Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms — Design and Analysis —

Kazumaro Aoki† Tetsuya Ichikawa‡ Masayuki Kanda† Mitsuru Matsui‡
Shiho Moriai† Junko Nakajima‡ Toshio Tokita‡

†Nippon Telegraph and Telephone Corporation
1-1 Hikarinooka, Yokosuka, Kanagawa, 239-0847 Japan
{maro,kanda,shiho}@isl.ntt.co.jp

‡Mitsubishi Electric Corporation
5-1-1 Ofuna, Kamakura, Kanagawa, 247-8501 Japan
{ichikawa,matsui,june15,tokita}@iss.isl.melco.co.jp

Abstract. We present a new 128-bit block cipher called *Camellia*. Camellia supports 128-bit block size and 128-, 192-, and 256-bit keys, i.e. the same interface specifications as the Advanced Encryption Standard (AES). Efficiency on both software and hardware platforms is a remarkable characteristic of Camellia in addition to its high level of security. It is confirmed that Camellia provides strong security against differential and linear cryptanalysis. Compared to the AES finalists, i.e. MARS, RC6, Rijndael, Serpent, and Twofish, Camellia offers at least comparable encryption speed in software and hardware. An optimized implementation of Camellia in assembly language can encrypt on a Pentium III (800MHz) at the rate of more than 276 Mbits per second, which is much faster than the speed of an optimized DES implementation. In addition, a distinguishing feature is its small hardware design. The hardware design, which includes both encryption and decryption, occupies approximately 11K gates, which is the smallest among all existing 128-bit block ciphers as far as we know.

1 Introduction

This paper presents a 128-bit block cipher called *Camellia*, which was jointly developed by NTT and Mitsubishi Electric Corporation. Camellia supports 128-bit block size and 128-, 192-, and 256-bit key lengths, and so offers the same interface specifications as the Advanced Encryption Standard (AES). The design goals of Camellia are as follows.

High level of security. The recent advances in cryptanalytic techniques are remarkable. A quantitative evaluation of security against powerful cryptanalytic techniques such as differential cryptanalysis [4] and linear cryptanalysis [21] is considered to be essential in designing any new block cipher. We evaluated the security of Camellia by utilizing state-of-art cryptanalytic techniques. We have confirmed that Camellia has no differential and linear characteristics that hold with probability more than 2^{-128}. Moreover, Camellia was designed to offer security against other advanced cryptanalytic attacks including higher order differential attacks [15, 12], interpolation attacks [12, 2], related-key attacks [5, 18], truncated differential attacks [15, 26], boomerang attacks [29], and slide attacks [6, 7].

Efficiency on multiple platforms. As cryptographic systems are needed in various applications, encryption algorithms that can be implemented efficiently on a wide range of platforms are desirable, however, few 128-bit block ciphers are suitable for both software and hardware implementation. Camellia was designed to offer excellent efficiency in hardware and software implementations, including gate count for hardware design, memory requirements in smart card implementations, as well as performance on multiple platforms.

Camellia consists of only 8-by-8-bit substitution tables (s-boxes) and logical operations that can be efficiently implemented on a wide variety of platforms. Therefore, it can be implemented efficiently in software, including the 8-bit processors used in low-end smart cards, 32-bit processors widely used in PCs, and 64-bit processors. Camellia doesn't use 32-bit integer additions and multiplications, which are extensively used in some software-oriented 128-bit block ciphers. Such operations perform well on platforms providing a high degree of support, e.g., Pentium II/III or Athlon, but not as well on others. These operations can cause a longer critical path and larger hardware implementation requirements.

The s-boxes of Camellia are designed to minimize hardware size. The four s-boxes are affine equivalent to the inversion function in the finite field GF(2^8). Moreover, we reduced the inversion function in GF(2^8) to a few GF(2^4) arithmetic operations. It enabled us to implement the s-boxes by fewer gate counts.

The key schedule is simple and shares part of its procedure with encryption. It supports on-the-key subkey generation and subkeys are computable in any order. The memory requirement for generating subkeys is quite small; an efficient implementation requires about 32-byte RAM for 128-bit keys and about 64-byte RAM for 192- and 256-bit keys.

Future developments. NTT and Mitsubishi Electric Corporation will propose Camellia in response to the call for contributions from ISO/IEC JTC 1/SC 27, aiming at its being adopted as an international standard. We will submit Camellia to NESSIE (New European Schemes for Signature, Integrity, and Encryption) project as a strong cryptographic primitive.

Outline of the paper. This paper is organized as follows: Section 2 describes the notations and high-level structure of Camellia. Section 3 defines each components of the cipher. Section 4 describes the rationale behind Camellia's design. In Section 5 we evaluate Camellia's strength against known attacks. Section 6 contains the performance of Camellia. We conclude in Section 7.

2 Structure of *Camellia*

Camellia uses an 18-round Feistel structure for 128-bit keys, and a 24-round Feistel structure for 192- and 256-bit keys, with additional input/output whitenings and logical functions called the FL-function and FL^{-1}-function inserted every 6 rounds. Figures 1 shows an overview of encryption using 128-bit keys. The key schedule generates 64-bit subkeys kw_t $(t = 1, 2, 3, 4)$ for input/output whitenings, k_u $(u = 1, 2, \ldots, r)$ for round functions and $kl_v (v = 1, 2, \ldots, r/3 - 2)$ for FL- and FL^{-1}-functions from the secret key K.

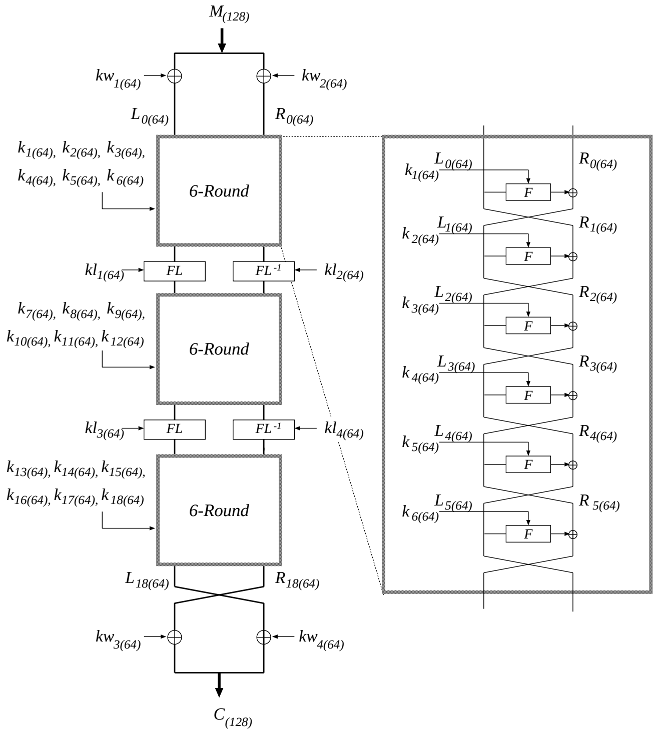

Figure 1: Encryption procedure of Camellia for 128-bit keys

2.1 Notations

X_L	the left-half data of X.
X_R	the right-half data of X.
$\oplus$	bitwise exclusive-OR operation.
$\|\|$	concatenation of two operands.
$\ggg_n$	rotation to the right by n bits.
$\lll_n$	rotation to the left by n bits.
$\cap$	bitwise AND operation.
$\cup$	bitwise OR operation.
`0x`	hexadecimal representation.

2.2 Encryption for 128-bit keys

First a 128-bit plaintext M is XORed with $kw_1 || kw_2$ and separated into two 64-bit data L_0 and R_0, i.e., $M \oplus (kw_1 || kw_2) = L_0 || R_0$. Then, the following operations are performed from $r = 1$ to 18, except for $r = 6$ and 12;

$$
\begin{aligned}
L_r &= R_{r-1} \oplus F(L_{r-1}, k_r), \\
R_r &= L_{r-1}.
\end{aligned}
$$

For $r = 6$ and 12, the following is carried out;

$$
L'_r = R_{r-1} \oplus F(L_{r-1}, k_r),
$$

$$\begin{aligned} R'_r &= L_{r-1}, \\ L_r &= FL(L'_r, kl_{r/3-1}), \\ R_r &= FL^{-1}(R'_r, kl_{r/3}). \end{aligned}$$

Lastly, R_{18} and L_{18} are concatenated and XORed with $kw_3||kw_4$. The resultant value is the 128-bit ciphertext, i.e., $C = (R_{18}||L_{18}) \oplus (kw_3||kw_4)$.

2.3 Encryption for 192- and 256-bit keys

Similarly to the encryption for 128-bit keys, first a 128-bit plaintext M is XORed with $kw_1||kw_2$ and separated into two 64-bit data L_0 and R_0, i.e., $M \oplus (kw_1||kw_2) = L_0||R_0$. Then, the following operations are performed from $r = 1$ to 24, except for $r = 6$, 12, and 18;

$$\begin{aligned} L_r &= R_{r-1} \oplus F(L_{r-1}, k_r), \\ R_r &= L_{r-1}. \end{aligned}$$

For $r = 6$, 12, and 18, the following are performed;

$$\begin{aligned} L'_r &= R_{r-1} \oplus F(L_{r-1}, k_r), \\ R'_r &= L_{r-1}, \\ L_r &= FL(L'_r, kl_{2r/6-1}), \\ R_r &= FL^{-1}(R'_r, kl_{2r/6}). \end{aligned}$$

Lastly, R_{24} and L_{24} are concatenated and XORed with $kw_3||kw_4$. The resultant value is the 128-bit ciphertext, i.e., $C = (R_{24}||L_{24}) \oplus (kw_3||kw_4)$.

2.4 Decryption

The decryption procedure of *Camellia* can be done in the same way as the encryption procedure by reversing the order of the subkeys, which is one of merits of Feistel networks. In *Camellia*, FL/FL^{-1}-function layers are inserted every 6 rounds, but this property is still preserved.

2.5 Key Schedule

Figure 2 shows the key schedule of Camellia. Two 128-bit variables K_L and K_R are defined as follows. For 128-bit keys, the 128-bit key K is used as K_L and K_R is 0. For 192-bit keys, the left 128-bit of the key K is used as K_L, and concatenation of the right 64-bit of K and the complement of the right 64-bit of K is used as K_R. For 256-bit keys, the left 128-bit of the key K is used as K_L and the right 128-bit of K is used as K_R.

Two 128-bit variables K_A and K_B are generated from K_L and K_R as shown in Figure 2. Note that K_B is used only if the length of the secret key is 192 or 256 bits. The 64-bit constants Σ_i $(i = 1, 2, \ldots, 6)$ are used as "keys" in the Feistel network. They are defined as continuous values from the second hexadecimal place to the seventeenth hexadecimal place of the hexadecimal representation of the square root of the i-th prime. These constant values are shown in Table 1.

Table 1: The key schedule constants

Σ_1	0xA09E667F3BCC908B
Σ_2	0xB67AE8584CAA73B2
Σ_3	0xC6EF372FE94F82BE
Σ_4	0x54FF53A5F1D36F1C
Σ_5	0x10E527FADE682D1D
Σ_6	0xB05688C2B3E6C1FD

The 64-bit subkeys kw_t, k_u, and kl_v are generated from K_L, K_R, K_A, and K_B. The subkeys are generated by rotating K_L, K_R, K_A, and K_B and taking the left- or right-half of them. Details are shown in Tables 2 and 3.

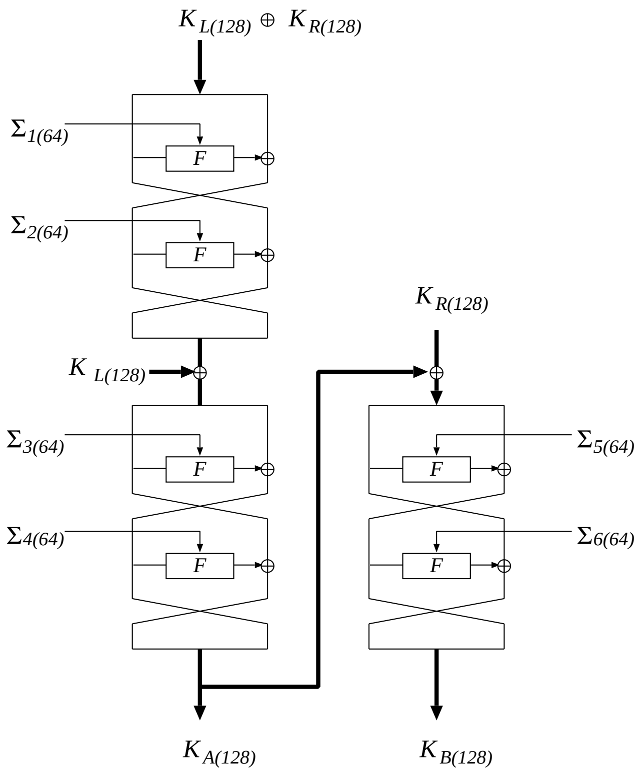

Figure 2: Key Schedule

Table 2: Subkeys for 128-bit keys

	subkey	value
Prewhitening	kw_1	$(K_L \lll_0)_L$
	kw_2	$(K_L \lll_0)_R$
F (Round1)	k_1	$(K_A \lll_0)_L$
F (Round2)	k_2	$(K_A \lll_0)_R$
F (Round3)	k_3	$(K_L \lll_{15})_L$
F (Round4)	k_4	$(K_L \lll_{15})_R$
F (Round5)	k_5	$(K_A \lll_{15})_L$
F (Round6)	k_6	$(K_A \lll_{15})_R$
FL	kl_1	$(K_A \lll_{30})_L$
FL^{-1}	kl_2	$(K_A \lll_{30})_R$
F (Round7)	k_7	$(K_L \lll_{45})_L$
F (Round8)	k_8	$(K_L \lll_{45})_R$
F (Round9)	k_9	$(K_A \lll_{45})_L$
F (Round10)	k_{10}	$(K_L \lll_{60})_R$
F (Round11)	k_{11}	$(K_A \lll_{60})_L$
F (Round12)	k_{12}	$(K_A \lll_{60})_R$
FL	kl_3	$(K_L \lll_{77})_L$
FL^{-1}	kl_4	$(K_L \lll_{77})_R$
F (Round13)	k_{13}	$(K_L \lll_{94})_L$
F (Round14)	k_{14}	$(K_L \lll_{94})_R$
F (Round15)	k_{15}	$(K_A \lll_{94})_L$
F (Round16)	k_{16}	$(K_A \lll_{94})_R$
F (Round17)	k_{17}	$(K_L \lll_{111})_L$
F (Round18)	k_{18}	$(K_L \lll_{111})_R$
Postwhitening	kw_3	$(K_A \lll_{111})_L$
	kw_4	$(K_A \lll_{111})_R$

Table 3: Subkeys for 192/256-bit keys

	subkey	value
Prewhitening	kw_1	$(K_L \lll_0)_L$
	kw_2	$(K_L \lll_0)_R$
F (Round1)	k_1	$(K_B \lll_0)_L$
F (Round2)	k_2	$(K_B \lll_0)_R$
F (Round3)	k_3	$(K_R \lll_{15})_L$
F (Round4)	k_4	$(K_R \lll_{15})_R$
F (Round5)	k_5	$(K_A \lll_{15})_L$
F (Round6)	k_6	$(K_A \lll_{15})_R$
FL	kl_1	$(K_R \lll_{30})_L$
FL^{-1}	kl_2	$(K_R \lll_{30})_R$
F (Round7)	k_7	$(K_B \lll_{30})_L$
F (Round8)	k_8	$(K_B \lll_{30})_R$
F (Round9)	k_9	$(K_L \lll_{45})_L$
F (Round10)	k_{10}	$(K_L \lll_{45})_R$
F (Round11)	k_{11}	$(K_A \lll_{45})_L$
F (Round12)	k_{12}	$(K_A \lll_{45})_R$
FL	kl_3	$(K_L \lll_{60})_L$
FL^{-1}	kl_4	$(K_L \lll_{60})_R$
F (Round13)	k_{13}	$(K_R \lll_{60})_L$
F (Round14)	k_{14}	$(K_R \lll_{60})_R$
F (Round15)	k_{15}	$(K_B \lll_{60})_L$
F (Round16)	k_{16}	$(K_B \lll_{60})_R$
F (Round17)	k_{17}	$(K_L \lll_{77})_L$
F (Round18)	k_{18}	$(K_L \lll_{77})_R$
FL	kl_5	$(K_A \lll_{77})_L$
FL^{-1}	kl_6	$(K_A \lll_{77})_R$
F (Round19)	k_{19}	$(K_R \lll_{94})_L$
F (Round20)	k_{20}	$(K_R \lll_{94})_R$
F (Round21)	k_{21}	$(K_A \lll_{94})_L$
F (Round22)	k_{22}	$(K_A \lll_{94})_R$
F (Round23)	k_{23}	$(K_L \lll_{111})_L$
F (Round24)	k_{24}	$(K_L \lll_{111})_R$
Postwhitening	kw_3	$(K_B \lll_{111})_L$
	kw_4	$(K_B \lll_{111})_R$

3 Components of *Camellia*

3.1 F-function

The F-function is shown in Figure 3. The F-function uses the SPN (Substitution-Permutation Network) structure. The S-function is the non-linear layer and the P-function is the linear layer.

3.2 S-function, s-boxes

The S-function consists of eight s-boxes and four different s-boxes, s_1, s_2, s_3, and s_4 are used. All of them are affine equivalent to the inversion function in GF(2^8). The data of s_2, s_3, and s_4 can be generated from the s_1 table. The tables are shown in [1].

$$\begin{aligned}
s_1 &: \mathrm{GF}(2)^8 \to \mathrm{GF}(2)^8, \\
&\quad x \mapsto \mathbf{h}(\mathbf{g}(\mathbf{f}(\texttt{0xc5} \oplus x))) \oplus \texttt{0x6e} \\
s_2 &: \mathrm{GF}(2)^8 \to \mathrm{GF}(2)^8,\ x \mapsto s_1(x) \lll_1 \\
s_3 &: \mathrm{GF}(2)^8 \to \mathrm{GF}(2)^8,\ x \mapsto s_1(x) \ggg_1 \\
s_4 &: \mathrm{GF}(2)^8 \to \mathrm{GF}(2)^8,\ x \mapsto s_1(x \lll_1)
\end{aligned}$$

where functions $\mathbf{f}$ and $\mathbf{h}$ are affine functions and function $\mathbf{g}$ is the inversion function in GF(2^8) as given below.

$$\mathbf{f} : \mathrm{GF}(2)^8 \to \mathrm{GF}(2)^8,\ (a_1, a_2, \ldots, a_8) \mapsto (b_1, b_2, \ldots, b_8),$$

where

$$\begin{aligned}
b_1 &= a_6 \oplus a_2 & b_5 &= a_7 \oplus a_4 \\
b_2 &= a_7 \oplus a_1 & b_6 &= a_5 \oplus a_2 \\
b_3 &= a_8 \oplus a_5 \oplus a_3 & b_7 &= a_8 \oplus a_1 \\
b_4 &= a_8 \oplus a_3 & b_8 &= a_6 \oplus a_4.
\end{aligned}$$

$$\mathbf{h} : \mathrm{GF}(2)^8 \to \mathrm{GF}(2)^8,\ (a_1, a_2, \ldots, a_8) \mapsto (b_1, b_2, \ldots, b_8),$$

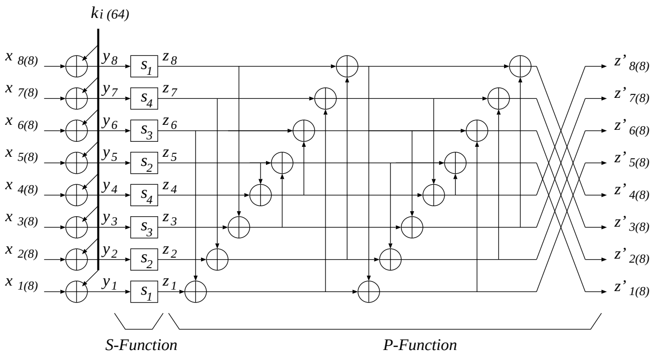

Figure 3: F-function

where

$$\begin{array}{ll} b_1 = a_5 \oplus a_6 \oplus a_2 & b_5 = a_7 \oplus a_3 \\ b_2 = a_6 \oplus a_2 & b_6 = a_8 \oplus a_1 \\ b_3 = a_7 \oplus a_4 & b_7 = a_5 \oplus a_1 \\ b_4 = a_8 \oplus a_2 & b_8 = a_6 \oplus a_3. \end{array}$$

$$\mathbf{g} : \mathrm{GF}(2)^8 \to \mathrm{GF}(2)^8,\ (a_1, a_2, \ldots, a_8) \mapsto (b_1, b_2, \ldots, b_8),$$

where

$$\begin{array}{ll} & (b_8 + b_7\alpha + b_6\alpha^2 + b_5\alpha^3) + (b_4 + b_3\alpha + b_2\alpha^2 + b_1\alpha^3)\beta \\ = & ((a_8{+}a_7\alpha{+}a_6\alpha^2{+}a_5\alpha^3){+}(a_4{+}a_3\alpha{+}a_2\alpha^2{+}a_1\alpha^3)\beta)^{-1}. \end{array}$$

This inversion is performed in $\mathrm{GF}(2^8)$ assuming $\frac{1}{0} = 0$, where β is an element in $\mathrm{GF}(2^8)$ that satisfies $\beta^8 + \beta^6 + \beta^5 + \beta^3 + 1 = 0$, and $\alpha = \beta^{238} = \beta^6 + \beta^5 + \beta^3 + \beta^2$ is an element in $\mathrm{GF}(2^4)$ that satisfies $\alpha^4 + \alpha + 1 = 0$.

3.3 $\boldsymbol{P}$-function

The P-function is defined as follows:

$$\begin{array}{ll} P: & (\mathrm{GF}(2)^8)^8 \to (\mathrm{GF}(2)^8)^8, \\ & (z_1, z_2, \ldots, z_8) \mapsto (z'_1, z'_2, \ldots, z'_8), \end{array}$$

where

$$\begin{aligned} z'_1 &= z_1 \oplus z_3 \oplus z_4 \oplus z_6 \oplus z_7 \oplus z_8 \\ z'_2 &= z_1 \oplus z_2 \oplus z_4 \oplus z_5 \oplus z_7 \oplus z_8 \\ z'_3 &= z_1 \oplus z_2 \oplus z_3 \oplus z_5 \oplus z_6 \oplus z_8 \\ z'_4 &= z_2 \oplus z_3 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7 \\ z'_5 &= z_1 \oplus z_2 \oplus z_6 \oplus z_7 \oplus z_8 \\ z'_6 &= z_2 \oplus z_3 \oplus z_5 \oplus z_7 \oplus z_8 \\ z'_7 &= z_3 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_8 \\ z'_8 &= z_1 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7. \end{aligned}$$

3.4 $\boldsymbol{FL}$-function

The FL-function is shown in Figure 4, and is defined as follows.

$$\begin{array}{rcl} FL : \mathrm{GF}(2)^{64} \times \mathrm{GF}(2)^{64} & \to & \mathrm{GF}(2)^{64}, \\ (X_L \| X_R,\ kl_L \| kl_R) & \mapsto & Y_L \| Y_R, \end{array}$$

where

$$\begin{array}{rcl} Y_R & = & ((X_L \cap kl_L) \lll_1) \oplus X_R \\ Y_L & = & (Y_R \cup kl_R) \oplus X_L. \end{array}$$

3.5 $\boldsymbol{FL^{-1}}$-function

The FL^{-1}-function is shown in Figure 5. The following equation holds.

$$FL^{-1}(FL(x, k), k) = x.$$

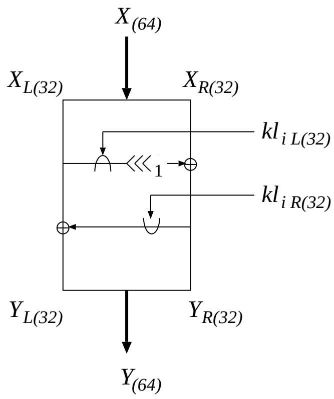

Figure 4: FL-function

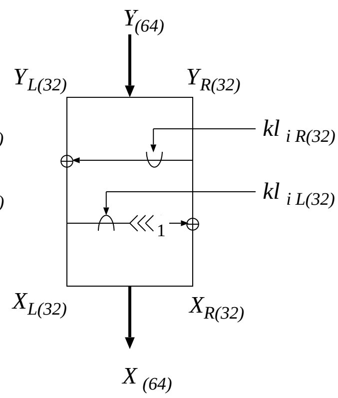

Figure 5: FL^{-1}-function

4 Design Rationale

4.1 $\boldsymbol{F}$-function

The design strategy of the F-function of Camellia follows that of the F-function of E2 [28]. The main difference between E2 and Camellia is the adoption of the 1-round (conservative) SPN (Substitution-Permutation Network), not the 2-round SPN, i.e. S-P-S. When the 1-round SPN is used as the round function in a Feistel cipher, the theoretical evaluation of the upper bound of differential and linear characteristic probability becomes more complicated, but the speed under the same level of "real" security is expected to be improved. See Section 6 for detailed discussions on security.

4.2 *P*-function

The design rationale of the P-function is similar to that of the P-function of E2. That is, for computational efficiency, it should be represented using only bytewise exclusive-ORs and for security against differential and linear cryptanalysis, its branch number should be optimal [19]. From among the linear transformations that satisfy these conditions, we chose one considering highly efficient implementation on 32-processors [3] and high-end smart cards, as well as 8-bit processors.

4.3 *s*-boxes

As the s-boxes we adopted functions affine equivalent to the inversion function in GF(2^8) for enhanced security and small hardware design.

It is well known that the smallest of the maximum differential probability of functions in GF(2^8) was proven to be 2^{-6}, and the smallest of the maximum linear probability of functions in GF(2^8) is conjectured to be 2^{-6}. There is a function affine equivalent to the inversion function in GF(2^8) that achieves the best known of the maximum differential and linear probabilities, 2^{-6}. We choose this kind of functions as s-boxes. Moreover, the high degree of the Boolean polynomial of every output bit of the s-boxes makes it difficult to attack Camellia by higher order differential attacks. The two affine functions that are performed at the input and output of the inversion function in GF(2^8) complicates the expressions of the s-boxes in GF(2^8), which is expected to make interpolation attacks ineffective. Making the four s-boxes different slightly improves security against truncated differential cryptanalysis [26].

For small hardware design, the elements in GF(2^8) can be represented as polynomials with coefficients in the subfield GF(2^4). In other words, we can implement the s-boxes by using a few operations in the subfield GF(2^4) [25]. Two affine functions at the input and output of the inversion function in GF(2^8) also play a role in complicating the expressions of the s-boxes in GF(2^4).

4.4 *FL*- and *FL*$^{-1}$-functions

FL- and FL^{-1}-functions are "inserted" between every 6 rounds of a Feistel network to provide non-regularity across rounds. One of the goals for such a design is to thwart future unknown attacks. It is one of merits of regular Feistel networks that encryption and decryption procedures are the same except for the order of the subkeys. In Camellia, FL/FL^{-1}-function layers are inserted every 6 rounds, but this property is still preserved.

The design criteria of FL- and FL^{-1}-functions are similar to those of the FL-function of MISTY [23]. The difference between MISTY and Camellia is the addition of 1-bit rotation. This is expected to make bytewise cryptanalysis harder, but it has no negative impact on hardware size or speed. The design criteria are that these functions must be linear for any fixed key and that their forms depend on key values. Since these functions are linear as long as the key is fixed, they do not make the average differential and linear probabilities of the cipher higher. Moreover, these functions are fast in both software and hardware since they are constructed by logical operations such as AND, OR, XOR, and rotations.

4.5 Key Schedule

The design criteria of the key schedule are as follows.

1. It should be simple and share part of its procedure with encryption/decryption.

2. Subkey generation for 128-, 192- and 256-bit keys can be performed by using the same key schedule (circuit). Moreover, the key schedule for 128-bit keys can be performed by using a part of this circuit.

3. Key setup time should be shorter than encryption time. In cases where large amounts of data are processed with a single secret key, the setup time for key scheduling may be unimportant. On the other hand, in applications in which the key is changed frequently, key agility is a factor. One basic component of key agility is key setup time.

4. It should support on-the-fly subkey generation.

5. On-the-fly subkey generation should be computable in the same way in both encryption and decryption. Some ciphers have separate key schedules for encryption and decryption. In other ciphers, e.g., Rijndael or Serpent, subkeys are computable in the forward direction only and require unwinding for decryption.

6. There should be no equivalent keys.

7. There should be no related-key attacks or slide attacks.

Criteria 1 and 2 mainly address small hardware requirements, Criteria 3, 4, and 5 are advantageous in terms of practical applications, and Criteria 6 and 7 are for security.

The memory requirement for generating subkeys is quite small. An efficient implementation of Camellia for 128-bit keys requires 16 bytes (=128 bits) for the original secret key, K_L, and 16 bytes (=128 bits) for the intermediate key, K_A. Thus the required memory is 32 bytes. Similarly, an efficient implementation of Camellia for 192- and 256-bit keys needs only 64 bytes.

5 Security

5.1 Differential and Linear Cryptanalysis

The most well-known and powerful approaches to attacking many block ciphers are differential cryptanalysis, proposed by Biham and Shamir [4], and linear cryptanalysis, introduced by Matsui [21]. There are several methods of evaluating security against these attacks, where there is a kind of "duality" relation between them [22, 9]: in other words, the security against both attacks can be evaluated in similar ways.

It is known that the upper bounds of differential/linear characteristic probabilities can, for several block ciphers, be estimated using the minimum numbers of differential/linear active s-boxes in some consecutive rounds. Kanda [13] shows the minimum numbers of differential/linear active s-boxes for Feistel ciphers with conservative SPN (S-P) round function. Hereafter, we assume that linear transformation P is bijective.

Definition 1 *The branch number $\mathcal{B}$ of linear transformation P is defined by*

$$\mathcal{B} = \min_{x \neq 0}(w_{\mathrm{H}}(x) + w_{\mathrm{H}}(P(x))),$$

where $w_{\mathrm{H}}(x)$ denotes the bytewise Hamming weight of x.

Definition 2 *A differential active s-box is defined as an s-box given a non-zero input difference. A linear active s-box is defined as an s-box given a non-zero output mask value.*

Theorem 1 *The minimum number of differential/ linear active s-boxes in any eight consecutive rounds is equal or larger than $2\mathcal{B} + 1$.*

Definition 3 *Let p_s and q_s be the maximum differential/linear probabilities of all s-boxes $\{s_1, s_2, \ldots\}$.*

$$\begin{aligned} p_s &= \max_i \max_{\Delta x \neq 0, \Delta y} \Pr_x[s_i(x) \oplus s_i(x \oplus \Delta x) = \Delta y] \\ q_s &= \max_i \max_{\Gamma y \neq 0, \Gamma x} (2 \Pr_x[x \cdot \Gamma x = s_i(x) \cdot \Gamma y] - 1)^2 \end{aligned}$$

Theorem 2 *Let $\mathcal{D}$ and $\mathcal{L}$ be the minimum numbers of total differential/linear active s-boxes. Then, the maximum differential/linear characteristic probabilities are bounded by $p_s^{\mathcal{D}}$ and $q_s^{\mathcal{L}}$, respectively.*

With the above-mentioned techniques, we prove that Camellia offers immunity to these attacks by showing the upper bounds of maximum differential/linear characteristic probabilities, since Camellia is a Feistel cipher whose round function uses the S-P round function.

In the case of Camellia, the maximum differential/linear probabilities of the s-boxes are $p_s = q_s = 2^{-6}$. The branch number of the linear transformation (P-function) is 5, i.e. $\mathcal{B} = 5$. Letting p, q be the maximum differential/linear characteristic probabilities of Camellia reduced to 16-round without FL- and FL^{-1}-functions, respectively, we have $p \le p_s^{2(2\mathcal{B}+1)} = (2^{-6})^{22} = 2^{-132}$ and $q \le q_s^{2(2\mathcal{B}+1)} = (2^{-6})^{22} = 2^{-132}$ from Theorems 1 and 2. Both probabilities are below the security threshold of 128-bit block ciphers: 2^{-128}. It follows that there is no effective differential characteristic or linear characteristic for Camellia reduced to more than 15 rounds without FL- and FL^{-1}-functions. Since FL- and FL^{-1}-functions are linear for any fixed key, they do not make the average differential/linear probabilities of the cipher higher. Hence, it is proven that Camellia offers enough security against differential and linear attacks.

Note that the result above are based on Theorems 1 and 2. Both theorems deal with general cases of Feistel ciphers with SPN round function, so we expect that Camellia is actually more secure than shown by the result above. As supporting evidence, we counted the number of active s-boxes of Camellia with reduced rounds. The counting algorithm is similar to that described in [24] except following three items.

- Prepare the table for the number of active s-boxes instead of transition probability table.

- Count the number of active s-boxes instead of computing transition probability.

- FL- and FL^{-1}-functions set all elements to the minimum number of active s-boxes in the table. This means that the algorithm gives consideration to existence of weak subkeys inserted to FL- and FL^{-1}-functions, since there may be some possibility of connecting every later differential/linear characteristic with the previous one with the highest probability, which is equivalent to the minimum number of active s-boxes.

As a result, we confirmed that 12-round Camellia with FL- and FL-functions has no differential/linear characteristic with probability higher than 2^{-128} (see Tables 4 and 5).

Table 4: Upper bounds of differential characteristic probability of Camellia

# of rounds	1	2	3	4	5	6	7	8	9	10	11	12
Estimation based on Th. 1 and 2			2^{-12} (2)	2^{-30} (5)		2^{-42} (7)		2^{-66} (11)				2^{-96} (16)
Camellia	1 (0)	2^{-6} (1)	2^{-12} (2)	2^{-42} (7)	2^{-54} (9)	2^{-66} (11)	2^{-72} (12)	2^{-72} (12)	2^{-78} (13)	2^{-108} (18)	2^{-120} (20)	2^{-132} (22)
without FL/FL^{-1}-functions	1 (0)	2^{-6} (1)	2^{-12} (2)	2^{-36} (6)	2^{-54} (9)	2^{-66} (11)	2^{-78} (13)	2^{-90} (15)	2^{-108} (18)	2^{-126} (21)	2^{-132} (22)	

Note: The numbers in brackets are the number of active s-boxes.

Table 5: Upper bounds of linear characteristic probability of Camellia

# of rounds	1	2	3	4	5	6	7	8	9	10	11	12
Estimation based on Th. 1 and 2			2^{-12} (2)	2^{-30} (5)		2^{-42} (7)		2^{-66} (11)				2^{-96} (16)
Camellia	1 (0)	2^{-6} (1)	2^{-12} (2)	2^{-36} (6)	2^{-54} (9)	2^{-66} (11)	2^{-72} (12)	2^{-72} (12)	2^{-78} (13)	2^{-102} (17)	2^{-120} (20)	2^{-132} (22)
without FL/FL^{-1}-functions	1 (0)	2^{-6} (1)	2^{-12} (2)	2^{-36} (6)	2^{-54} (9)	2^{-66} (11)	2^{-78} (13)	2^{-84} (14)	2^{-108} (18)	2^{-120} (20)	2^{-132} (22)	

Note: The numbers in brackets are the number of active s-boxes.

5.2 Truncated Differential Cryptanalysis

The attacks using truncated differentials were introduced by Knudsen [15]. He defined them as differentials where only a part of the difference can be predicted. The notion of truncated differentials introduced by him is wide, but with a byte-oriented cipher it is natural to study bytewise differentials as truncated differentials [26].

The maximum differential probability is considered to provide the strict evaluation of security against differential cryptanalysis, but computing its value is impossible in general, since a differential is a set of all differential characteristics with the same input difference and the same output difference for a Markov cipher [20]. On the other hand, a truncated differential can be regarded as a subset of the differential characteristics which are exploitable in cryptanalysis. For some ciphers, e.g., byte-oriented ciphers, the probability of truncated differential can be computed easily and correctly, and it gives a more strict evaluation than the maximum differential characteristic probability.

A truncated differential cryptanalysis of reduced-round variants of E2 was presented by Matsui and Tokita at FSE'99 [26]. Their analysis was based on the "byte characteristic," where the values to the difference in a byte are distinguished between non-zero and zero. They found a 7-round byte characteristic, which leads to a possible attack on an 8-round variant of E2 without IT-Function (the initial transformation) and FT-Function (the final transformation). The best attack of E2 shown in [27] breaks an 8-round variant of E2 with either IT-Function or FT-Function using 2^{94} chosen plaintexts. In [27] we also show the attack which distinguishes a 7-round variant of E2 with IT- and FT-Functions from a random permutation using 2^{91} chosen plaintexts.

Camellia is a byte-oriented cipher similar to E2, and it is important to evaluate its security against truncated differential cryptanalysis. We searched for truncated differentials using an algorithm similar to the one described in [26, 27]. The main difference of the round function between E2 and Camellia is the adoption of the 1-round SPN not the 2-round SPN, i.e. S-P-S. In the search for truncated differentials of E2, we used about 2^{-8} as the probability of difference cancellation in byte at the XOR of Feistel network. However, the round function of Camellia doesn't have the second s-boxes-layer, and the difference cancellation sometimes occurs with probability 1. As a result, Camellia with more than 10 rounds is indistinguishable from a random permutation, in both cases with/without FL-/FL^{-1}-function layers.

5.3 Truncated Linear Cryptanalysis

We introduce a new cryptanalysis called truncated linear cryptanalysis.

Due to the duality between differential and linear cryptanalysis, we can evaluate security against truncated linear cryptanalysis by using a similar algorithm to that above. To put it concretely, we can perform the search by replacing the matrix of P-function with the transposed matrix. As a result, Camellia with more than 10 rounds without FL-/FL^{-1}-function layers is indistinguishable from a random permutation.

5.4 Cryptanalysis with Impossible Differential

The impossible differential means the differential which holds with probability 0, or the differential which never exists. Using such an impossible differential, it is possible to narrow down the candidates of the subkey. It is known that there is at least one 5-round impossible differential in any Feistel network with a bijective round function. Since Camellia has the Feistel network (with FL- and FL^{-1}-functions inserted between every 6 rounds) and the round function is bijective, Camellia has 5-round impossible differentials. We have not found impossible differentials with more than 6 rounds. Moreover, we expect FL- and FL^{-1}-functions make attacking Camellia using impossible differentials difficult, since the functions change differential paths depending on key values. In consequent, Camellia with full rounds will not be broken by cryptanalysis using impossible differentials.

5.5 Boomerang Attack

Boomerang attack [29] requires 2 differentials. Let the probability of the differentials be $p_\triangle$ and p_∇. An boomerang attack that is superior than exhaustive key search requires

$$p_\triangle p_\nabla \geq 2^{-64}. \tag{1}$$

Using Table 4, there is no combination that satisfies Inequality (1) for Camellia without FL- and FL^{-1}-functions. The best boomerang probability for Camellia without FL- and FL^{-1}-functions reduced to 8-round is bounded by 2^{-66} that is obtained by $p_\triangle = 2^{-12}$ (3 rounds) and $p_\nabla = 2^{-54}$ (5 rounds). Since attackable rounds for Camellia without FL- and FL^{-1}-functions is bounded by much shorter than the specification of Camellia, 18 or 24, Camellia seems secure against a boomerang attack.

5.6 Higher Order Differential Attack

Higher order differential attack is generally applicable to ciphers that can be represented as Boolean polynomials of low degree. All intermediate bits in the encryption process can be represented as Boolean polynomials, i.e. polynomials $\mathrm{GF}(2)[x_1, x_2, \ldots, x_n]$ in the bits of the plaintext: $\{x_1, x_2, \ldots, x_n\}$. In the higher order differential attack described in [12, Theorem 1], if the intermediate bits are represented by Boolean polynomials of degree at least d, the $(d+1)$-th order differential of the Boolean polynomial becomes 0.

For the degrees of Boolean polynomials of the s-boxes of Camellia, the functions affine equivalent to the inversion function in $\mathrm{GF}(2^8)$ are adopted as the s-boxes. It is known that the degree of the Boolean polynomial of every output bit of the inversion function in $\mathrm{GF}(2^8)$ is 7, but the degree for the s-boxes of Camellia is not trivial, since affine functions are added at the input and output. We confirmed that the degree of the Boolean polynomial of every output bit of the s-boxes is 7 by finding Boolean polynomial for every output bit of the s-boxes. In Camellia, it is expected that the degree of an intermediate bit in the encryption process increases as the data pass through many s-boxes. For example, the degree becomes $7^3 > 128$ after passing through three s-boxes. Therefore, we expect that higher order differential attacks fail against Camellia with full rounds.

Table 6: Smallest number of unknown coefficients for 128-, 192-, and 256-bit keys

whitening×1 + round×r ($r < 4$)	1
whitening×1 + round×4	255
More rounds	256

5.7 Interpolation Attack and Linear Sum Attack

The interpolation attack proposed in [12] is typically applicable to attacking ciphers that use simple algebraic functions.

The principle of interpolation attack is that, roughly speaking, if the ciphertext is represented as a polynomial or rational expression of the plaintext whose number of unknown coefficients is N, the polynomial or rational expression can be constructed using N pairs of plaintexts and ciphertexts. Once the attacker constructs the polynomial or rational expression, he can encrypt any plaintext into the corresponding ciphertext or decrypt any ciphertext into the corresponding plaintext for the key without knowing the key. Since N determines the complexity and the number of pairs required for the attack, it is important to make N as large as possible. If N is so large that it is impractical for the attackers to gather N plaintext-ciphertext pairs, the cipher is secure against interpolation attack.

Linear sum attack [2] is a generalization of the interpolation attack [12]. A practical algorithm that evaluates the security against linear sum attack was proposed in [2]. We searched for linear relations between any plaintext byte and any ciphertext byte over $\mathrm{GF}(2^8)$ using the algorithm. Table 6 summarizes the results.

Table 6 shows that Camellia is secure against linear sum attack including interpolation attack. It also implies that Camellia is secure against SQUARE attack [10] followed by [2, Theorem 3].

5.8 No Equivalent Keys

Since the set of subkeys generated by the key schedule contain the original secret key, there is no equivalent set of subkeys generated from distinct secret keys. Therefore, we expect that there are no distinct secret keys both of which encrypt each of many plaintexts into the same ciphertext.

5.9 Slide Attack

In [6, 7] the slide attacks were introduced, based on earlier work in [5, 14]. In particular it was shown that iterated ciphers with identical round functions, that is, equal structures and equal subkeys in the round functions, are susceptible to slide attacks.

In Camellia, FL- and FL^{-1}-functions are "inserted" between every 6 rounds of a Feistel network to provide non-regularity across rounds. Moreover, from the viewpoint of the key schedule, slide attacks seems to be very unlikely to succeed (See Section 5.10).

5.10 Related-key Attack

We are convinced that the key schedule of Camellia makes related-key attacks [5, 18] very difficult. In these attacks, an attacker must be able to get encryptions using several related keys. If the relation between, say, two keys, is known then if the corresponding relations between the subkeys can be predetermined, it might become possible to predict how the keys would encrypt a pair of different plaintexts. However, since the subkeys depend on K_A and K_B, which are the results of encryption of a secret key, and if an attacker wants to change the secret key, he can't get K_A and K_B desired, and vice versa, these subkey relations will be very hard to control and predict.

5.11 Implementation Attacks

It is well known that a poor implementation can leak information by timing attacks [16] or power analysis attacks [17]. Using the classification proposed in [11], Camellia is in the group of "favorable" algorithms, since it uses only logical operations and table-lookups and fixed rotations.

On the other hand, Chari et al. [8] claims that all AES candidates are susceptible to power analysis attacks. As these two papers contradict with each other, how to resist against power analysis attacks is not known, since study on power analysis attacks has just begun. We think that Camellia should be protected by the hardware techniques and should not be evaluated by the security directly derived from the specification, considering the current art. We hope that the study on power analysis attack will be progressed in the near future.

6 Performance

6.1 Software Implementations

Table 7 summarizes the current software implementations of Camellia. The table shows that Camellia can be efficiently implemented on low-end smart cards, and 32-bit and 64-bit processors. We use the abbreviations M (mega) for 10^6 and m (milli) for 10^{-3} in the table.

6.2 Hardware Performance

We measured the hardware performance of 128bit-key Camellia on ASIC (Application Specific Integrated Circuit) and FPGA (Field Programmable Gate Array). Table 8 shows the environment of our hardware design and evaluation. We evaluated hardware performance of the three types: Type 1, Type 2 and Type 3 logic. The hardware design policy of each type is as follows.

Type 1 Fast implementation from the viewpoint of Enc(Dec) speed

Type 2 Small implementation from the viewpoint of total logic size

Type 3 Small implementation (special case for FPGA)

Tables 9 through 12 summarize the hardware performance of 128bit-key Camellia on ASIC (Application Specific Integrated Circuit) and FPGA (Field Programmable Gate Array).

7 Conclusion

We have presented Camellia, the rationale behind its design, its suitability for both software and hardware implementation, and the results of our cryptanalyses. For further information, please refer to the specification of Camellia [1] or full paper, which are available on the Camellia home page: `http://info.isl.ntt.co.jp/camellia/`.

The performances shown in this paper leave room for further optimizations. The latest performance results will be posted on the Camellia home page.

We have analyzed Camellia and found no important weakness. The cipher has a conservative design and any practical attacks against Camellia would require a major breakthrough in the area of cryptanalysis. We think that Camellia is a very strong cipher, which matches the security of the existing best block ciphers.

Table 7: Camellia software performance

Processor	Language	Key len. (bits)	Timing[a] Setup[e] ([f])	Timing[a] Enc.[g] ([h])	Dynamic[b] Setup[e]	Dynamic[b] Enc.[g]	Code[c] Setup[e]	Code[c] Enc.[g]	Table[d]
P III[i]	Assembly	128	160 (4.4M)	371 (242M)	28	36	1,046	2,150	8,224
		192	222 (3.2M)	494 (181M)	28	36	1,469	3,323	8,240
		256	226 (3.1M)	494 (181M)	28	36	1,485	3,323	8,240
P II[j]	ANSI C[k]	128	263 (1.1M)	577 (67M)	44	64	1,600	3,733	4,128
Alpha[l]	Assembly	128	118 (5.7M)	339 (252M)	48	48	1,132	3,076	16,528
		192	176 (3.7M)	445 (192M)	48	48	1,668	4,000	16,528
		256	176 (3.7M)	445 (192M)	48	48	1,676	4,000	16,528
		128	158 (4.2M)	326 (262M)	48	48	1,600	2,928	16,512
8051[m]	Assembly	128	0 (0)	10217 (10m)	0	32	0	702	288

[a]Number of cycles needed for setup or encryption.
[b]Dynamically used memory in bytes including stack area, excluding text and key area, which is usually located in RAM.
[c]Code size in bytes, which is sometimes located in ROM.
[d]Table size in bytes, which is sometimes located in ROM.
[e]Key schedule may be included.
[f]Seconds for 8051, and keys/s for other processors.
[g]Numbers of this column is the same as decryption, since Camellia is symmetric between encryption and decryption.
[h]Seconds for 8051, and b/s for other processors.
[i]IBM PC/AT compatible PC, Intel Pentium III (700MHz), 256KB on-die L2 cache, FreeBSD 4.0R, 128MB main memory.
[j]IBM PC/AT compatible PC, Intel Pentium II (300MHz), 512KB L2 cache, MS-Windows 95, 160MB main memory.
[k]Microsoft Visual C++ 6 with the optimization options `/G6 /Zp16 /ML /Ox /Ob2`.
[l]COMPAQ Professional Workstation XP1000, Alpha 21264 (667MHz), Compaq Tru64 UNIX 4.0F, 2GB main memory.
[m]Intel 8051 (12MHz; 1 cycle = 12 oscillator periods) simulator on Unix.

Table 8: Hardware evaluation environment (ASIC, FPGA)

Language	(ASIC, FPGA) Verilog-HDL
Simulator	(ASIC, FPGA) Verilog-XL
Design library	(ASIC) Mitsubishi Electric 0.35μ CMOS ASIC library (FPGA) Xilinx XC4000XL series
Login synthesis	(ASIC) Design Compiler version 1998.08 (FPGA) Synplify version 5.3.1 and ALLIANCE version 2.1i

Table 9: Hardware performance (Type 1: Fast implementation [ASIC(0.35μ CMOS)])

Algorithm name	Area [Gate] Enc.&Dec.[a]	Area [Gate] Key expan.[b]	Area [Gate] Total logic[c]	Key setup time [ns]	Critical-path [ns][d]	Throughput [Mb/s]
DES	42,204	12,201	54,405	—	55.11	1161.31
Triple-DES	124,888	23,207	128,147	—	157.09	407.40
MARS	690,654	2,245,096	2,935,754	1740.99	567.49	225.55
RC6	741,641	901,382	1,643,037	2112.26	627.57	203.96
Rijndael	518,508	93,708	612,834	57.39	65.64	1950.03
Serpent	298,533	205,096	503,770	114.07	137.40	931.58
Twofish	200,165	231,682	431,857	16.38	324.80	394.08
Camellia	216,911	55,907	272,819	24.36	109.35	1170.55

[a]including output registers
[b]including subkey registers
[c]including buffers for fan-out adjustment
[d]Critical path of data encryption (or decryption)

Table 10: Hardware performance (Type 2: Small implementation [ASIC(0.35μ CMOS)])

Algorithm name	Area [Gate] Enc.&Dec.[a]	Key sched.[b]	Total logic [c]	Key setup time [ns]	Critical-path [ns][d]	Throughput [Mb/s]
Camellia	6,367	4,979	11,350	110.2	27.67	220.28

[a]including output registers and data selector
[b]including subkey registers and a part of key expansion logic
[c]including buffers for fan-out adjustment
[d]Critical path of data encryption (or decryption)

Table 11: Hardware performance (Type 2: Small implementation [FPGA(XC4000XL series)])

Algorithm name	Area [CLBs] Total	Critical-path [ns][a]	Throughput [Mb/s]
Camellia	1,296	78.815	77.34

Table 12: Hardware performance (Type 3: Small implementation [FPGA(XC4000XL series)])

Algorithm name	Area [CLBs] Total	Critical-path [ns][a]	Throughput [Mb/s]
Camellia	874	49.957	122.01

[a]Critical path of data encryption (or decryption)

References

[1] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Specification of Camellia — a 128-bit Block Cipher," 2000. `http://info.isl.ntt.co.jp/camellia/`.

[2] K. Aoki, "Practical Evaluation of Security against Generalized Interpolation Attack," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences (Japan), Vol.E83-A, No.1, pp.33–38, 2000. (A preliminary version was presented at SAC'99.)

[3] K. Aoki and H. Ueda, "Optimized Software Implementations of E2," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences (Japan), Vol.E83-A, No.1, pp.101–105, 2000.

[4] E. Biham and A. Shamir, "Differential Cryptanalysis of the Data Encryption Standard," Springer-Verlag, 1993.

[5] E. Biham, "New Types of Cryptanalytic Attacks Using Related Keys," Journal of Cryptology, Vol.7, No.4, pp.229–246, Springer-Verlag, 1994.

[6] A. Biryukov and D. Wagner, "Slide Attacks," Fast Software Encryption — 6th International Workshop, FSE'99, Lecture Notes in Computer Science 1636, pp.245–259, Springer-Verlag, 1999.

[7] A. Biryukov and D. Wagner, "Advanced Slide Attacks," Advances in Cryptology — EUROCRYPT2000, Lecture Notes in Computer Science 1807, pp.589–606, Springer-Verlag, 2000.

[8] S. Chari, C. Jutla, J. R. Rao and P. Rohatgi, "A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards," Second Advanced Encryption Standard Candidate Conference, pp.133–147, 1999",

[9] F. Chabaud and S. Vaudenay, "Links Between Differential and Linear Cryptanalysis," Advances in Cryptology — EUROCRYPT'94, Lecture Notes in Computer Science 950, pp.356–365, Springer-Verlag, 1995.

[10] J. Daemen, L. R. Knudsen and V. Rijmen "The Block Cipher SQUARE," Fast Software Encryption, FSE'97, Lecture Notes in Computer Science 1267, pp.54–68, Springer-Verlag, 1997.

[11] J. Daemen and V. Rijmen, "Resistance Against Implementation Attacks. A Comparative Study of the AES Proposals," Second Advanced Encryption Standard Candidate Conference, 1999.

[12] T. Jakobsen and L. R. Knudsen, "The Interpolation Attack on Block Ciphers," Fast Software Encryption, FSE'97, Lecture Notes in Computer Science 1267, pp.28–40, Springer-Verlag, 1997.

[13] M. Kanda, "Practical Security Evaluation against Differential and Linear Attacks for Feistel Ciphers with SPN Round Function," SAC2000, Seventh Annual Workshop on Selected Areas in Cryptography, 14-15 August 2000, Workshop Record, 2000.

[14] L.R.Knudsen, "Cryptanalysis of LOKI91," Advances in Cryptology — AUSCRYPT'92, Lecture Notes in Computer Science 718, pp.196–208, Springer-Verlag, 1993.

[15] L.R.Knudsen, "Truncated and Higher Order Differentials," Fast Software Encryption — Second International Workshop, Lecture Notes in Computer Science 1008, pp.196–211, Springer-Verlag, 1995.

[16] P.Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," Advances in Cryptology — CRYPTO'96, Lecture Notes in Computer Science 1109, pp.104–113, Springer-Verlag, 1996.

[17] P.Kocher, J.Jaffe and B.Jun, "Differential Power Analysis," Advances in Cryptology — CRYPTO'99, Lecture Notes in Computer Science 1666, pp.388–397, Springer-Verlag, 1999.

[18] J.Kelsey, B.Schneier and D.Wagner, "Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES," Advances in Cryptology — CRYPTO'96, Lecture Notes in Computer Science 1109, pp.237–251, Springer-Verlag, 1996.

[19] M.Kanda, Y.Takashima, T.Matsumoto, K.Aoki and K.Ohta, "A Strategy for Constructing Fast Round Functions with Practical Security against Differential and Linear Cryptanalysis," Selected Areas in Cryptography, 5th Annual International Workshop, SAC'98, Lecture Notes in Computer Science 1556, pp.264–279, Springer-Verlag, 1999.

[20] X. Lai, J. L. Massey, and S. Murphy, "Markov Ciphers and Differential Cryptanalysis," Advances in Cryptology — EUROCRYPT'91, Lecture Notes in Computer Science 547, pp.17–38, Springer-Verlag, 1991.

[21] M. Matsui, "Linear Cryptanalysis Method for DES Cipher," Advances in Cryptology — EUROCRYPT'93, Lecture Notes in Computer Science 765, pp.386–397, Springer-Verlag, 1994.

[22] M. Matsui, "On Correlation Between the Order of S-boxes and the Strength of DES," Advances in Cryptology — EUROCRYPT'94, Lecture Notes in Computer Science 950, pp.366–375, Springer-Verlag, 1995.

[23] M. Matsui, "New Block Encryption Algorithm MISTY," Fast Software Encryption, FSE'97, Lecture Notes in Computer Science 1267, pp.54–68, Springer-Verlag, 1997.

[24] M. Matsui, "Differential Path Search of the Block Cipher E2," (in Japanese), Technical report of IEICE, ISEC99-19, pp.57–64, The Institute of Electronics, Information and Communication Engineers, 1999.

[25] M. Matsui, T. Inoue, A. Yamagishi, and H. Yoshida, "A note on calculation circuits over GF(2^{2n})," Technical Report of IEICE, IT88-14, The Institute of Electronics, Information and Communication Engineers, 1988. (in Japanese)

[26] M. Matsui and T. Tokita, "Cryptanalysis of a Reduced Version of the Block Cipher E2," Fast Software Encryption — 6th International Workshop, FSE'99, Lecture Notes in Computer Science 1636, pp.71–80, Springer-Verlag, 1999.

[27] S. Moriai, M. Sugita, K. Aoki, and M. Kanda, "Security of E2 against Truncated Differential Cryptanalysis," Selected Areas in Cryptography, 6th Annual International Workshop, SAC'99, Lecture Notes in Computer Science 1758, pp.106–117, Springer-Verlag, 2000.

[28] Nippon Telegraph and Telephone Corporation, "Specification of E2 — a 128-bit Block Cipher," 1999. `http://info.isl.ntt.co.jp/e2/`

[29] D. Wagner, "The Boomerang Attack," Fast Software Encryption — 6th International Workshop, FSE'99, Lecture Notes in Computer Science 1636, pp.156–170, Springer-Verlag, 1999.