The New Block Cipher: BC2

Yusuf Kurniawan[1], Adang Suwandi A.[2], M. Sukrisno Mardiyanto[2],
Iping Supriana S.[2], and Sarwono Sutikno[2]
(Corresponding author: Yusuf Kurniawan)

Universitas Pasundan, Department of Informatics[1]
Jl Setiabudi 193 Bandung 40153, Jawa Barat, Indonesia (Email: ysfk2002@yahoo.com)
Institute of Technology Bandung, School of Electrical Engineering and Informatics[2]
Jl Ganesha 10, Bandung 40132, Indonesia

(Received Mar. 23, 2006; revised and accepted May 7, 2006)

Abstract

In this paper, we propose a new block cipher called BC2 (Block Cipher 2). We make a cipher using components that are believed secure. The structure of BC2 is very simple. We use Feistel network with input-output 128 bits, matrix Maximum Distance Separable (MDS) 8x8 with branch number 9 to give high diffusion, a function affine equivalent to the inverse function in GF(2^8) that we get from Camellia and Hierocrypt S-Box for confusion and we make FN function, based on FL function of Camellia. We use a heuristic method to count the minimum number of active substitution box at Feistel Network. And we also construct a new key schedule that is fast and secure.

Keywords: BC2, block cipher, FN function, heuristic method

1 Introduction

In here we give some definition and list of symbols that we use.

In this paper we use finite field $GF(2^8)$ that we can represent as $GF(2)[x]/m(x)$, where $m(x) = x^8+x^4+x^3+x^2+1$. We can write m(x) as '11d' like as Khazad [11]. And we use subscript $_x$ as representation of hexadecimal. In this paper, multiplication with x is expressed as xT(number). For example, multiplication $7f_x \bullet 2_x = xT(7f) = fe_x$, and $fe_x \bullet 2_x = xT(fe) = e1_x$. This is similar to Rijndael proposal [7].

Some notations used in this paper are listed as follows:

- $\cup$ is OR;
- $\cap$ is AND;
- $\lll$ is left circular rotation by one bit;
- $\ggg$ is right circular rotation by one bit;
- $\oplus$ is bitwise XOR;
- || is concatenation of two operators;
- K_{nl} is the left side of 2n-bit key. This key part has size of n bits;
- K_r is the right side of K. The size is a half of full key.

The rest of this paper is organized as follows. The Section 2 describes the new block cipher BC2, its randomizing part and key schedule, Section 3 explains how to implement BC2 at various platforms efficiently, Section 4 explains cryptanalysis of BC2, Section 5 explains the design rationale of BC2 and Section 6 gives conclusion.

2 BC2 (Block Cipher 2)

The BC2 is a 128-bit block cipher using Feistel Network that supports 128, 192 and 256-bit key lengths. Like many other ciphers, we use Substitution Boxes to give confusion, linear layer to give diffusion and mixed key to give dependent on key. The structure of BC2 for 128-bit key length, is showed in Figure 1. For 128-bit key length, the number of round is 13. There are two FN functions. One of them is located after round 4, and the other is after round 9. The FN function have a very slow diffusion, so if we place it before first round, then attacker can arrange the input and output of FN function to easier cryptanalysis. It follows that FN function is unusable.

For 192 and 256-bit key length, the number of round is 18. There are 3 FN/FN^{-1} functions that are located after rounds 4, 9, and 14.

All F functions are same, like Figure 2. The number in F function only show the number of round.

For decryption, the order of round subkey is reversed. So, KW3 replace KW1, KW4 replace KW2, KW1 replace KW3, and KW2 replace KW4. K13 replace K1 and so forth. And then, KFN1 is replaced by KFN4, KFN2 is replaced by KFN3 and so forth.

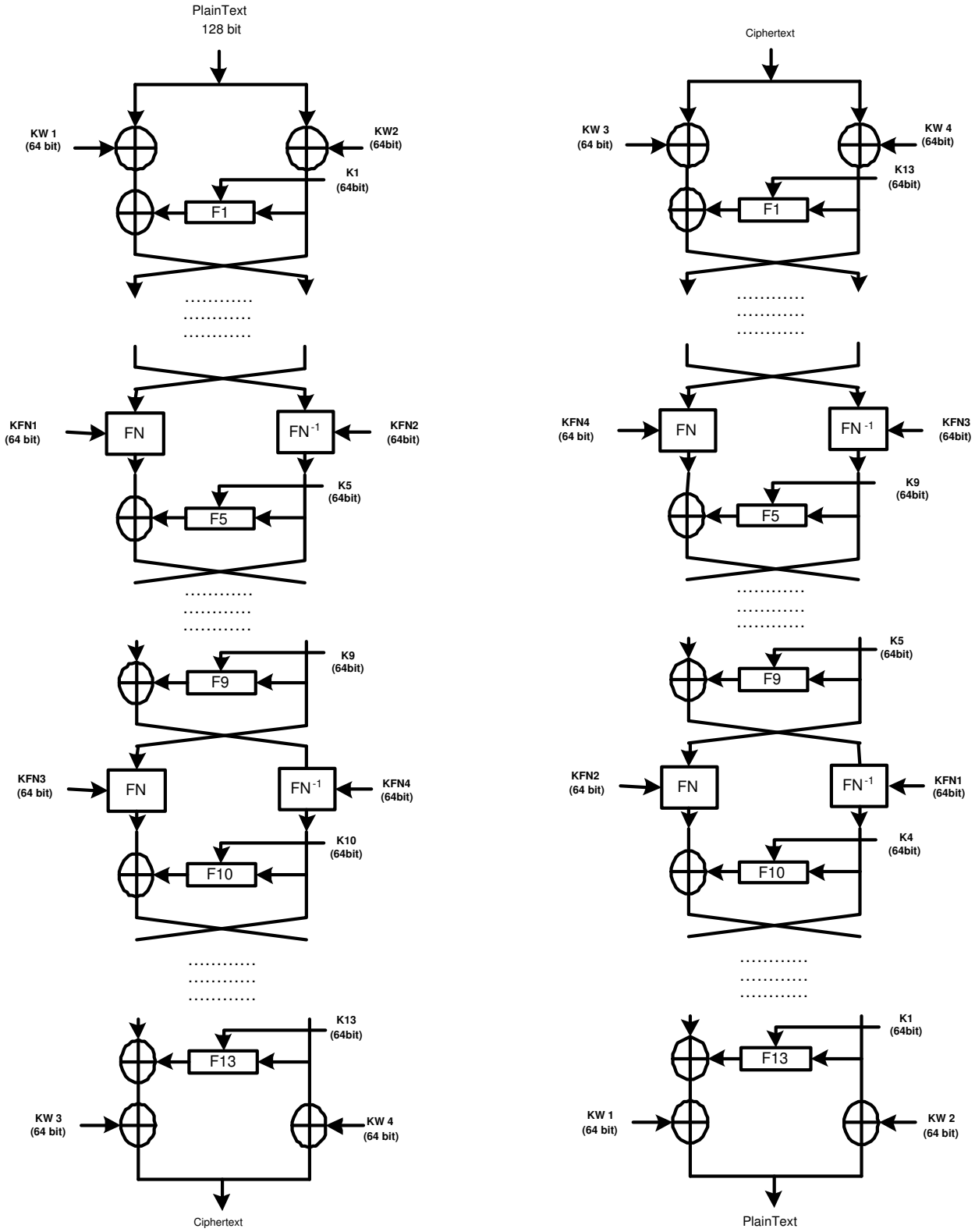

Figure 1: Encryption and decryption of BC2-128

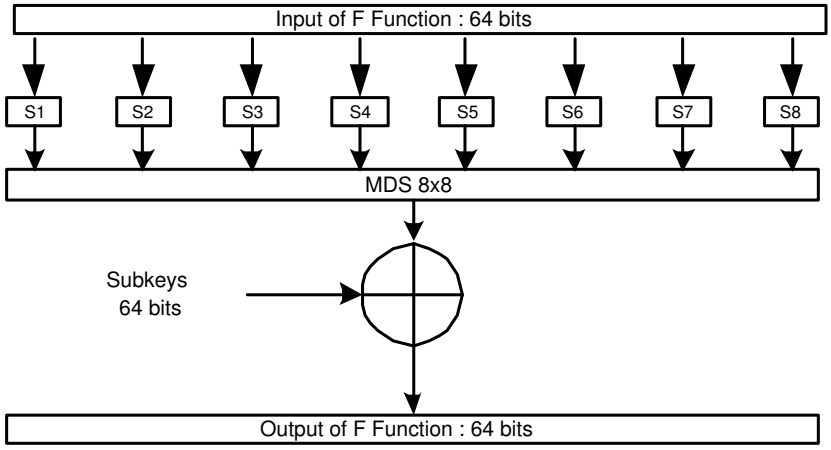

Figure 2: F function of BC2

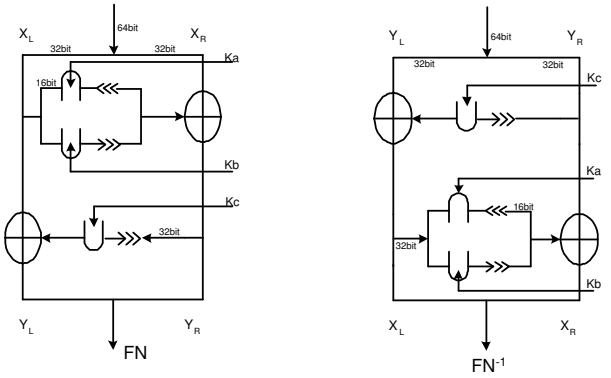

Figure 3: FN function and its inversion

2.1 Substitution Box

We use Camellia's S-Box [8] and Hierocrypt's S-Box [12] for BC2. The maximum differential probability of these S-Boxes is 2^{-6} and maximum linear probability is 2^{-4} according to our experiment with PC. The degree of them is 7.

2.2 Linear Layer *L*

We use MDS (Maximum Distance Separable) matrix to realize linear component to give high diffusion. We do not use XORs component like in Camellia cipher, because it does not give branch number exactly. We use circular matrix with low number in order to be able to be implemented efficiently in hardware.

A linear $[n, k, d]$ code $\mathcal{C}$ with generator matrix $G = [I_{k\times k}\, L_{k\times(n-k)}]$ is MDS if, and only if, every square submatrix formed from rows and columns of L is nonsingular (cf. [4], Chapter 11, § 4, Theorem 8).

We make MDS code using trial and error method until the matrix satisfies the requirement above.

Table 1: The constant for key schedule

c_1	$\text{frac}(\sqrt{0.8})$	0xe4f92e2dff6ec9ab294a33804a57d359
c_2	$\text{frac}(\sqrt{0.9})$	0xf2dce89b636cb24692e711b6e1c3ff31

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 2 & 1 & 3 & 1 & 2 \\ 2 & 1 & 2 & 3 & 2 & 1 & 3 & 1 \\ 1 & 2 & 1 & 2 & 3 & 2 & 1 & 3 \\ 3 & 1 & 2 & 1 & 2 & 3 & 2 & 1 \\ 1 & 3 & 1 & 2 & 1 & 2 & 3 & 2 \\ 2 & 1 & 3 & 1 & 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 3 & 1 & 2 & 1 & 2 \\ 2 & 3 & 2 & 1 & 3 & 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix}$$

where a_i is input of MDS and b_i is output of MDS. So b = L a.

2.3 Add Key *AK*

In this part, we use only XOR component to avoid weakness that we can find in IDEA cipher.

2.4 Key Schedule

We construct a new key schedule with the criteria:

1) simple and fast for many platforms
2) it should be resistant to related key attack
3) it should be hard to find masterkey if attacker can get (partial) subkey(s).
4) there are no weak keys.
5) every bit of masterkey gives influence to all subkeys.

We use the basic instructions (like XOR, AND, OR, 1-bit rotation) to achieve Objectives 1, 2, and 3. We also use the matrix component(like in Rijndael) in key schedule to achieve Objective 4. This component gives high diffusion

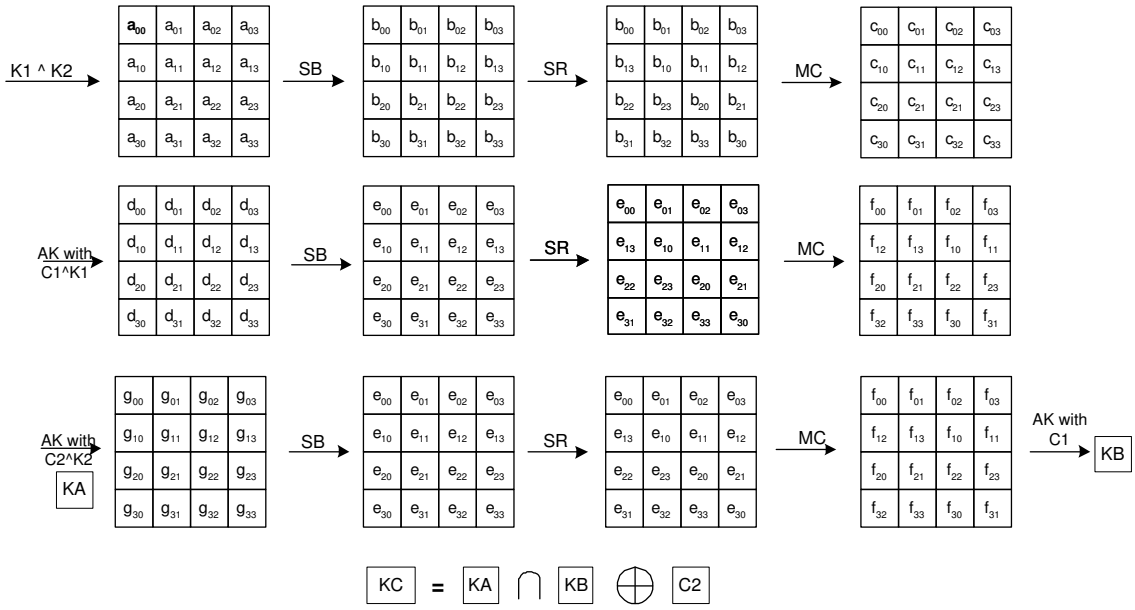

Figure 4: key schedule of BC2

and confusion. To achieve the last objective, we use high diffusion that we get from MixColumn function. We can see key schedule at Figure 4. Masterkey is composed from K1 and K2, K1 || K2.

If we only need 128 bits, so we set K2=0, and if we need 192 bits, the last half of K2 is set to zero. From Figure 4 we get KA, KB, and KC. We use square matrix (like as Rijndael) to create subkeys. At first, we perform XOR operation between K1 and K2 (AK). Then we substitute them with Camellia and Hierocrypt S-Box (SB). Then we rotate their bytes ShiftRows (SR)and use MixColumn (MC) to give high diffusion. The matrix of MixColumn is similar to matrix at linear component in randomizing part as follows:

$$\begin{bmatrix} 1 & 2 & 3 & 2 \\ 2 & 1 & 2 & 3 \\ 1 & 2 & 1 & 2 \\ 3 & 1 & 2 & 1 \end{bmatrix}$$

Outputs of MCs are XORed with constant (Table 1) and Masterkey. The outputs of this process are KA, KB, and KC. From these keys, we compose all subkeys, like as Tables 2 and 3.

3 Implementation

In this section, we explain how to implement BC2 at various platform. If input of F function is *IF*, substitution operation is *SB*, *L* is linear operation, *AK* is AddKey, and output of F function is OF, then we can write $OF = AK(L(SB(IF)))$.

3.1 64-bit Processors

In this platform, BC2 can be implemented very efficiently. Like as Khazad or Rijndael cipher, we can write

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} SB[x_0] \\ SB[x_0] \bullet 2 \\ SB[x_0] \\ SB[x_0] \bullet 3 \\ SB[x_0] \\ SB[x_0] \bullet 2 \\ SB[x_0] \bullet 3 \\ SB[x_0] \bullet 2 \end{bmatrix} \oplus \begin{bmatrix} SB[x_1] \bullet 2 \\ SB[x_1] \\ SB[x_1] \bullet 2 \\ SB[x_1] \\ SB[x_1] \bullet 3 \\ SB[x_1] \\ SB[x_1] \bullet 2 \\ SB[x_1] \bullet 3 \end{bmatrix} \oplus$$

$$\begin{bmatrix} SB[x_2] \bullet 3 \\ SB[x_2] \bullet 2 \\ SB[x_2] \\ SB[x_2] \bullet 2 \\ SB[x_2] \\ SB[x_2] \bullet 3 \\ SB[x_2] \\ SB[x_2] \bullet 2 \end{bmatrix} \oplus \begin{bmatrix} SB[x_3] \bullet 2 \\ SB[x_3] \bullet 3 \\ SB[x_3] \bullet 2 \\ SB[x_3] \\ SB[x_3] \bullet 2 \\ SB[x_3] \\ SB[x_3] \bullet 3 \\ SB[x_3] \end{bmatrix} \oplus \begin{bmatrix} SB[x_4] \\ SB[x_4] \bullet 2 \\ SB[x_4] \bullet 3 \\ SB[x_4] \bullet 2 \\ SB[x_4] \\ SB[x_4] \bullet 2 \\ SB[x_4] \\ SB[x_4] \bullet 3 \end{bmatrix} \oplus$$

$$\begin{bmatrix} SB[x_5] \bullet 3 \\ SB[x_5] \\ SB[x_5] \bullet 2 \\ SB[x_5] \bullet 3 \\ SB[x_5] \bullet 2 \\ SB[x_5] \\ SB[x_5] \bullet 2 \\ SB[x_5] \end{bmatrix} \oplus \begin{bmatrix} SB[x_6] \\ SB[x_6] \bullet 3 \\ SB[x_6] \\ SB[x_6] \bullet 2 \\ SB[x_6] \bullet 3 \\ SB[x_6] \bullet 2 \\ SB[x_6] \\ SB[x_6] \bullet 2 \end{bmatrix} \oplus \begin{bmatrix} SB[x_7] \bullet 2 \\ SB[x_7] \\ SB[x_7] \bullet 3 \\ SB[x_7] \\ SB[x_7] \bullet 2 \\ SB[x_7] \bullet 3 \\ SB[x_7] \bullet 2 \\ SB[x_7] \end{bmatrix}$$

where x_i is input of SBox-i and x is input of F function.

Table 2: key schedule for 128 bit key

$KW1$	$KA_l \oplus KB_l \oplus KC_l$	$SK8$	$(SK6_{32l} \lll 1)\|\|(SK6_{32r} \lll 1)$
$KW2$	$KA_r \oplus KB_r \oplus KC_r$	$SK9$	$(SK7_{32l} \lll 1)\|\|(SK7_{32r} \lll 1)$
$SK1$	$(KW1 \cup KW2) \oplus KA_l$	$KFN3$	$KA_l \oplus SK8 \oplus KB_l$
$SK2$	$(KW1 \cap KW2) \oplus KB_l$	$KFN4$	$(KA_r \cap SK9) \oplus KB_r$
$SK3$	$(SK1 \cup SK2) \oplus KA_r$	$SK10$	$SK1 \oplus SK5 \oplus KFN3$
$SK4$	$(SK1 \cap SK2) \oplus KB_r$	$SK11$	$(SK2 \cup SK6) \oplus KFN4$
$KFN1$	$(KA_l \cup SK3) \oplus KC_l$	$SK12$	$(SK8 \cap SK10) \oplus SK5$
$KFN2$	$(KB_l \cup SK4) \oplus KC_r$	$SK13$	$(SK9 \cup SK10) \oplus SK6$
$SK5$	$(KA_l \cup KB_l) \oplus KFN2$	$KW3$	$SK10 \oplus SK11 \oplus SK12$
$SK6$	$(KA_r \cup KB_r) \oplus KC_r$	$KW4$	$SK5 \oplus SK6 \oplus SK7$
$SK7$	$(SK5_{32l} \lll 1)\|\|(SK5_{32r} \lll 1)$		

Table 3: key schedule for 192 and 256-bit key

$KW1$	$KA_l \oplus KB_l \oplus KC_l$	$KFNcentre2$	$(KA_r \cap SK9) \oplus KB_r$
$KW2$	$KA_r \oplus KB_r \oplus KC_r$	$SK10$	$SK1 \oplus SK5 \oplus KFNcentre1$
$SK1$	$(KW1 \cup KW2) \oplus KA_l$	$SK11$	$(SK2 \cup SK6) \oplus KFNcentre2$
$SK2$	$(KW1 \cap KW2) \oplus KB_l$	$SK12$	$(SK8 \cap SK10) \oplus SK5$
$SK3$	$(SK1 \cup SK2) \oplus KA_r$	$SK13$	$(SK9 \cup SK10) \oplus SK6$
$SK4$	$(SK1 \cap SK2) \oplus KB_r$	$SK14$	$(SK11_{32l} \lll 1)\|\|(SK11_{32r} \lll 1)$
$KFN1$	$(KA_l \cup SK3) \oplus KC_l$	$KFN3$	$(SK1 \cup SK5) \oplus KC_l)$
$KFN2$	$(KB_l \cup SK4) \oplus KC_r$	$KFN4$	$SK2 \oplus SK6 \oplus SK11$
$SK5$	$(KA_l \cup KB_l) \oplus KFN2$	$SK15$	$(SK7 \cap KC_l) \oplus SK12$
$SK6$	$(KA_r \cup KB_r) \oplus KC_r$	$SK16$	$(SK8 \cup KC_r) \oplus SK13$
$SK7$	$(SK5_{32l} \lll 1)\|\|(SK5_{32r} \lll 1)$	$SK17$	$(SK9 \cup KW1) \oplus SK14$
$SK8$	$(SK6_{32l} \lll 1)\|\|(SK6_{32r} \lll 1)$	$SK18$	$(SK10 \cup KW2) \oplus SK15$
$SK9$	$(SK7_{32l} \lll 1)\|\|(SK7_{32r} \lll 1)$	$KW3$	$SK10 \oplus SK11 \oplus SK12$
$KFNcentre1$	$KA_l \oplus SK8 \oplus KB_l$	$KW4$	$SK5 \oplus SK6 \oplus SK7$

If we define

$$T_0 = \begin{bmatrix} SB[x_0] \\ SB[x_0] \bullet 2 \\ SB[x_0] \\ SB[x_0] \bullet 3 \\ SB[x_0] \\ SB[x_0] \bullet 2 \\ SB[x_0] \bullet 3 \\ SB[x_0] \bullet 2 \end{bmatrix} \qquad T_1 = \begin{bmatrix} SB[x_1] \bullet 2 \\ SB[x_1] \\ SB[x_1] \bullet 2 \\ SB[x_1] \\ SB[x_1] \bullet 3 \\ SB[x_1] \\ SB[x_1] \bullet 2 \\ SB[x_1] \bullet 3 \end{bmatrix}$$

and so forth, then we have:

$$OF = T_0 \oplus T_1 \oplus T_2 \oplus T_3 \oplus T_4 \oplus T_5 \oplus T_6 \oplus T_7 \oplus SK,$$

where SK is subkey at each round. All T tables require 16 k bytes.

3.2 32-bit Processors

To this platform we can write:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = a_0 \begin{bmatrix} 1 \\ 2 \\ 1 \\ 3 \end{bmatrix} \oplus a_1 \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \end{bmatrix} \oplus a_2 \begin{bmatrix} 3 \\ 2 \\ 1 \\ 2 \end{bmatrix} \oplus a_3 \begin{bmatrix} 2 \\ 3 \\ 2 \\ 1 \end{bmatrix} \oplus a_4 \begin{bmatrix} 1 \\ 2 \\ 3 \\ 2 \end{bmatrix} \oplus a_5 \begin{bmatrix} 3 \\ 1 \\ 2 \\ 3 \end{bmatrix} \oplus a_6 \begin{bmatrix} 1 \\ 3 \\ 1 \\ 2 \end{bmatrix} \oplus a_7 \begin{bmatrix} 2 \\ 1 \\ 3 \\ 1 \end{bmatrix}$$

or $OF_{0-3} = T[0] \oplus T[1] \oplus T[2] \oplus T[3] \oplus T[4] \oplus T[5] \oplus T[6] \oplus T[7] \oplus SK_{0-3}$.

$$\begin{bmatrix} b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = a_0 \begin{bmatrix} 1 \\ 2 \\ 3 \\ 2 \end{bmatrix} \oplus a_1 \begin{bmatrix} 3 \\ 1 \\ 2 \\ 3 \end{bmatrix} \oplus a_2 \begin{bmatrix} 1 \\ 3 \\ 1 \\ 2 \end{bmatrix} \oplus a_3 \begin{bmatrix} 2 \\ 1 \\ 3 \\ 1 \end{bmatrix} \oplus a_4 \begin{bmatrix} 1 \\ 2 \\ 1 \\ 3 \end{bmatrix} \oplus a_5 \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \end{bmatrix} \oplus a_6 \begin{bmatrix} 3 \\ 2 \\ 1 \\ 2 \end{bmatrix} \oplus a_7 \begin{bmatrix} 2 \\ 3 \\ 2 \\ 1 \end{bmatrix}$$

or $OF_{4-7} = T[8] \oplus T[9] \oplus T[10] \oplus T[11] \oplus T[12] \oplus T[13] \oplus T[14] \oplus T[15] \oplus SK_{4-7}$ and $a_i = SB[x_i]$.

In this method, All T Tables require 2^4 x 4 x 2^8 = 2^{14} bytes. If we use one table for T[0] and T[12], one for T[4] and T[8], and so forth, then we need only 8 k bytes. The speed comparison of BC2 with other block ciphers at personal computer can be seen at appendix.

3.3 8-bit Processors

For this platform, the method that we describe above is unsuitable. So we use other method. We can write linear

layer as follows:

$$
\begin{aligned}
r_0 &= a_0 \oplus a_2 \oplus a_4 \oplus a_6 \\
r_1 &= a_1 \oplus a_3 \oplus a_5 \oplus a_7 \\
r_2 &= xT(r_0) \\
r_3 &= xT(r_1) \\
b_0 &= r_0 \oplus a_5 \oplus r_3 \oplus xT(a_2) \\
b_1 &= r_1 \oplus a_6 \oplus r_2 \oplus xT(a_3) \\
b_2 &= r_0 \oplus a_7 \oplus r_3 \oplus xT(a_4) \\
b_3 &= r_1 \oplus a_0 \oplus r_2 \oplus xT(a_5) \\
b_4 &= r_0 \oplus a_1 \oplus r_3 \oplus xT(a_6) \\
b_5 &= r_1 \oplus a_2 \oplus r_2 \oplus xT(a_7) \\
b_6 &= r_0 \oplus a_3 \oplus r_3 \oplus xT(a_0) \\
b_7 &= r_1 \oplus a_4 \oplus r_2 \oplus xT(a_1).
\end{aligned}
$$

In this method we need four registers, 30 exors, 10 xT operations, and 12 assignments for linear layer implementation. If we have six registers, then we can reduce the operation. We write r4=$r_0 \oplus r_3$ and $r5 = r_1 \oplus r_2$. The operations of SBox and Addkey are performed per byte.

3.4 Key Schedule Implementation

We use the same component in key schedule and randomizing part to give efficiency in implementation. We also use the basic instruction (OR, XOR, AND, 1-bit rotation) in key schedule in order to be able to be implemented efficiently at various platforms.

4 Cryptanalysis

4.1 Differential and Linear Cryptanalysis

In this section we discuss about how to measure maximum differential and linear probability (DP_{max} and LP_{max}) of BC2 without FN and FN^{-1} functions. We use heuristic method to count the minimal number of active substitution boxes. For differential attack [2], we use characteristics $a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7$ for left side plaintext(64 bits). So, the size of a_i is 1 byte. And for right side we use b',0,0,0,0,0,0,0,0 (64 bits). Difference b' is chosen so that the output of F function at round 1 is same as $a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7$ so the difference input of round 2 is b',0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0, because there is cancellation between output of F function with difference of left side in plaintext. So we get the minimal number of active S-Box for first two round of BC2 is 1. And difference at input of round 3 is 0,0,0,0,0,0,0,0, b',0,0,0,0,0,0,0,0 (look at Figure 5). So, number active SBox in round 1 is one, in round 2 is zero, in round 3 is one.

Since the branch number of linear layer is 9, and the left difference of input at round 3 is zero, then the difference of input at round 4 become b',0,0,0,0,0,0,0,0, $c_0, c_2, c_3, c_4, c_5, c_6, c_7$. And it follows the number of active S-Box in round 4 become 8.

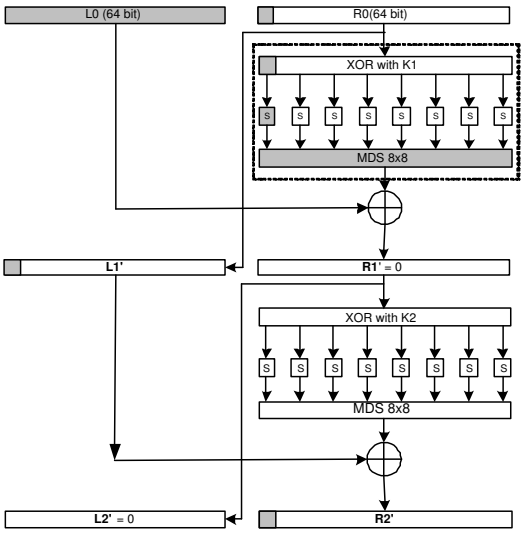

Figure 5: Active S-box in BC2

If we continue this method, then after 10 rounds, the minimum number of active S-Box become 28. So, DP_{max} is $(2^{-6})^{28} = 2^{-168}$ and since the behavior of linear attack [10] looks like differential attack, so LP_{max} is $(2^{-4})^{28} = 2^{-112}$. For differential attack, we need 2^{168} chosen plaintext pairs, and for linear attack, we need 2^{224} known plaintext. And since we also consider 3R-attack, so we need 13 rounds. And since we consider the worst case of linear/differential attack of BC2, so we hope BC2 stronger against these attacks than we predict, moreover if we consider FN function.

As comparison, maximum differential/linear characteristic probabilities of Camellia cipher reduced to 16 round without FL and FL^{-1}, respectively, are 2^{-132} and 2^{-88}. If we use this heuristic method to count active SBox in Camellia, we get 26 active SBox, at least, in 16 rounds, so DP_{max}=$(2^{-6})^{26} = 2^{-156}$ and $LP_{max} = (2^{-4})^{26} = 2^{-104}$. This probability can cryptanalysis Camellia with 2R attack.

4.2 Square Attack and Its Variant

Cipher having byte oriented is vulnerable with square attack [6] and its variant [5]. In BC2, the property of $\oplus_{i=0}^{255} p = 0$ where p is byte plaintext, is hold till the input of round 5. So square attack and its variant are very unlikely to succeed for full round (13 rounds).

4.3 Higher Order Differential Attack

In general, a cipher with a low non-linear order is vulnerable to this attack. Since BC2 use non-linear component with degree 7, so after a few rounds, the degree will increase rapidly. Moreover, BC2 has 13 rounds, so this attack is impossible to be done. Moreover, the FN function in BC2 increase resistance to this attack, like Misty cipher that have only low degree in its S-Box.

4.4 Interpolation Attack

A cipher with S-Box having simple algebraic is vulnerable to interpolation attack [13]. But, S-Box of BC2 use addition of affine function, so this attack seems very unlikely to succeed for this cipher.

4.5 Related-key Attack and Slide Attack

Related-key attack [3] can work if there is slow diffusion or symmetry in the key schedule. Since key schedule of BC2 uses function that has fast diffusion and nonlinear operation and uses different mixing operation of XOR, OR, AND and rotation at each round subkey, we hope this method is very effective in countering all kinds of known key based attacks. Every bit of masterkey influences KB and KC. And every round subkey is influenced by KA, KB and KC directly or indirectly. So we hope the weakness in the key schedule of SAFER can be hindered. And since the confusion component is not influenced by subkey directly (BC2 use XOR to mix subkey) so the weakness that one find in IDEA, is very unlikely to succeed for BC2.

Slide Attack [1] can work if there is symmetry in the randomizing part of cipher and in the key schedule. Since BC2 has FN function, so the symmetry in the randomizing part decreases. Moreover, the different process in the each round of key schedule, make this attack very unlikely to succeed.

5 Design Rationale

5.1 Non-linear Component

We choose S-Box from Camellia and Hierocrypt because these components have very excellent features. They have maximum differential probability 2^{-6}, maximum linear probability 2^{-4} and degree 7. So these components can be resistance against differential, linear and higher order differential attacks. The affine function in these components can improve the BC2 strength to interpolation attack and other algebraic attacks.

5.2 Linear Component

We use MDS (Maximum Distance Separable) to increase the number of active S-Box, so BC2 can be resistance to linear/differential attack. MDS gives high diffusion that is also important to face boomerang attack.

5.3 FN Function

This component is made to face unknown attacks. FN component also can damage path of linear hull and impossible differential attack. FN is designed more complicated than Camellia has, in order to give more protection, for example, against truncated differential attack [9]. This attack use partial of plaintext to predict partial of ciphertext with high probability. A byte-oriented cipher is vulnerable to this attack, so we add two more rotation to break this alignment.

6 Conclusions

We proposed a new block cipher algorithm BC2. We design a new keyschedule that is fast and one-way function. So, it should hard to find masterkey if attacker can get subkey. We also use differential and linear attack to attack BC2. Our method to search linear/differential path can be used to attack other BC2-like ciphers if we know their branch number.

References

[1] A. Biryukov and D. Wagner, "Slide attacks," in *Proceedings of Fast Software Encryption*, LNCS 1636, pp. 245-259, Springer-Verlag, 1999.

[2] E. Biham and A. Shamir, "Differential cryptanalysis of the DES-like cryptosystems," in *Advances in Cryptology (Crypto'90)*, pp. 2-21, Springer Verlag, 1993.

[3] E. Biham, "New types of cryptanalytic attacks using related keys," *Journal of Cryptology*, vol. 7, no. 4, pp. 229-246, 1994.

[4] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland Mathematical Library, vol. 16, 1977.

[5] H. Gilbert and M. Minier, "A collision attack on 7 rounds of Rijndael," in *the Proceedings of The Third AES Candidate Conference*, pp. 230-241, 2000.

[6] J. Daemen, L. Knudsen, and V. Rijmen, "The block cipher SQUARE," in *the Proceedings of Fast Software Encryption 1997*, LNCS 1267, pp. 149-165, Springer-Verlag, 1997.

[7] J. Daemen and V. Rijmen, "AES proposal: Rijndael," *AES submission*. (http://www.nist.gov/aes)

[8] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moria, J. Nakajima, and T. Tokita, "Camellia: A 128-bit block cipher suitable for multiple platform - Design and analysis," in *Proceedings of Selected Areas in Cryptography* , LNCS 2012, pp. 39-56, Springer-Verlag, 2001.

[9] L. R. Knudsen, "Truncated and higher order differentials," in *Fast Software Encryption*, LNCS 1008, pp. 196-211, Springer-Verlag, 1995.

[10] M. Matsui, "Linear cryptanalysis method for DES cipher," in *Advances in Cryptology (Eurocrypt'93)*, pp. 386-397, 1993.

[11] P. S. L. M. Barreto and V. Rijmen, "The Khazad legacy-level block cipher," in *New European Schemes for Signature, Integrity, and Encryption*, pp. 84-87, 2000.

[12] K. Ohkuma, H. Shimizu, F. Sano, and S. Kawamura, "The block cipher Hierocrypt," in *Proceedings of Selected Areas in Cryptography*, LNCS 2012, pp. 72-88, Springer-Verlag, 2001.

[13] T. Jakobsen and L.R. Knudsen, "The interpolation attack on block ciphers," *Fast Software Encryption*, LNCS 1267, pp. 28-40, Springer-Verlag, 1997.

Appendix A: Substitution Boxes

In this section, we can see the substitution box from Camellia called SB_C and one from Hierocrypt called SB_H.

```
const byte SB_C[256] = {
```

$70_x, 2c_x, b3_x, c0_x, e4_x, 57_x, ea_x, ae_x, 23_x, 6b_x, 45_x, a5_x, ed_x, 4f_x, 1d_x, 92_x,$
$86_x, af_x, 7c_x, 1f_x, 3e_x, dc_x, 5e_x, 0b_x, a6_x, 39_x, d5_x, 5d_x, d9_x, 5a_x, 51_x, 6c_x,$
$8b_x, 9a_x, fb_x, b0_x, 74_x, 2b_x, f0_x, 84_x, df_x, cb_x, 34_x, 76_x, 6d_x, a9_x, d1_x, 04_x,$
$14_x, 3a_x, de_x, 11_x, 32_x, 9c_x, 53_x, f2_x, fe_x, cf_x, c3_x, 7a_x, 24_x, e8_x, 60_x, 69_x,$
$aa_x, a0_x, a1_x, 62_x, 54_x, 1e_x, e0_x, 64_x, 10_x, 00_x, a3_x, 75_x, 8a_x, e6_x, 09_x, dd_x,$
$87_x, 83_x, cd_x, 90_x, 73_x, f6_x, 9d_x, bf_x, 52_x, d8_x, c8_x, c6_x, 81_x, 6f_x, 13_x, 63_x,$
$e9_x, a7_x, 9f_x, bc_x, 29_x, f9_x, 2f_x, b4_x, 78_x, 06_x, e7_x, 71_x, d4_x, ab_x, 88_x, 8d_x,$
$72_x, b9_x, f8_x, ac_x, 36_x, 2a_x, 3c_x, f1_x, 40_x, d3_x, bb_x, 43_x, 15_x, ad_x, 77_x, 80_x,$
$82_x, ec_x, 27_x, e5_x, 85_x, 35_x, 0c_x, 41_x, ef_x, 93_x, 19_x, 21_x, 0e_x, 4e_x, 65_x, bd_x,$
$b8_x, 8f_x, eb_x, ce_x, 30_x, 5f_x, c5_x, 1a_x, e1_x, ca_x, 47_x, 3d_x, 01_x, d6_x, 56_x, 4d_x,$
$0d_x, 66_x, cc_x, 2d_x, 12_x, 20_x, b1_x, 99_x, 4c_x, c2_x, 7e_x, 05_x, b7_x, 31_x, 17_x, d7_x,$
$58_x, 61_x, 1b_x, 1c_x, 0f_x, 16_x, 18_x, 22_x, 44_x, b2_x, b5_x, 91_x, 08_x, a8_x, fc_x, 50_x,$
$d0_x, 7d_x, 89_x, 97_x, 5b_x, 95_x, ff_x, d2_x, c4_x, 48_x, f7_x, db_x, 03_x, da_x, 3f_x, 94_x,$
$5c_x, 02_x, 4a_x, 33_x, 67_x, f3_x, 7f_x, e2_x, 9b_x, 26_x, 37_x, 3b_x, 96_x, 4b_x, be_x, 2e_x,$
$79_x, 8c_x, 6e_x, 8e_x, f5_x, b6_x, fd_x, 59_x, 98_x, 6a_x, 46_x, ba_x, 25_x, 42_x, a2_x, fa_x,$
$07_x, 55_x, ee_x, 0a_x, 49_x, 68_x, 38_x, a4_x, 28_x, 7b_x, c9_x, c1_x, e3_x, f4_x, c7_x, 9e_x,$

```
};
```

So, $SB\_C[0] = 70$ hex, $SB\_C[8] = 23$ hex, $SB\_C[16] = 86$ hex and so forth.

```
const byte SB_H[256] = {
```

$07_x, fc_x, 55_x, 70_x, 98_x, 8e_x, 84_x, 4e_x, bc_x, 75_x, ce_x, 18_x, 02_x, e9_x, 5d_x, 80_x,$
$1c_x, 60_x, 78_x, 42_x, 9d_x, 2e_x, f5_x, e8_x, c6_x, 7a_x, 2f_x, a4_x, b2_x, 5f_x, 19_x, 87_x,$
$0b_x, 9b_x, 9c_x, d3_x, c3_x, 77_x, 3d_x, 6f_x, b9_x, 2d_x, 4d_x, f7_x, 8c_x, a7_x, ac_x, 17_x,$
$3c_x, 5a_x, 41_x, c9_x, 29_x, ed_x, de_x, 27_x, 69_x, 30_x, 72_x, a8_x, 95_x, 3e_x, f9_x, d8_x,$
$21_x, 8b_x, 44_x, d7_x, 11_x, 0d_x, 48_x, fd_x, 6a_x, 01_x, 57_x, e5_x, bd_x, 85_x, ec_x, 1e_x,$
$37_x, 9f_x, b5_x, 9a_x, 7c_x, 09_x, f1_x, b1_x, 94_x, 81_x, 82_x, 08_x, fb_x, c0_x, 51_x, 0f_x,$
$61_x, 7f_x, 1a_x, 56_x, 96_x, 13_x, c1_x, 67_x, 99_x, 03_x, 5e_x, b6_x, ca_x, fa_x, 9e_x, df_x,$
$d6_x, 83_x, cc_x, a2_x, 12_x, 23_x, b7_x, 65_x, d0_x, 39_x, 7d_x, 3b_x, d5_x, b0_x, af_x, 1f_x,$
$06_x, c8_x, 34_x, c5_x, 1b_x, 79_x, 4b_x, 66_x, bf_x, 88_x, 4a_x, c4_x, ef_x, 58_x, 3f_x, 0a_x,$
$2c_x, 73_x, d1_x, f8_x, 6b_x, e6_x, 20_x, b8_x, 22_x, 43_x, b3_x, 33_x, e7_x, f0_x, 71_x, 7e_x,$
$52_x, 89_x, 47_x, 63_x, 0e_x, 6d_x, e3_x, be_x, 59_x, 64_x, ee_x, f6_x, 38_x, 5c_x, f4_x, 5b_x,$
$49_x, d4_x, e0_x, f3_x, bb_x, 54_x, 26_x, 2b_x, 00_x, 86_x, 90_x, ff_x, fe_x, a6_x, 7b_x, 05_x,$
$ad_x, 68_x, a1_x, 10_x, eb_x, c7_x, e2_x, f2_x, 46_x, 8a_x, 6c_x, 14_x, 6e_x, cf_x, 35_x, 45_x,$
$50_x, d2_x, 92_x, 74_x, 93_x, e1_x, da_x, ae_x, a9_x, 53_x, e4_x, 40_x, cd_x, ba_x, 97_x, a3_x,$
$91_x, 31_x, 25_x, 76_x, 36_x, 32_x, 28_x, 3a_x, 24_x, 4c_x, db_x, d9_x, 8d_x, dc_x, 62_x, 2a_x,$
$ea_x, 15_x, dd_x, c2_x, a5_x, 0c_x, 04_x, 1d_x, 8f_x, cb_x, b4_x, 4f_x, 16_x, ab_x, aa_x, a0_x,$

```
};
```

So, $SB\_H[0] = 7$ hex, $SB\_H[8] = bc$ hex, $SB\_H[16] = 1c$ hex and so forth.

Table 5: The speed comparison

Block ciphers	keyschedule time(μs)	encryption time per 128-bit data (μs)	encryption rate (Mbit/s)
3DES-168	4.5516	5.7	22.456
BC2-128	0.7926	1.4373	89.0506
BC2-192	0.9693	2.0123	63.6076
BC2-256	0.965	2.0121	63.6223
Camellia-128	1.4811	1.6724	76.53671
Camellia-192	2.0029	2.1681	59.03787
Camellia-256	2.0139	2.1751	58.84687
AES-128	2.3403	1.0405	123.0178
AES-192	2.4405	1.2428	102.9932
AES-256	3.0464	1.3029	98.24238
Serpent-128	12.3297	2.6038	49.1589
Serpent-192	14.2815	2.6047	49.1419
Serpent-256	16.4357	2.5997	49.236

Appendix B: The Characteristic for Differential/linear Attack

In this section, we give the complete table of minimum number of active Sbox in BC2. From Table 4 we can count that the minimum number of active SBox of BC2 10 round is 28, for BC2 13 round is 37 and for BC2 16 round is 46. So for 3R attack, the DP_{max} is $(2^{-6})^{28} = 2^{-168}$, $(2^{-6})^{37} = 2^{-222}$ and $(2^{-6})^{46} = 2^{-276}$, respectively. For linear attack, LP_{max} is $2^{-112}, 2^{-148}$, and 2^{-184} respectively.

We also use Table 4 to count minimum number of active SBox of other Feistel cipher, if we know their branch number. For example, the branch number of Camellia is 5 so we can change the number "8" in the table with "4". And then, we get the minimum number of active SBox is 11 for Camellia 8 round. And for Camellia 16 round, the minimum number of active SBox is 26.

Appendix C: The Speed Comparison of BC2 with Other Ciphers

In this section, we give speed comparison of BC2 with other block ciphers at personal computer. We use C ANSI with Borland C++ v6.0 compiler, 1200 Mhz AMD Duron processor, 512 MB RAM, and Windows XP sp2 to compare them.

The key schedule of BC2 is one of the fastest of all other ciphers.

Yusuf Kurniawan received the B.S. degree and the master degree in electrical engineering from Institut Teknologi Bandung (ITB), Bandung, Indonesia, in 1994 and 1997, respectively. He is currently the Doctoral Student of School of Electrical Engineering and Informatics at the ITB. His research interests focus on the design of

Table 4: The characteristic for differential/linear attack of BC2

round	left	right	Minimum number of active SBox
1	$a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8$	$b_1, 0, 0, 0, 0, 0, 0, 0$	1
2	$b_1, 0, 0, 0, 0, 0, 0, 0$	0,0,0,0,0,0,0,0	0
3	0,0,0,0,0,0,0,0	$b_1, 0, 0, 0, 0, 0, 0, 0$	1
4	$b_1, 0, 0, 0, 0, 0, 0, 0$	$c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8$	8
5	$c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8$	$0, 0, 0, 0, 0, 0, 0, 0$	0
6	0,0,0,0,0,0,0,0	$c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8$	8
7	$c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8$	$e_1, 0, 0, 0, 0, 0, 0, 0$	1
8	$e_1, 0, 0, 0, 0, 0, 0, 0$	0,0,0,0,0,0,0,0	0
9	0,0,0,0,0,0,0,0	$e_1, 0, 0, 0, 0, 0, 0, 0$	1
10	$e_1, 0, 0, 0, 0, 0, 0, 0$	$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8$	8
11	$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8$	$0, 0, 0, 0, 0, 0, 0, 0$	0
12	0,0,0,0,0,0,0,0	$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8$	8
13	$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8$	$f_1, 0, 0, 0, 0, 0, 0, 0$	1
14	$f_1, 0, 0, 0, 0, 0, 0, 0$	0,0,0,0,0,0,0,0	0
15	0,0,0,0,0,0,0,0	$f_1, 0, 0, 0, 0, 0, 0, 0$	1
16	$f_1, 0, 0, 0, 0, 0, 0, 0$	$g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8$	8

block cipher and cryptology.

Adang Suwandi A. received B.S. degree in Electrical & Control Engineering from Electrical Engineering Department ITB, Bandung, Indonesia, in 1976 and the Docteur Ingenieur in Signaux et Bruits Option Electronique from Universite des Sciences Technique du Languedoc Montpellier France He is currently Chair and Professor of School of Electrical Engineering and Informatics. at the ITB. His field interests are intelligent system instrumentation & Bioinformatics.

M. Sukrisno Mardiyanto received DEA and Docteur Ingenieur from Institute National Polytechnique de Grenoble (INPG), France in 1982 and 1986, respectively. He has been chair of Study Program of Informatics at ITB since 2005. His research interests focus on Software Engineering, Computer Network and Computer Security.

Iping Supriana S. received Doctoral degree from INPG, France in 1985. He is a senior lecturer at School of Electrical Engineering and Informatics at the ITB. He is chair of security software project of digital mark reader at ITB.

Sarwono Sutikno received B.S in Electronics degree from Institute Technoloy of Bandung, Bandung, Indonesia, in 1984, and received the Master of Engineering degree and Doctor of Engineering degree in Integrated System from Tokyo Institute of Technology, Tokyo, Japan in 1990 and 1994, respectively. His research interests focus on implementation of cryptographics algorithms in Integrated Circuits including Embedded System Security. His Security Engineering focus includes Information Security Management System. He holds several professional certifications including Certified Information System Auditor and ISMS Provisional Auditor, he is also appointed ISACA Academic Advocate.