byte

Table 1: Software Performance of MISTY1 (Pentium II 500MHz 128MB Memory / Windows 98)

For another implementation example, Sano et al. showed a software implementation of MISTY1 on a typical 8-bit microprocessor Z80 [19]. According the paper, MISTY1 can be implemented on the processor in 1598 ROM bytes and 44 RAM bytes. It worked in 25486 cycles for one block encryption with key scheduling, which is approximately 5 msec in 5 MHz. This result shows that MISTY1 is highly practical in smart card applications. Note that this implementation is based on a straightforward method. Further speeding-up can be expected in return for some increase in ROM size by using the above reduction form.

3.2 Hardware implementations

MISTY1 is suitable not only for high-speed applications but also for low cost applications with small number of gate counts and low power consumption. MISTY1 is composed of only two small basic components, namely the FI-function and the FL-function. Hence by using repeatedly these functions one can design small hardware. The designer and developers of MISTY1 have published several results about how to implement MISTY1 on various processors [20][21].

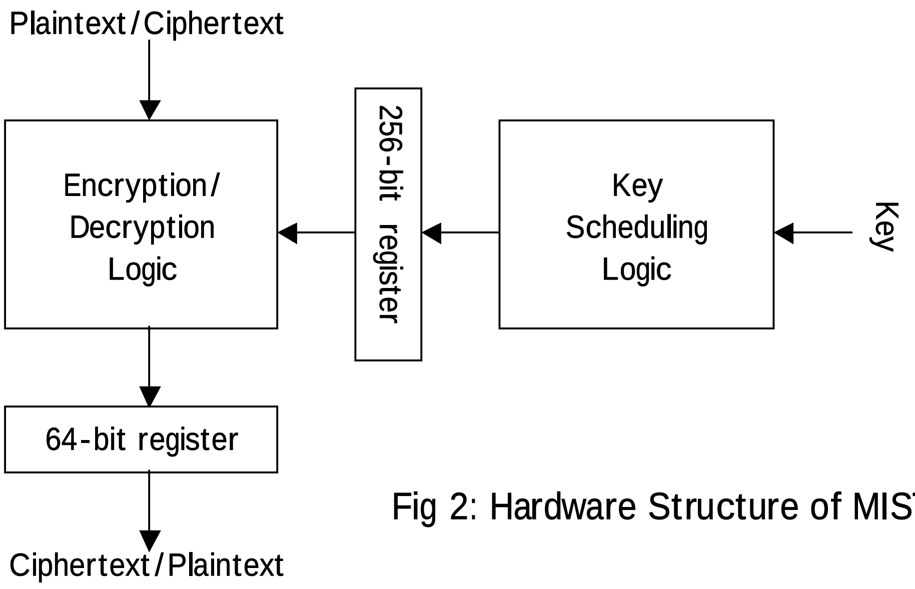

Fig 2: Hardware Structure of MISTY1

A recent example of hardware implementation of MISTY1 realizes 7.6 Kgates at an encryption speed of 72Mbit/sec, which can encrypts one block plaintext in 35 cycles. Another implementation works at an encryption speed of 800Mbps in approximately 50Kgates, whose structure is able to encrypt one block plaintext in one cycle (see fig.2). Both circuits, which contain full encryption, decryption and key-scheduling functions without pipeline architecture --- hence feedback modes can be applied ---, were designed using Mitsubishi’s 0.35-micron ASIC design library and evaluated as a worst-case speed. We believe that this is in the smallest and fastest level among currently used secure 64-bit block ciphers.

4. Further Information of MISTY1

As was written in the first section, MISTY1 is now used in various real applications. Including some of them, information of MISTY1 and related encryption technology can be seen at http://www.security.melco.co.jp/.

5. References

[1] M. Matsui, “Block Encryption Algorithm MISTY”,
Technical Report of IEICE, ISEC96-11 (1996) [in Japanese]

[2] M. Matsui, “New Block Encryption Algorithm MISTY”,
FSE’97, LNCS 1267 (1997)

[3] K. Nyberg, L. Knudsen, “Provable Security against Differential Cryptanalysis”,
Journal of Cryptology, Vol.8, no.1 (1995)

[4] K. Nyberg, “Linear Approximation of Block Ciphers”,
Eurocrypt’94, LNCS 950 (1995)

[5] M. Matsui, “New Structure of Block Ciphers with Provable Security against
Differential and Linear Cryptanalysis”, FSE’96, LNCS 1039 (1996)

[6] X. Lai, “Higher order derivatives and differential cryptanalysis”,
In Proceeding of "Symposium on Communication, Coding and Cryptography",
in honor of James L. Massey on the occasion of his 60'th birthday (1994)

[7] L. Knudsen, “Truncated and Higher Order Differentials”,
FSE’95, LNCS 1008 (1994)

[8] H. Tanaka, H. Hisamatsu, K. Kaneko, “Higher Order Differential Attack of
MISTY1 without FL functions”,
Japan-Singapore Joint Workshop on Information Security, JWIS’98 (1998)

[9] A. Biryukov and D. Wagner, “Slide Attacks”, FSE’99, LNCS 1636 (1999)

[10] E. Biham, “New Types of Cryptanalytic Attacks Using Related Keys”,
Journal of Cryptology, Vol.7 (1994)

[11] S. Furuya and K. Takaragi, “Slide Attacks Combined with Known-Plaintext Attacks”,
Symposium on Cryptography and Information Security SCIS’2000 (2000) [in Japanese]

[12] E. Biham, A. Biryukov, A. Shamir, “Cryptanalysis of Skipjack Reduced to 31
Rounds Using Impossible Differentials”, Eurocrypt’99, LNCS 1592 (1999)

[13] L. Knudsen, “DEAL – A 128-bit Block Cipher”
Technical report no. 151, Department of Informatics, University of Bergen, Norway,

(1998)

[14] M. Luby, C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions", SIAM J. Computation, Vol.17, no.2 (1988)

[15] M. Sugita, "On pseudo-randomness of the block cipher MISTY1", Technical Report of IEICE, ISEC97-109 (1997)

[16] J. Nakajima, M. Matsui, "On Fast Software Implementation of MISTY (I)", Technical Report of IEICE, ISEC97-12 (1997) [in Japanese]

[17] J. Nakajima, M. Matsui, "On Fast Software Implementation of MISTY (II)", Symposium on Cryptography and Information Security SCIS'98 (1998) [in Japanese]

[18] J. Nakajima, M. Matsui, "Fast Software Implementations of MISTY1 on Alpha Processors", IEICE Trans. Fundamentals, Vol.E82-A, no.1 (1999)

[19] F. Sano, M. Koike, S. Kawamura, M. Shiba, "Performance Evaluation of AES Finalists on the High-End Smart Card", Proceedings of the third Advanced Encryption Standard conference (2000)

[20] T. Ichikawa, T. Sorimachi, M. Matsui, "On Hardware Design of Secret Key Ciphers", Symposium on Cryptography and Information Security SCIS'97 (1997) [in Japanese]

[21] T. Ichikawa, J. Katoh, M. Matsui, "A Method for Hardware Implementation of MISTY1", Symposium on Cryptography and Information Security SCIS'98 (1998) [in Japanese]

New Block Encryption Algorithm MISTY

Mitsuru Matsui

Information Technology R&D Center
Mitsubishi Electric Corporation
5-1-1, Ofuna, Kamakura, Kanagawa, 247, Japan
matsui@iss.isl.melco.co.jp

Abstract. We propose secret-key cryptosystems MISTY1 and MISTY2, which are block ciphers with a 128-bit key, a 64-bit block and a variable number of rounds. MISTY is a generic name for MISTY1 and MISTY2. They are designed on the basis of the theory of provable security against differential and linear cryptanalysis, and moreover they realize high speed encryption on hardware platforms as well as on software environments. Our software implementation shows that MISTY1 with eight rounds can encrypt a data stream in CBC mode at a speed of 20Mbps and 40Mbps on Pentium/100MHz and PA-7200/120MHz, respectively. For its hardware performance, we have produced a prototype LSI by a process of 0.5μ CMOS gate-array and confirmed a speed of 450Mbps. In this paper, we describe the detailed specifications and design principles of MISTY1 and MISTY2.

1 Fundamental Design Policies of MISTY

Our purpose of designing MISTY is to offer secret-key cryptosystems that are applicable to various practical systems as widely as possible; for example, software stored in IC cards and hardware used in fast ATM networks. To realize this, we began its design with the following three fundamental policies:

1. *MISTY should have a numerical basis for its security,*
2. *MISTY should be reasonably fast in software on any processor,*
3. *MISTY should be sufficiently fast in hardware implementation.*

For the first policy, we have adopted the theory of provable security against differential and linear cryptanalysis [1][2][4], which was originally introduced by Kaisa Nyberg and Lars Knudsen. As far as we know, MISTY is the first block encryption algorithm designed for practical use with provable security against differential and linear cryptanalysis. Although this advantage does not mean information theoretic provable security, we believe that it is a good starting point for discussing secure block ciphers.

Secondly, we have noticed the fact that many recent block ciphers were designed so that they could be fastest and/or smallest on specific targets; for example, 32-bit microprocessors. This often results in slow and/or big implementation on other types of processors. Since we regarded seeking applicability to various systems as more important than pursuing maximum performance on specific targets, we decided to design a cipher that could be reasonably fast and small on any platform, and hence not to adopt software instructions that are effective on special processors only.

For the last policy, we should note that DES is reasonably fast in both software and hardware, while many recent ciphers are seriously slow and/or big when they are implemented in hardware because of their software-oriented structure. On the other hand, since one of our target systems is a fast ATM network of several hundreds Mbps, which cannot be reached in software for the present, we have carefully optimized the look-up tables of MISTY from the viewpoint of its hardware performance. It should be also noted that, in general, a choice of substitution tables does not significantly affect their software execution speed; i.e. memory access time.

2 Discussions on Basic Operations

In this section we classify basic operations that are frequently used in block ciphers into four categories and discuss their applicability to MISTY in terms of compatibility between their security level and software/hardware efficiency.

- **Logical Operations**
  Logical operations such as AND, OR and especially XOR are most common components of secret-key ciphers and are clearly small and fast in any software or hardware system. However we cannot expect much security of them.
- **Arithmetic Operations**
  Arithmetic operations such as additions, subtractions and sometimes multiplications are also commonly used in software-oriented ciphers because they can be carried out by one instruction on many processors and fairly contribute to their security. However, in hardware, their effect on data diffusion is not necessarily high enough, considering their encryption speed, since their delay time due to carry-spreading is often long and expensive.
- **Shift Operations**
  Shift operations, especially rotate-shifting, are frequently used in designing secret-key ciphers. They indirectly improve data diffusion, and in hardware they are obviously cheap and fast if the number of shift counts is fixed. We should note, however, that software performance of shift operations heavily depends on their target size; for instance, when a rotate shift of 32-bit data is executed on 8-bit or 16-bit microprocessors, its speed may be quite slow.
- **Look-up Tables**
  In software, efficiency of loop-up tables strongly depends on memory access speed. In early microprocessors, memory access was much more expensive than register access, while many recent processors can read from and write to memory in one cycle (or often less than one cycle due to parallel processing) under certain conditions. On the other hand, in hardware, the use of ROM is slow in general, but if the tables are optimized for direct construction

by logic gates, their delay time can be drastically reduced. Moreover, as for the security, the look-up table method clearly contributes to data diffusion effectively.

Taking the above discussion into consideration, we have concluded that logical operations and look-up tables arranged in terms of security level and hardware performance meet our design policies and hence they are desirable as basic components of MISTY.

3 Theory of Provable Security

This section briefly summarizes the theory of provable security against differential and linear cryptanalysis. For more detail, see [4]. This theory forms a basis of the security of MISTY.

Definition 1. Let $F_k(x)$ be a function with an n-bit input x and an ℓ-bit parameter k. We define average differential probability DP^F and average linear probability LP^F of the function F as

$$DP^F \overset{def}{=} \frac{1}{2^\ell} \sum_k \max_{\Delta x \neq 0, \Delta y} \frac{\#\{x | F_k(x) \oplus F_k(x \oplus \Delta x) = \Delta y\}}{2^n}, \qquad (1)$$

$$LP^F \overset{def}{=} \frac{1}{2^\ell} \sum_k \max_{\Gamma x, \Gamma y \neq 0} \left(2 \frac{\#\{x | x \bullet \Gamma x = F_k(x) \bullet \Gamma y\}}{2^n} - 1\right)^2, \qquad (2)$$

respectively. We also apply this definition to a function $F(x)$ without the parameter k by setting $\ell = 0$.

When $F_k(x)$ is an encryption function with a key k, DP^F and LP^F represent a strict level of security of the function against differential and linear cryptanalysis, respectively. Since we can prove that F is secure against the two attacks when these values are small, we say that F is provably secure if DP^F and LP^F are proved to be sufficiently small.

The following three theorems give relationships between average differential/linear probability of a "small" function and that of a "large" function that is a combination of the small functions. That is to say, using these theorems, we can construct a "large and strong" function from "small and strong" functions. Theorem 2 was first proved for average differential probability by Nyberg and Knudsen [1], and then shown for average linear probability by Nyberg [2].

Theorem 2. *In figure 1, assume that each f_i is bijective and DP^{f_i} (resp. LP^{f_i}) is smaller than p. If the entire function F_k ($k = k_1 || k_2 || k_3 ...$) shown in the figure has at least three rounds, then DP^F (resp. LP^F) is smaller than p^2.*

Note: The authors of [1] originally proved $2p^2$ (not p^2) for a cipher with bijective f_i and at least three rounds, and for a cipher with any f_i and at least four rounds. Recently Aoki and Ohta improved this bound to p^2 when f_i is bijectvie [3].

We proved in [4] that the above theorem is valid for the algorithm shown in figure 2. An essential difference between figures 1 and 2 is that the functions f_i can be processed in parallel in figure 2, and consequently the structure of figure 2 is faster than that of figure 1.

Theorem 3. *In figure 2, assume that each f_i is bijective and DP^{f_i} (resp. LP^{f_i}) is smaller than p. If the entire function F_k ($k = k_1||k_2||k_3...$) shown in the figure has at least three rounds, then DP^F (resp. LP^F) is smaller than p^2.*

We found that a similar formula holds even if the input string is divided into two strings of unequal bit length. Specifically, consider the algorithm shown in figure 3, where the input string is divided into n_1 bits and n_2 bits ($n_1 \geq n_2$). Now assuming that in odd rounds the right n_2-bit string is zero-extended to n_1 bits before XOR-ed with the left n_1-bit string, and in even rounds the right n_1-bit string is truncated to n_2 bits before XOR-ed with the left n_2-bit string, we have the following general theorem [4]:

Theorem 4. *In figure 3, assume that each f_i is bijective and DP^{f_i} (resp. LP^{f_i}) is smaller than p. If the entire function F_k ($k = k_1||k_2||k_3...$) shown in the figure has at least three rounds, then DP^F (resp. LP^F) is smaller than*

$$max\{p_1p_2,\ p_2p_3,\ 2^{n_1-n_2}p_1p_3\}. \qquad (3)$$

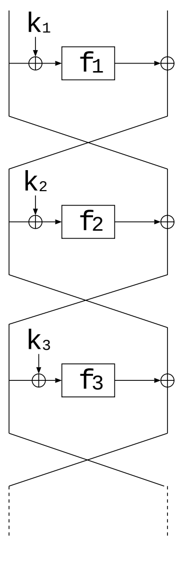

Figure 1.

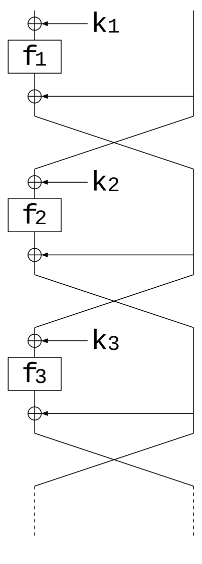

Figure 2.

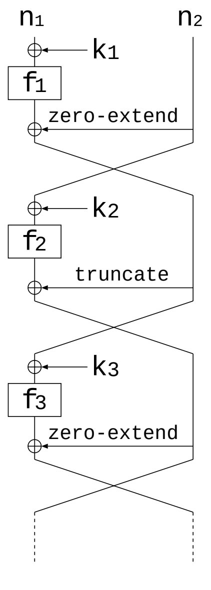

Figure 3.

4 Design of the Data Randomizing Part

In this section we discuss the structure of the data randomizing part of MISTY. For a complete description of MISTY1 and MISTY2, see an appendix.

4.1 The Framework

Our basic strategy in designing the data randomizing part of MISTY is to build the entire algorithm from small components using the methods shown in the previous section recursively. This enables us to easily evaluate the security level of the total algorithm by that of the small ones. For instance, let us apply the structure of figure 2 recursively to all f_i functions given in figure 2. In this case, if the average differential/linear probability of the smallest function is less than p, we can prove from theorem 3 that the probability of the entire algorithm is less than p^4.

Now by applying theorem 2 or theorem 3 to a 64-bit block cipher, where theorems 2 and 3 correspond to MISTY1 and MISTY2, respectively, we have a "small" function with 32-bit input/output, which is called an FO function in MISTY (figure 4). Next by applying theorem 2 again to the FO function, we have a "smaller" function with 16-bit input/output, which is referred to as an FI function in MISTY. Since the size of the FI function is still big to use as a look-up table, we have divided the 16-bit string into 9 bits and 7 bits, not 8 bits and 8 bits, using the algorithm given in figure 3.

This unequal division is due to the fact that bijective functions of odd size are generally better than those of even size from the viewpoint of provable security against differential and linear cryptanalysis. More specifically, when the size n of a function is odd, the possible minimal value of its average differential/linear probability is proved to be 2^{-n+1}, but when it is even, it is only conjectured that the possible minimal value is 2^{-n+2} (an open problem). Therefore, if we divide the 16-bit into 8 bits and 8 bits, the average differential/linear probability of the entire 64-bit cipher is proved to be less than $(((2^{-8+2})^2)^2)^2=2^{-48}$ (on condition that the above conjecture is correct), while if we divide it into 9 bits and 7 bits, then we can guarantee that the probability is less than $((2^{-9+1}2^{-7+1})^2)^2=2^{-56}$ from theorem 4 whenever all subkey bits are independent.

This shows that an unequal division generally has an advantage for security against differential and linear cryptanalysis. On the other hand, it has two penalties in implementation; the first is an obstruction to parallel computation, and the second is a decrease in software performance caused by handling data with an odd number of bits. We have nevertheless adopted the unequal division because of its security. In the following, we refer to the first and third functions of the lowest level as S_9, and the second function as S_7, which are "smallest" components of MISTY. For reducing the size of software, we use the same table in the first and third rounds.

In both MISTY1 and MISTY2, for the sake of flexibility of their security level, the number of rounds n of level 1 (see figure 4) is variable on condition that n is a multiple of four, while that of levels 2 and 3 is fixed to three rounds. Now compare encryption/decryption speed of MISTY1 and MISTY2. If we do not take any parallel processing into consideration, the total complexity of MISTY1 and MISTY2 with the same number of rounds is clearly the same; however if we allow parallel computations, their encryption speed is not the same. This is mainly because MISTY1 can carry out two FI's at a time, while MISTY2 can execute four FI's in parallel.

Table 1 gives encryption/decryption time of MISTY1 and MISTY2, where each entry shows the number of calculations of S_9 assuming the computation time of S_7 is the same as that of S_9. For simplicity we have ignored the time for XOR operations. It is clearly seen from table 1 that MISTY2 is faster than MISTY1 in encryption, but MISTY1 is faster in ECB and CBC decryption. This is because parallel computations are impossible in inverse calculation of MISTY2. MISTY2 is therefore suitable for OFB and CFB modes.

	Encryption	Decryption	Decryption
	ECB,CBC,OFB,CFB	ECB,CBC	OFB,CFB
n-round MISTY1	$3n$	$3n$	$3n$
n-round MISTY2	$1.5n$	$9n$	$1.5n$

Table 1. Encryption/Decryption time of MISTY1 and MISTY2 (number of calculations of S_9).

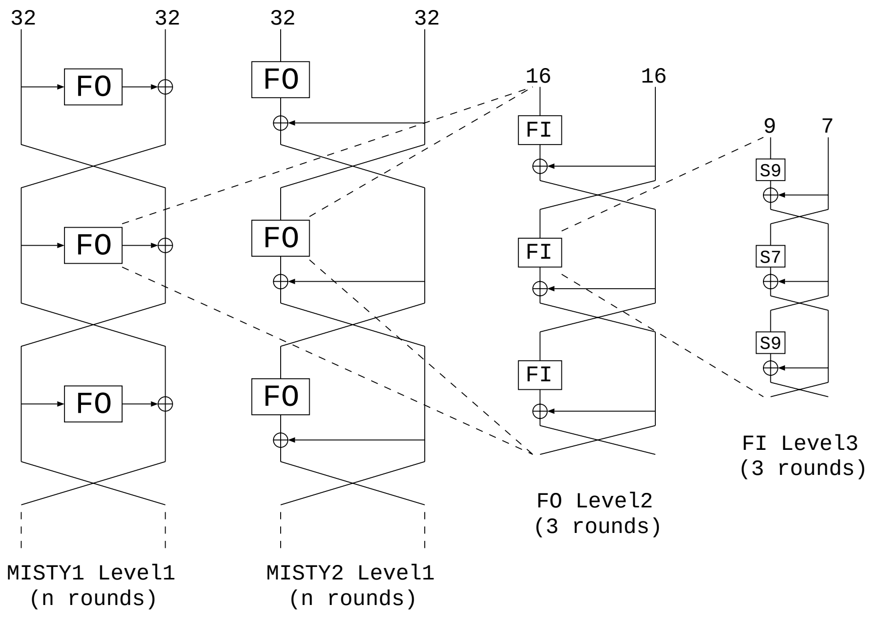

Figure 4: Recursive structure of MISTY

4.2 S_7 and S_9

In selecting S_7 and S_9, we have the following three criteria:

1. *Their average differential/linear probability must be minimal,*
2. *Their delay time in hardware is as short as possible,*
3. *Their algebraic degree is high, if possible.*

For the first criterion, a sequence of power functions over finite fields is known to attain the minimal value (that is, 2^{-6} for S_7 and 2^{-8} for S_9), and as far as we know, this is the only example that can be obtained in a systematic way. Hence we first planned to investigate the hardware delay, whose exact definition we adopted will be given below, for all functions that have the form $S_i(x) = A \circ x^{\alpha} \circ B$ $(i = 7, 9)$, where A and B are arbitrary bijective linear transformations and α is an integer such that $(2^i - 1, \alpha) = 1$. The last equality is a necessary and sufficient condition that a power function can be bijective.

However, because it was time-consuming for us to calculate the delay for all functions above, we next restricted our search to the functions that have the form $S_i(x) = A \circ x^{\alpha}$ $(i = 7, 9)$ and have a polynomial basis or a normal basis over $GF(2)$. In other words, we investigated all possible linear transformations for A and a limited number of linear transformations for B. Note that the average differential/linear probability does not depend on a selection of A or B, but the delay does. Now the following is our formal definition of the hardware delay and the algebraic degree of $S_i(x)$:

Definition 5. For a function $y = f(x)$ with an i-bit input $x = (x_0, x_1, x_2, ..., x_{i-1})$ and a j-bit output $y = (y_0, y_1, y_2, ..., y_{j-1})$, we call the following equation an algebraic normal form of the a-th output bit y_a of f:

$$\begin{aligned} y_a = e^{(a,0)} + \sum_{0 \le k_1 < i} e^{(a,1)}_{k_1} x_{k_1} + \sum_{0 \le k_1 < k_2 < i} e^{(a,2)}_{k_1,k_2} x_{k_1} x_{k_2} \\ + \sum_{0 \le k_1 < k_2 < k_3 < i} e^{(a,3)}_{k_1,k_2,k_3} x_{k_1} x_{k_2} x_{k_3} + ..., \end{aligned} \tag{4}$$

where $e^{(a,0)}, e^{(a,1)}_{k_1}, e^{(a,2)}_{k_1,k_2}, e^{(a,3)}_{k_1,k_2,k_3}$ .... are binary values, and the sum $\sum$ denotes an XOR operation.

The hardware length of y_a is defined as the number of non-zero terms of equation 4, and the hardware length of the function f is the maximal hardware length of all output bits of f. Also, the algebraic degree of y_a is defined as the maximal degree of equation 4, and the algebraic degree of the function f is the maximal algebraic degree of all output bits of f.

Note that the hardware length of y_a minus 1 is equivalent to the number of two-input XOR gates required for constructing y_a from x_k $(0 \le k \le i)$ in hardware, and the logarithm of the hardware length of f indicates its hardware delay time. Although we have to count the number of AND gates and fan-outs to see the exact delay time in hardware, we have adopted the above definition for simplicity. Also note that the algebraic degree of $S_i(x) = A \circ x^{\alpha} \circ B$ agrees with the binary hamming weight of α.

$\boxed{\text{Selection of } S_7}$

For all functions having the form $A \circ x^{\alpha}$ over $GF(2^7)$ with a polynomial or normal basis and $(2^7 - 1, \alpha) = 1$, we first calculated the algebraic degree and hardware length of each output bit; as a result, we obtained the following:

- If the algebraic degree is at least 4, then the hardware length of any output bit is at least 21.
- If the algebraic degree is equal to 3, then the hardware length of any output bit is at least 10.
- If the algebraic degree is equal to 2, then the hardware length of any output bit is at least 7.

Since we regarded the length as too long when the algebraic degree is four or more, we decided to adopt a function whose algebraic degree is equal to three. Then for all functions whose algebraic degree is three, we calculated their entire hardware length, and found that the minimal length is 13 and the function that attains this length is unique up to the order of output bits. Lastly, by adding a constant value to its output, we determined the final form of S_7, whose concrete logic is as follows:

$$\begin{aligned}
y_0 &= x_0 + x_1x_3 + x_0x_3x_4 + x_1x_5 + x_0x_2x_5 + x_4x_5 + x_0x_1x_6 + x_2x_6 + x_0x_5x_6 + x_3x_5x_6 + 1\\
y_1 &= x_0x_2 + x_0x_4 + x_3x_4 + x_1x_5 + x_2x_4x_5 + x_6 + x_0x_6 + x_3x_6 + x_2x_3x_6 + x_1x_4x_6 + x_0x_5x_6 + 1\\
y_2 &= x_1x_2 + x_0x_2x_3 + x_4 + x_1x_4 + x_0x_1x_4 + x_0x_5 + x_0x_4x_5 + x_3x_4x_5 + x_1x_6 + x_3x_6 +\\
&\quad x_0x_3x_6 + x_4x_6 + x_2x_4x_6\\
y_3 &= x_0 + x_1 + x_0x_1x_2 + x_0x_3 + x_2x_4 + x_1x_4x_5 + x_2x_6 + x_1x_3x_6 + x_0x_4x_6 + x_5x_6 + 1\\
y_4 &= x_2x_3 + x_0x_4 + x_1x_3x_4 + x_5 + x_2x_5 + x_1x_2x_5 + x_0x_3x_5 + x_1x_6 + x_1x_5x_6 + x_4x_5x_6 + 1\\
y_5 &= x_0 + x_1 + x_2 + x_0x_1x_2 + x_0x_3 + x_1x_2x_3 + x_1x_4 + x_0x_2x_4 + x_0x_5 + x_0x_1x_5 +\\
&\quad x_3x_5 + x_0x_6 + x_2x_5x_6\\
y_6 &= x_0x_1 + x_3 + x_0x_3 + x_2x_3x_4 + x_0x_5 + x_2x_5 + x_3x_5 + x_1x_3x_5 + x_1x_6 + x_1x_2x_6 +\\
&\quad x_0x_3x_6 + x_4x_6 + x_2x_5x_6
\end{aligned}$$

$\boxed{\text{Selection of } S_9}$

Similarly, for all functions having the form $S_9(x) = A \circ x^{\alpha}$ over $GF(2^9)$ with a polynomial or normal basis and $(2^9 - 1, \alpha) = 1$, we first calculated the algebraic degree and hardware length of each output bit; as a result, we had the following:

- If the algebraic degree is at least 3, then the hardware length of any output bit is at least 27.
- If the algebraic degree is equal to 2, then the hardware length of any output bit is at least 9.

Since we regarded the length as too long if the algebraic degree is three or more, we decided to adopt a function whose algebraic degree is equal to two. Then for all functions whose algebraic degree is two, we calculated their entire hardware length, and found that the minimal length is 12 and there are nine functions that attain this length up to the order of output bits. Lastly by selecting one of them randomly and adding a constant value to its output, we determined the final form of S_9, whose concrete logic is as follows:

$$
\begin{aligned}
y_0 &= x_0x_4 + x_0x_5 + x_1x_5 + x_1x_6 + x_2x_6 + x_2x_7 + x_3x_7 + x_3x_8 + x_4x_8 + 1\\
y_1 &= x_0x_2 + x_3 + x_1x_3 + x_2x_3 + x_3x_4 + x_4x_5 + x_0x_6 + x_2x_6 + x_7 + x_0x_8 + x_3x_8 + x_5x_8 + 1\\
y_2 &= x_0x_1 + x_1x_3 + x_4 + x_0x_4 + x_2x_4 + x_3x_4 + x_4x_5 + x_0x_6 + x_5x_6 + x_1x_7 + x_3x_7 + x_8\\
y_3 &= x_0 + x_1x_2 + x_2x_4 + x_5 + x_1x_5 + x_3x_5 + x_4x_5 + x_5x_6 + x_1x_7 + x_6x_7 + x_2x_8 + x_4x_8\\
y_4 &= x_1 + x_0x_3 + x_2x_3 + x_0x_5 + x_3x_5 + x_6 + x_2x_6 + x_4x_6 + x_5x_6 + x_6x_7 + x_2x_8 + x_7x_8\\
y_5 &= x_2 + x_0x_3 + x_1x_4 + x_3x_4 + x_1x_6 + x_4x_6 + x_7 + x_3x_7 + x_5x_7 + x_6x_7 + x_0x_8 + x_7x_8\\
y_6 &= x_0x_1 + x_3 + x_1x_4 + x_2x_5 + x_4x_5 + x_2x_7 + x_5x_7 + x_8 + x_0x_8 + x_4x_8 + x_6x_8 + x_7x_8 + 1\\
y_7 &= x_1 + x_0x_1 + x_1x_2 + x_2x_3 + x_0x_4 + x_5 + x_1x_6 + x_3x_6 + x_0x_7 + x_4x_7 + x_6x_7 + x_1x_8 + 1\\
y_8 &= x_0 + x_0x_1 + x_1x_2 + x_4 + x_0x_5 + x_2x_5 + x_3x_6 + x_5x_6 + x_0x_7 + x_0x_8 + x_3x_8 + x_6x_8 + 1
\end{aligned}
$$

4.3 The function FL

For the purpose of avoiding possible attacks other than differential and linear cryptanalysis, we have supplemented an additional simple function FL, whose design criteria are (1) to be a linear function for any fixed key and (2) to have a variable form depending on a key value.

Since this function is linear as long as the key is fixed, it does not affect the average differential/linear probability of the entire algorithm. Moreover, this function is obviously fast in both software and hardware since it is constructed by logical operations such as AND, OR and XOR only.

5 Design of the Key Scheduling Part

In designing the key scheduling part of MISTY, we set up the following criteria from the viewpoint of compatibility between its security level and applicability to various systems:

1. *The size of key is 128 bits,*
2. *The size of subkey is 256 bits,*
3. *Every round is affected by all key bits,*
4. *Every round is affected by as many subkey bits as possible.*

For security reasons we have adopted the 128-bit key, and for practical reasons we have limited the size of the subkey to 256 bits. Reducing the size of subkey has two important performance advantages. The first advantage can be obtained in systems whose resources are limited such as in IC cards. In these systems, since RAM size for temporary use is usually strictly limited, it is generally impossible to store all subkey bits in RAM if its size is large; hence we have to carry out the key scheduling part in every data block, which could be a heavy penalty on performance. We decided to choose subkeys of 256 bits, so that all the bits could be stored in RAM even for extremely restricted software environments.

The second advantage comes from the fact that in microprocessors with many integer registers such as RISC processors, the 256-bit subkey can be loaded completely into the registers. In most implementation of block ciphers, all subkey bits are written into memory in key scheduling process, and in encryption process they are read from the memory round by round. Hence if all the subkey bits are kept in the registers during the entire encryption process, the total performance is expected to be significantly improved.

On the other hand, in compensation for this small number of subkey bits and simple key scheduling algorithm, we have established the third and fourth design criteria. In MISTY, an FO function and an FL function use 112 subkey bits and 32 subkey bits, respectively. To generate the 112 subkey bits, all of 128 key bits are required. The number of total independent subkey bits of MISTY1 or MISTY2 with eight rounds, for example, is 1216.

6 Examples of Implementation of MISTY

In this section we show two examples of our software implementation and one example of our hardware implementation of MISTY1 with eight rounds.

6.1 Pentium

Pentium has two independent integer execution units called U-pipe and V-pipe, where the U-pipe is usually used for carrying out instructions. However some instructions can be also executed in the V-pipe while the U-pipe is being occupied by special "pairable" instructions. Though the number of these pairable instructions is small, if we write a program so that these two pipes can be efficiently used, the performance of the software is extremely improved, possibly twice or more due to resolution of register contentions.

We wrote an assembly language program of MISTY1 with eight rounds on Pentium 100MHz, which encrypts an input plaintext stream in CBC mode at a speed of 20Mbps. This is 30% faster than hightly optimized DES for Pentium. The program heavily uses V-pipe because of the highly parallel structure of MISTY; it takes approximately 300 cycles to process one block, where the U-pipe has no idle time and the V-pipe is used in more than 95% of the 300 cycles.

6.2 PA-7200

PA-7200 can also execute two integer instructions at a time under various restrictions. Moreover PA-RISC series microprocessors have 32 integer registers, almost all of which can be used freely by users; this means that it is easy to load all 256-bit subkey information of MISTY, even every 16 bits in each register.

PA-7200 has 512KB on-chip cache (256KB for code and 256KB for data), which enables us to reduce computational time of MISTY by having a big predefined table. That is to say, we can make a 128KB table that represents the first two rounds of the FI function in advance. By doing this, calculation of FI is significantly simplified. Note that this technique cannot be used in Pentium because Pentium has only small cache (8KB for data) which generally causes serious penalty cycles due to cache misses.

We wrote an assembly language program of MISTY1 with eight rounds on PA-7200 120MHz using the above techniques. It can encrypt an input plaintext stream in CBC mode at a speed of 40Mbps.

6.3 Hardware

We have also designed a prototype LSI of MISTY1 with eight rounds, which has the following specifications:

Encryption Speed:	*450Mbps (typical)*
Clock:	*14MHz*
I/O:	*32-bit parallel* $\times$ *3 (plaintext, ciphertext, key)*
Supported Modes:	*ECB,CBC,OFB-64,CFB-64*
Design Process:	*0.5μ CMOS gate-array*
Number of Gates:	*65K gates*
Package:	*208-pin flat package*

This LSI has no repetition structure; that is, it contains the full hardware of eight FO functions and ten FL functions. It takes two cycles to encrypt a 64-bit plaintext. It also has three independent 64-bit registers that store a plaintext, an intermediate text after the fourth round, and a ciphertext, respectively. This structure makes the following pipeline data processing possible:

	plaintext 1	*plaintext 2*	*plaintext 3*	*plaintext 4*
Cycles 1 and 2	*Input*			
Cycles 3 and 4	*Encryption*	*Input*		
Cycles 5 and 6	*Output*	*Encryption*	*Input*	
Cycles 7 and 8		*Output*	*Encryption*	*Input*

7 Conclusions

This paper proposed new secret-key block cryptosystems MISTY1 and MISTY2. At present, the author recommends to use MISTY1 with eight rounds, and to use MISTY2, which has a newer structure, with twelve rounds. The next four pages show a complete and self-contained description of MISTY1 and MISTY2.

References

1. Nyberg, K., Knudsen, L.,: Provable Security against Differential Cryptanalysis. Journal of Cryptology, Vol.8, no.1 (1995)
2. Nyberg, K.,: Linear Approximation of Block Ciphers. Advances in Cryptology – Eurocrypt'94, Lecture Notes in Computer Science **950**, Springer Verlag (1994)
3. Aoki, K., Ohta, K.,: Stricter Evaluation for the Maximum Average of Differential Probability and the Maximum Average of Linear Probability (in Japanese). Proceedings of SCIS'96, SCIS96-4A (1996)
4. Matsui, M.,: New Structure of Block Ciphers with Provable Security against Differential and Linear Cryptanalysis. Proceedings of the third international workshop of fast software encryption, Lecture Notes in Computer Science **1039**, Springer Verlag (1996)

Block Cipher Algorithms MISTY1 and MISTY2

Edition 2.1 December 16 1996

This document shows a complete description of encryption algorithms MISTY1 and MISTY2, which are secret-key ciphers with a 64-bit data block, a 128-bit secret key and a variable number of rounds n, where n is a multiple of four.

Data Randomizing Part

- Figure A and B show the data randomizing part of MISTY1 and MISTY2, respectively: The 64-bit plaintext P is divided into the left 32-bit string and the right 32-bit string, which are transformed into the 64-bit ciphertext C by means of bitwise XOR operations denoted by $\oplus$ and sub-functions FO_i $(1 \leq i \leq n)$ and FL_i $(1 \leq i \leq n+2)$. FO_i uses a 64-bit subkey KO_i and a 48-bit subkey KI_i. FL_i uses a 32-bit subkey KL_i.
- Figure C shows the structure of FO_i: The input is divided into the left 16-bit string and the right 16-bit string, which are transformed into the output by means of bitwise XOR operations and sub-functions FI_{ij} $(1 \leq j \leq 3)$, where KO_{ij} $(1 \leq j \leq 4)$ and KI_{ij} $(1 \leq j \leq 3)$ are the j-th (from left) 16 bits of KO_i and KI_i, respectively.
- Figure D shows the structure of FI_{ij}: The input is divided into the left 9-bit string and the right 7-bit string, which are transformed into the output by means of bitwise XOR operations and substitution tables S_7 and S_9. In the first and third XORs, the 7-bit string is zero-extended to 9 bits, and in the second XOR, the 9-bit string is truncated to 7 bits by discarding its highest two bits. KI_{ij1} and KI_{ij2} are the left 7 bits and the right 9 bits of KI_{ij}, respectively.
- Figure E shows the structure of FL_i. The input is divided into the left 16-bit string and the right 16-bit string, which are transformed into the output by means of bitwise XOR operations, a bitwise AND operation denoted by $\cap$ and a bitwise OR operation denoted by $\cup$, where KL_{ij} $(1 \leq j \leq 2)$ is the j-th (from left) 16 bits of KL_i.
- In the next page, the substitution tables S_7 and S_9 are shown in decimal form.

Key Scheduling Part

- Figure F shows the key scheduling part of MISTY1 and MISTY2: K_i $(1 \leq i \leq 8)$ is the i-th (from left) 16 bits of the secret key K, and K'_i $(1 \leq i \leq 8)$ is the output of FI_{ij} when the input of FI_{ij} is assigned to K_i and the key KI_{ij} is set to K_{i+1}, where K_9 is identified with K_1.
- The correspondence between the round subkeys KO_{ij},KI_{ij},KL_{ij} and the actual subkeys K_i,K'_i is as follows, where i is identified with $i-8$ when $i>8$:

Round	KO_{i1}	KO_{i2}	KO_{i3}	KO_{i4}	KI_{i1}	KI_{i2}	KI_{i3}	KL_{i1}	KL_{i2}
Actual	K_i	K_{i+2}	K_{i+7}	K_{i+4}	K'_{i+5}	K'_{i+1}	K'_{i+3}	$K_{\frac{i+1}{2}}$ (odd i) $K'_{\frac{i}{2}+2}$ (even i)	$K'_{\frac{i+1}{2}+6}$ (odd i) $K_{\frac{i}{2}+4}$ (even i)

Test Data of MISTY1 with eight rounds

Key (K_1 to K_8):	`00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff`
Subkey (K'_1 to K'_8):	`cf 51 8e 7f 5e 29 67 3a cd bc 07 d6 bf 35 5e 11`
Plaintext:	`01 23 45 67 89 ab cd ef`
Ciphertext:	`8b 1d a5 f5 6a b3 d0 7c`

Table of S7

```
27, 50, 51, 90, 59, 16, 23, 84, 91, 26,114,115,107, 44,102, 73,
31, 36, 19,108, 55, 46, 63, 74, 93, 15, 64, 86, 37, 81, 28,  4,
11, 70, 32, 13,123, 53, 68, 66, 43, 30, 65, 20, 75,121, 21,111,
14, 85,  9, 54,116, 12,103, 83, 40, 10,126, 56,  2,  7, 96, 41,
25, 18,101, 47, 48, 57,  8,104, 95,120, 42, 76,100, 69,117, 61,
89, 72,  3, 87,124, 79, 98, 60, 29, 33, 94, 39,106,112, 77, 58,
 1,109,110, 99, 24,119, 35,  5, 38,118,  0, 49, 45,122,127, 97,
80, 34, 17,  6, 71, 22, 82, 78,113, 62,105, 67, 52, 92, 88,125
```

Table of S9

```
451,203,339,415,483,233,251, 53,385,185,279,491,307,  9, 45,211,
199,330, 55,126,235,356,403,472,163,286, 85, 44, 29,418,355,280,
331,338,466, 15, 43, 48,314,229,273,312,398, 99,227,200,500, 27,
  1,157,248,416,365,499, 28,326,125,209,130,490,387,301,244,414,
467,221,482,296,480,236, 89,145, 17,303, 38,220,176,396,271,503,
231,364,182,249,216,337,257,332,259,184,340,299,430, 23,113, 12,
 71, 88,127,420,308,297,132,349,413,434,419, 72,124, 81,458, 35,
317,423,357, 59, 66,218,402,206,193,107,159,497,300,388,250,406,
481,361,381, 49,384,266,148,474,390,318,284, 96,373,463,103,281,
101,104,153,336,  8,  7,380,183, 36, 25,222,295,219,228,425, 82,
265,144,412,449, 40,435,309,362,374,223,485,392,197,366,478,433,
195,479, 54,238,494,240,147, 73,154,438,105,129,293, 11, 94,180,
329,455,372, 62,315,439,142,454,174, 16,149,495, 78,242,509,133,
253,246,160,367,131,138,342,155,316,263,359,152,464,489,  3,510,
189,290,137,210,399, 18, 51,106,322,237,368,283,226,335,344,305,
327, 93,275,461,121,353,421,377,158,436,204, 34,306, 26,232,  4,
391,493,407, 57,447,471, 39,395,198,156,208,334,108, 52,498,110,
202, 37,186,401,254, 19,262, 47,429,370,475,192,267,470,245,492,
269,118,276,427,117,268,484,345, 84,287, 75,196,446,247, 41,164,
 14,496,119, 77,378,134,139,179,369,191,270,260,151,347,352,360,
215,187,102,462,252,146,453,111, 22, 74,161,313,175,241,400, 10,
426,323,379, 86,397,358,212,507,333,404,410,135,504,291,167,440,
321, 60,505,320, 42,341,282,417,408,213,294,431, 97,302,343,476,
114,394,170,150,277,239, 69,123,141,325, 83, 95,376,178, 46, 32,
469, 63,457,487,428, 68, 56, 20,177,363,171,181, 90,386,456,468,
 24,375,100,207,109,256,409,304,346,  5,288,443,445,224, 79,214,
319,452,298, 21,  6,255,411,166, 67,136, 80,351,488,289,115,382,
188,194,201,371,393,501,116,460,486,424,405, 31, 65, 13,442, 50,
 61,465,128,168, 87,441,354,328,217,261, 98,122, 33,511,274,264,
448,169,285,432,422,205,243, 92,258, 91,473,324,502,173,165, 58,
459,310,383, 70,225, 30,477,230,311,506,389,140,143, 64,437,190,
120,  0,172,272,350,292,  2,444,162,234,112,508,278,348, 76,450
```

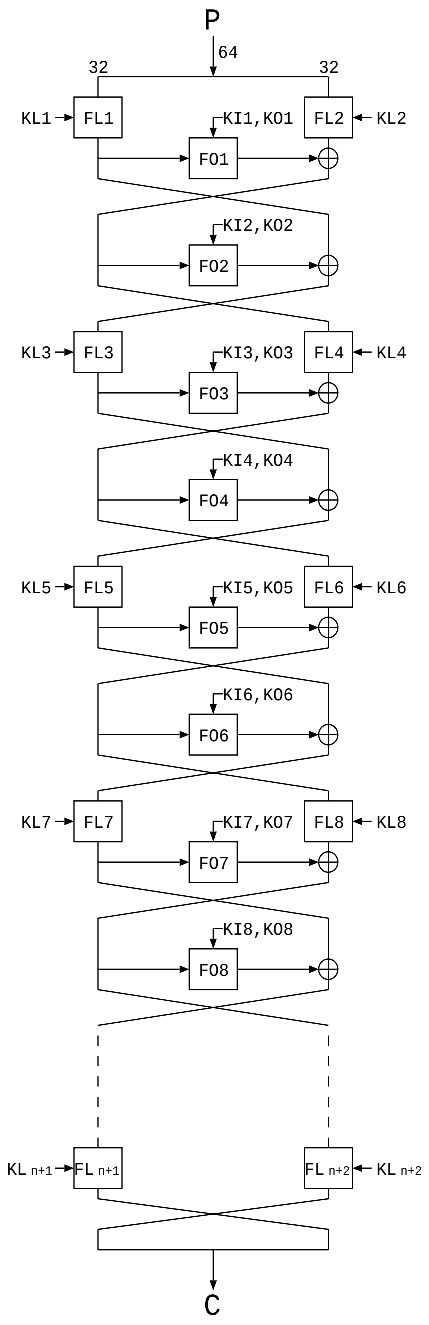

Figure A: MISTY1

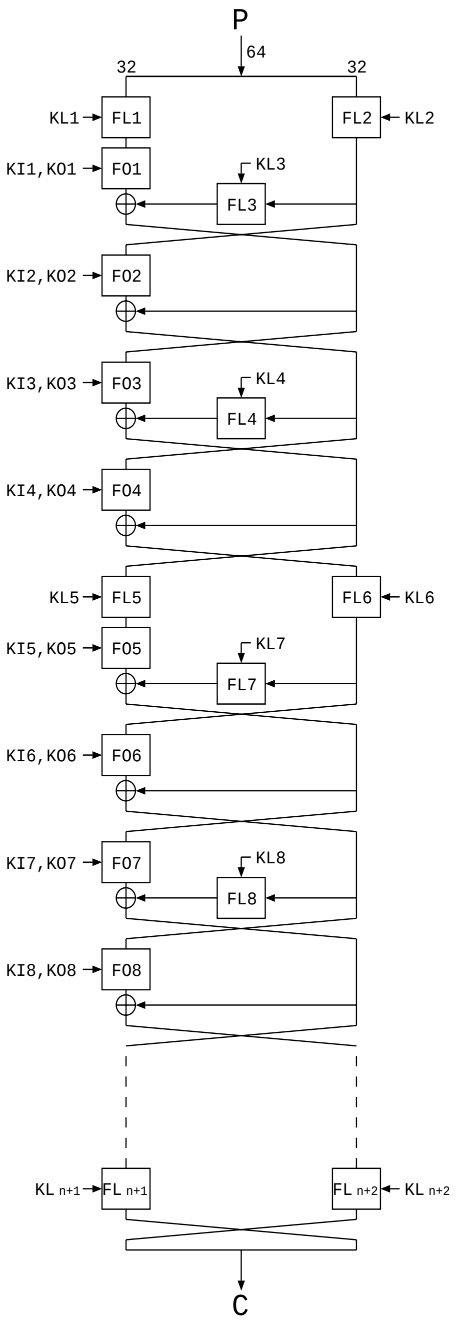

Figure B: MISTY2

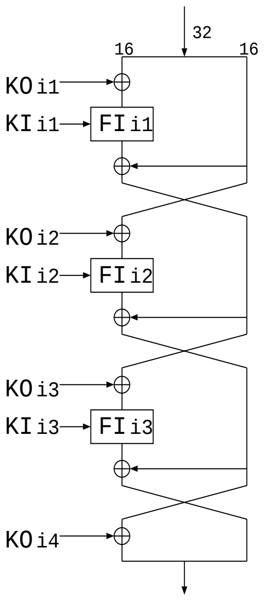

Figure C: FOi

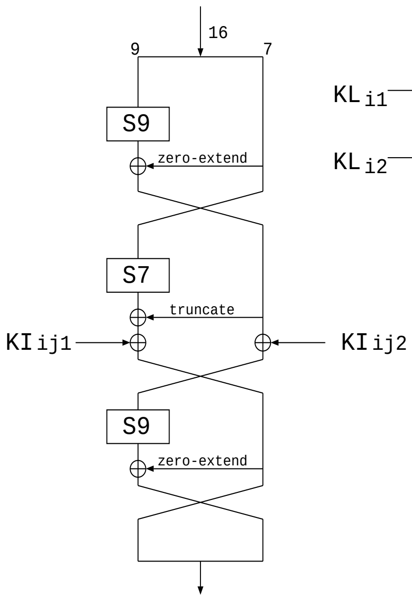

Figure D: FIij

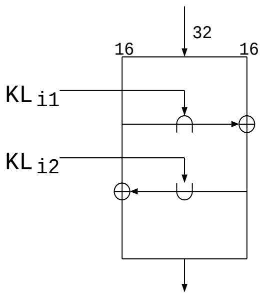

Figure E: FLi

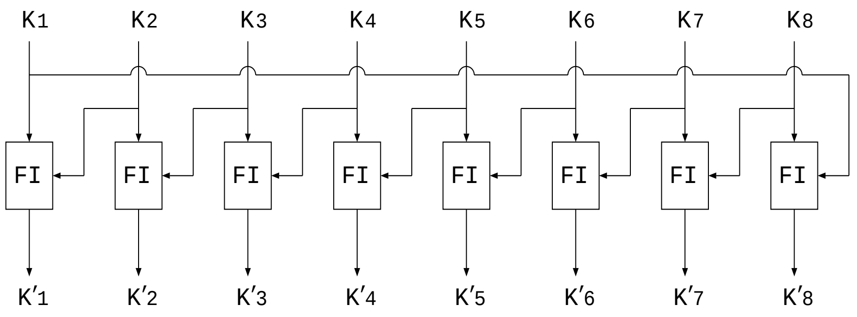

Figure F: Key Scheduling