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

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

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

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*

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

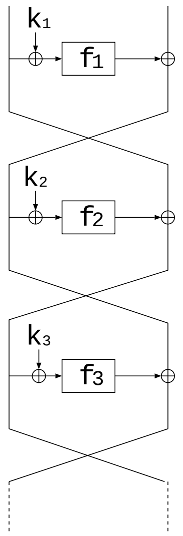

Figure 1.

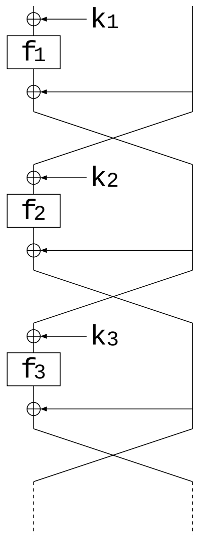

Figure 2.

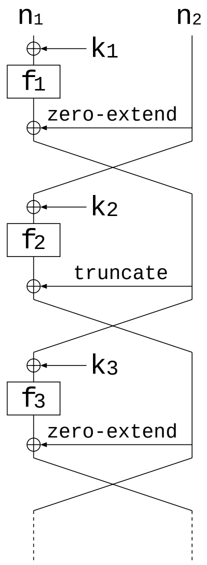

Figure 3.

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).

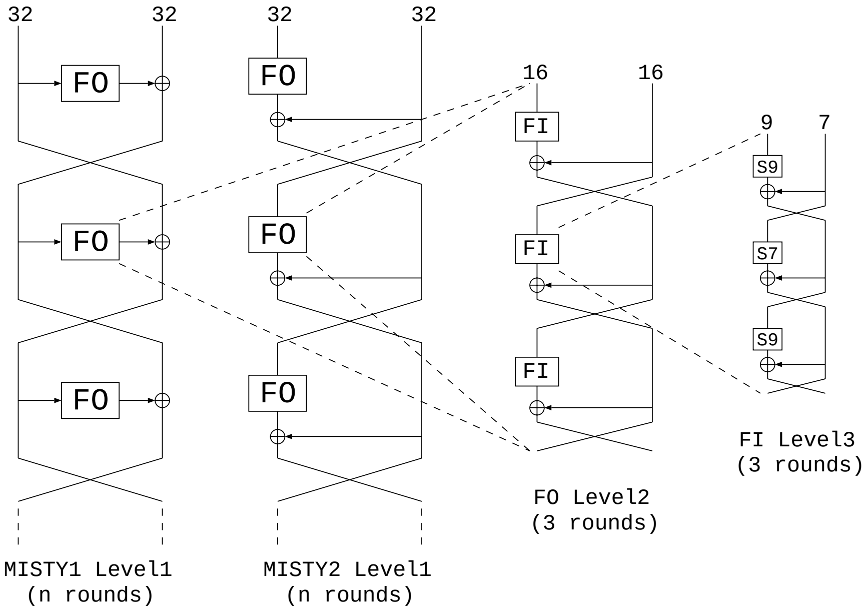

Figure 4: Recursive structure of MISTY

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.

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:

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

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 × 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)