

A Description of the MISTY1 Encryption Algorithm

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2000). All Rights Reserved.

Abstract

This document describes a secret-key cryptosystem MISTY1, which is block cipher with a 128-bit key, a 64-bit block and a variable number of rounds. It documents the algorithm description including key scheduling part and data randomizing part.

1. Introduction

This document describes a secret-key cryptosystem MISTY1, which is block cipher with a 128-bit key, a 64-bit block and a variable number of rounds. It is designed on the basis of the theory of provable security against differential and linear cryptanalysis, and moreover it realizes high-speed encryption on hardware platforms as well as on software environments. As the result of weighing strength and speed, 8-rounds of MISTY1 is recommended and used in most cases.

Our implementation shows that MISTY1 with eight rounds can encrypt a data stream in CBC mode at a speed of 57Mbps and 40Mbps on Pentium II/266MHz and PA-7200/120MHz, respectively. For its hardware performance, we have produced a prototype LSI by a process of 0.8-micron CMOS gate-array and confirmed a speed of 512Mbps.

2. Algorithm Description

Algorithm [1] could be divided into two parts, namely "key scheduling part" and "data randomizing part". Key scheduling part takes a 128-bit input key and produces a 128-bit expanded key. Data randomizing

part takes a 64-bit input data and mixes it, namely encryption. If data randomizing part is processed in reverse order, mixed data is transformed to input data, namely decryption.

2.1 Terminology

Some operators are used in this document to describe the algorithm. The operator '+' indicates two's complement addition. The operator '*' indicates multiplication. The operator '/' yields the quotient, and the operator '%' yields the remainder from the division. The operator '&' indicates bitwise AND operation. The operator '|' indicates bitwise inclusive OR operation. The operator '^' indicates bitwise exclusive OR operation. The operator '<<' indicates bitwise left shift operation. The operator '>>' indicates bitwise right shift operation.

2.2 Key Scheduling Part

Key scheduling part consists of the following operations.

```
for i = 0, ..., 7 do
    EK[i] = K[i*2]*256 + K[i*2+1];
for i = 0, ..., 7 do
begin
    EK[i+ 8] = FI(EK[i], EK[(i+1)%8]);
    EK[i+16] = EK[i+8] & 0x1fff;
    EK[i+24] = EK[i+8] >> 9;
end
```

K is an input key, and each element of K, namely K[i], holds an 8-bit of the key, respectively. EK denotes an expanded key, and each element of EK, namely EK[i], holds a 16-bit of the expanded key. Input data of K[0], ..., K[15] are copied to EK[0], ..., EK[7]. Expanded key is produced from EK[0], ..., EK[7] by using function FI, and stored in EK[8], ..., EK[15]. Function FI is described in the following section.

2.3 Data Randomizing Part

Data randomizing part uses two kinds of function, which are called function FO and function FL. Function FO calls another function, namely FI. The key expansion part also uses function FI. Function FI uses two S-boxes, namely S7, S9. Each function is described as follows.

Function FO takes two parameters. One is a 32-bit width input data, namely FO_IN. The other is an index of EK, namely k. And FO returns a 32-bit width data, namely FO_OUT.

```
FO(FO_IN, k)
begin
  var t0, t1 as 16-bit integer;
  t0 = FO_IN >> 16;
  t1 = FO_IN & 0xffff;
  t0 = t0 ^ EK[k];
  t0 = FI(t0, EK[(k+5)%8+8]);
  t0 = t0 ^ t1;
  t1 = t1 ^ EK[(k+2)%8];
  t1 = FI(t1, EK[(k+1)%8+8]);
  t1 = t1 ^ t0;
  t0 = t0 ^ EK[(k+7)%8];
  t0 = FI(t0, EK[(k+3)%8+8]);
  t0 = t0 ^ t1;
  t1 = t1 ^ EK[(k+4)%8];
  FO_OUT = (t1<<16) | t0;
  return FO_OUT;
end.
```

Function FI takes two parameters. One is a 16-bit width input data, namely FI_IN. The other is a part of EK, namely FI_KEY, which is also 16-bit width. And FI returns a 16-bit width data, namely FI_OUT.

```
FI(FI_IN, FI_KEY)
begin
  var d9 as 9-bit integer;
  var d7 as 7-bit integer;
  d9 = FI_IN >> 7;
  d7 = FI_IN & 0x7f;
  d9 = S9TABLE[d9] ^ d7;
  d7 = S7TABLE[d7] ^ d9;
  ( d7 = d7 & 0x7f; )
  d7 = d7 ^ (FI_KEY >> 9);
  d9 = d9 ^ (FI_KEY & 0x1ff);
  d9 = S9TABLE[d9] ^ d7;
  FI_OUT = (d7<<9) | d9;
  return FI_OUT;
end.
```

S7TABLE and S9TABLE denote the S-boxes S7 and S9 respectively in terms of look up table notation. Here are the description of S7TABLE and S9TABLE in hexadecimal notation.

S7TABLE:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00: | 1b | 32 | 33 | 5a | 3b | 10 | 17 | 54 | 5b | 1a | 72 | 73 | 6b | 2c | 66 | 49 |
| 10: | 1f | 24 | 13 | 6c | 37 | 2e | 3f | 4a | 5d | 0f | 40 | 56 | 25 | 51 | 1c | 04 |
| 20: | 0b | 46 | 20 | 0d | 7b | 35 | 44 | 42 | 2b | 1e | 41 | 14 | 4b | 79 | 15 | 6f |
| 30: | 0e | 55 | 09 | 36 | 74 | 0c | 67 | 53 | 28 | 0a | 7e | 38 | 02 | 07 | 60 | 29 |
| 40: | 19 | 12 | 65 | 2f | 30 | 39 | 08 | 68 | 5f | 78 | 2a | 4c | 64 | 45 | 75 | 3d |
| 50: | 59 | 48 | 03 | 57 | 7c | 4f | 62 | 3c | 1d | 21 | 5e | 27 | 6a | 70 | 4d | 3a |
| 60: | 01 | 6d | 6e | 63 | 18 | 77 | 23 | 05 | 26 | 76 | 00 | 31 | 2d | 7a | 7f | 61 |
| 70: | 50 | 22 | 11 | 06 | 47 | 16 | 52 | 4e | 71 | 3e | 69 | 43 | 34 | 5c | 58 | 7d |

S9TABLE:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 000: | 1c3 | 0cb | 153 | 19f | 1e3 | 0e9 | 0fb | 035 | 181 | 0b9 | 117 | 1eb | 133 | 009 | 02d | 0d3 |
| 010: | 0c7 | 14a | 037 | 07e | 0eb | 164 | 193 | 1d8 | 0a3 | 11e | 055 | 02c | 01d | 1a2 | 163 | 118 |
| 020: | 14b | 152 | 1d2 | 00f | 02b | 030 | 13a | 0e5 | 111 | 138 | 18e | 063 | 0e3 | 0c8 | 1f4 | 01b |
| 030: | 001 | 09d | 0f8 | 1a0 | 16d | 1f3 | 01c | 146 | 07d | 0d1 | 082 | 1ea | 183 | 12d | 0f4 | 19e |
| 040: | 1d3 | 0dd | 1e2 | 128 | 1e0 | 0ec | 059 | 091 | 011 | 12f | 026 | 0dc | 0b0 | 18c | 10f | 1f7 |
| 050: | 0e7 | 16c | 0b6 | 0f9 | 0d8 | 151 | 101 | 14c | 103 | 0b8 | 154 | 12b | 1ae | 017 | 071 | 00c |
| 060: | 047 | 058 | 07f | 1a4 | 134 | 129 | 084 | 15d | 19d | 1b2 | 1a3 | 048 | 07c | 051 | 1ca | 023 |
| 070: | 13d | 1a7 | 165 | 03b | 042 | 0da | 192 | 0ce | 0c1 | 06b | 09f | 1f1 | 12c | 184 | 0fa | 196 |
| 080: | 1e1 | 169 | 17d | 031 | 180 | 10a | 094 | 1da | 186 | 13e | 11c | 060 | 175 | 1cf | 067 | 119 |
| 090: | 065 | 068 | 099 | 150 | 008 | 007 | 17c | 0b7 | 024 | 019 | 0de | 127 | 0db | 0e4 | 1a9 | 052 |
| 0a0: | 109 | 090 | 19c | 1c1 | 028 | 1b3 | 135 | 16a | 176 | 0df | 1e5 | 188 | 0c5 | 16e | 1de | 1b1 |
| 0b0: | 0c3 | 1df | 036 | 0ee | 1ee | 0f0 | 093 | 049 | 09a | 1b6 | 069 | 081 | 125 | 00b | 05e | 0b4 |
| 0c0: | 149 | 1c7 | 174 | 03e | 13b | 1b7 | 08e | 1c6 | 0ae | 010 | 095 | 1ef | 04e | 0f2 | 1fd | 085 |
| 0d0: | 0fd | 0f6 | 0a0 | 16f | 083 | 08a | 156 | 09b | 13c | 107 | 167 | 098 | 1d0 | 1e9 | 003 | 1fe |
| 0e0: | 0bd | 122 | 089 | 0d2 | 18f | 012 | 033 | 06a | 142 | 0ed | 170 | 11b | 0e2 | 14f | 158 | 131 |
| 0f0: | 147 | 05d | 113 | 1cd | 079 | 161 | 1a5 | 179 | 09e | 1b4 | 0cc | 022 | 132 | 01a | 0e8 | 004 |
| 100: | 187 | 1ed | 197 | 039 | 1bf | 1d7 | 027 | 18b | 0c6 | 09c | 0d0 | 14e | 06c | 034 | 1f2 | 06e |
| 110: | 0ca | 025 | 0ba | 191 | 0fe | 013 | 106 | 02f | 1ad | 172 | 1db | 0c0 | 10b | 1d6 | 0f5 | 1ec |
| 120: | 10d | 076 | 114 | 1ab | 075 | 10c | 1e4 | 159 | 054 | 11f | 04b | 0c4 | 1be | 0f7 | 029 | 0a4 |
| 130: | 00e | 1f0 | 077 | 04d | 17a | 086 | 08b | 0b3 | 171 | 0bf | 10e | 104 | 097 | 15b | 160 | 168 |
| 140: | 0d7 | 0bb | 066 | 1ce | 0fc | 092 | 1c5 | 06f | 016 | 04a | 0a1 | 139 | 0af | 0f1 | 190 | 00a |
| 150: | 1aa | 143 | 17b | 056 | 18d | 166 | 0d4 | 1fb | 14d | 194 | 19a | 087 | 1f8 | 123 | 0a7 | 1b8 |
| 160: | 141 | 03c | 1f9 | 140 | 02a | 155 | 11a | 1a1 | 198 | 0d5 | 126 | 1af | 061 | 12e | 157 | 1dc |
| 170: | 072 | 18a | 0aa | 096 | 115 | 0ef | 045 | 07b | 08d | 145 | 053 | 05f | 178 | 0b2 | 02e | 020 |
| 180: | 1d5 | 03f | 1c9 | 1e7 | 1ac | 044 | 038 | 014 | 0b1 | 16b | 0ab | 0b5 | 05a | 182 | 1c8 | 1d4 |
| 190: | 018 | 177 | 064 | 0cf | 06d | 100 | 199 | 130 | 15a | 005 | 120 | 1bb | 1bd | 0e0 | 04f | 0d6 |
| 1a0: | 13f | 1c4 | 12a | 015 | 006 | 0ff | 19b | 0a6 | 043 | 088 | 050 | 15f | 1e8 | 121 | 073 | 17e |
| 1b0: | 0bc | 0c2 | 0c9 | 173 | 189 | 1f5 | 074 | 1cc | 1e6 | 1a8 | 195 | 01f | 041 | 00d | 1ba | 032 |
| 1c0: | 03d | 1d1 | 080 | 0a8 | 057 | 1b9 | 162 | 148 | 0d9 | 105 | 062 | 07a | 021 | 1ff | 112 | 108 |
| 1d0: | 1c0 | 0a9 | 11d | 1b0 | 1a6 | 0cd | 0f3 | 05c | 102 | 05b | 1d9 | 144 | 1f6 | 0ad | 0a5 | 03a |
| 1e0: | 1cb | 136 | 17f | 046 | 0e1 | 01e | 1dd | 0e6 | 137 | 1fa | 185 | 08c | 08f | 040 | 1b5 | 0be |
| 1f0: | 078 | 000 | 0ac | 110 | 15e | 124 | 002 | 1bc | 0a2 | 0ea | 070 | 1fc | 116 | 15c | 04c | 1c2 |

Function FL takes two parameters. One is a 32-bit data, namely FL_IN. The other is an index of EK, namely k. And FL returns a 32-bit width data, namely FL_OUT.

```
FL(FL_IN, k)
begin
  var d0, d1 as 16-bit integer;
  d0 = FL_IN >> 16;
  d1 = FL_IN & 0xffff;
  if (k is an even number) then
    d1 = d1 ^ (d0 & EK[k/2]);
    d0 = d0 ^ (d1 | EK[(k/2+6)%8+8]);
  else
    d1 = d1 ^ (d0 & EK[((k-1)/2+2)%8+8]);
    d0 = d0 ^ (d1 | EK[((k-1)/2+4)%8]);
  endif
  FL_OUT = (d0<<16) | d1;
  return FL_OUT;
end.
```

When the algorithm is used for decryption, function FLINV is used instead of function FL.

```
FLINV(FL_IN, k)
begin
  var d0, d1 as 16-bit integer;
  d0 = FL_IN >> 16;
  d1 = FL_IN & 0xffff;
  if (k is an even number) then
    d0 = d0 ^ (d1 | EK[(k/2+6)%8+8]);
    d1 = d1 ^ (d0 & EK[k/2]);
  else
    d0 = d0 ^ (d1 | EK[((k-1)/2+4)%8]);
    d1 = d1 ^ (d0 & EK[((k-1)/2+2)%8+8]);
  endif
  FL_OUT = (d0<<16) | d1;
  return FL_OUT;
end.
```

In most cases, data randomizing part consists of 8 "rounds". Round contains the call of function FO. Additionally, even-number round includes the calls of function FL. After the final round, FLs are called again. The detail description is as follows.

64-bit plaintext P is divided into the leftmost 32-bit D0 and the rightmost 32-bit D1.

```
// 0 round
D0 = FL(D0, 0);
D1 = FL(D1, 1);
D1 = D1 ^ FO(D0, 0);
// 1 round
D0 = D0 ^ FO(D1, 1);
// 2 round
D0 = FL(D0, 2);
D1 = FL(D1, 3);
D1 = D1 ^ FO(D0, 2);
// 3 round
D0 = D0 ^ FO(D1, 3);
// 4 round
D0 = FL(D0, 4);
D1 = FL(D1, 5);
D1 = D1 ^ FO(D0, 4);
// 5 round
D0 = D0 ^ FO(D1, 5);
// 6 round
D0 = FL(D0, 6);
D1 = FL(D1, 7);
D1 = D1 ^ FO(D0, 6);
// 7 round
D0 = D0 ^ FO(D1, 7);
// final
D0 = FL(D0, 8);
D1 = FL(D1, 9);
```

64-bit ciphertext C is constructed from D0 and D1 as following operation.

$$C = (D1 \ll 32) \mid D0;$$

When data randomizing part is used as decrypting operation, it should be executed in reverse order. The detail description is as follows.

```
D0 = C & 0xffffffff;
D1 = C >> 32;
D0 = FLINV(D0, 8);
D1 = FLINV(D1, 9);
D0 = D0 ^ FO(D1, 7);
D1 = D1 ^ FO(D0, 6);
D0 = FLINV(D0, 6);
D1 = FLINV(D1, 7);
D0 = D0 ^ FO(D1, 5);
D1 = D1 ^ FO(D0, 4);
D0 = FLINV(D0, 4);
D1 = FLINV(D1, 5);
```

```
D0 = D0 ^ FO(D1, 3);
D1 = D1 ^ FO(D0, 2);
D0 = FLINV(D0, 2);
D1 = FLINV(D1, 3);
D0 = D0 ^ FO(D1, 1);
D1 = D1 ^ FO(D0, 0);
D0 = FLINV(D0, 0);
D1 = FLINV(D1, 1);
P = (D0<<32) | D1;
```

3. Object Identifier

The Object Identifier for MISTY1 in Cipher Block Chaining (CBC) mode is as follows:

```
MISTY1-CBC OBJECT IDENTIFIER ::=
  {iso(1) member-body(2) jisc(392)
   mitsubishi-electric-corporation(200011) isl(61) security(1)
   algorithm(1) symmetric-encryption-algorithm(1) mistyl-cbc(1)}
```

MISTY1-CBC needs Initialization Vector (IV) as like as other algorithms, such as DES-CBC, DES-EDE3-CBC and so on. To determine the value of IV, MISTY1-CBC takes parameter as:

```
MISTY1-CBC Parameter ::= IV
```

where IV ::= OCTET STRING -- 8 octets.

When this Object Identifier is used, plaintext is padded before encrypt it. At least 1 padding octet is appended at the end of the plaintext to make the length of the plaintext to the multiple of 8 octets. The value of these octets is as same as the number of appended octets. (e.g., If 5 octets are needed to pad, the value is 0x05.)

4. Security Considerations

The algorithm, which is described in this document, is designed in consideration of the theory of provable security against differential cryptanalysis and linear cryptanalysis [2][3][4]. According to the recent result, when the algorithm consists of 8 rounds, both differential characteristic probability and liner characteristic probability are 2^{-140} . For reference, probabilities of DES are 2^{-62} and 2^{-46} , respectively.

5. Legal Issues

The algorithm description is applied for a patent in several countries as PCT/JP96/02154. However, the algorithm is freely available for academic (non-profit) use. Additionally, the algorithm can be used for commercial use without paying the patent fee if you contract with Mitsubishi Electric Corporation. For more information, please contact at MISTY@isl.melco.co.jp.

6. References

- [1] M. Matsui, "New Block Encryption Algorithm MISTY", Fast Software Encryption - 4th International Workshop (FSE'97), LNCS 1267, Springer Verlag, 1997, pp.54-68
- [2] K. Nyberg and L.R. Knudsen, "Provable Security Against a Differential Attack", Journal of Cryptology, Vol.8, No.1, 1995, pp. 27-37
- [3] K. Nyberg, "Linear Approximation of Block Ciphers", Advances in Cryptology - Eurocrypt'94, LNCS 950, Springer Verlag, 1995, pp.439-444
- [4] M. Matsui, "New Structure of Block Ciphers with Provable Security Against Differential and Linear Cryptanalysis", Fast Software Encryption - Third International Workshop, LNCS 1039, Springer Verlag, 1996, pp.205-218

7. Authors' Addresses

Hidenori Ohta
Mitsubishi Electric Corporation, Information Technology R&D Center
5-1-1 Ofuna, Kamakura, Kanagawa 247-8501, Japan

Phone: +81-467-41-2183
Fax: +81-467-41-2185
EMail: hidenori@iss.isl.melco.co.jp

Mitsuru Matsui
Mitsubishi Electric Corporation, Information Technology R&D Center
5-1-1 Ofuna, Kamakura, Kanagawa 247-8501, Japan

Phone: +81-467-41-2181
Fax: +81-467-41-2185
EMail: matsui@iss.isl.melco.co.jp

Appendix A. Example Data of MISTY1

Here is an example ciphertext of MISTY1 when the key and the plaintext are set as following value.

```
Key:          00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
Plaintext:    01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10
Ciphertext:   8b 1d a5 f5 6a b3 d0 7c 04 b6 82 40 b1 3b e9 5d
```

In the above example, because the plaintext has a length of 128-bit, MISTY1 is used two times to each 64-bit, namely ECB mode.

Following example is ciphertext of MISTY1 in CBC mode.

```
Key:          00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
IV:           01 02 03 04 05 06 07 08
Plaintext:    01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10
Ciphertext:   46 1c 1e 87 9c 18 c2 7f b9 ad f2 d8 0c 89 03 1f
```

Full Copyright Statement

Copyright (C) The Internet Society (2000). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.