The New Block Cipher RAINBOW

Chang-Hyi Lee[1] and Jeong-Soo Kim[2]

[1] Samsung Advanced Institute of Technology,
chang@saitgw.sait.samsung.co.kr
[2] integer@saitgw.sait.samsung.co.kr

Abstract. In this paper we propose a new block cipher called RAINBOW. It has the substitution and permutation network structure which is similar to the block ciphers Square[3] and SHARK[9]. The motivation for our algorithm development came out from the hope that, first, the block cipher has the structure of parallelism, second, it has a self-reciprocal structure, i.e. the whole processing architecture(body) for encryption and decryption is the same one except feeding its round keys like as the case of Feistel network structure, and third, its round encryption involves nonlinear(over $GF(2^m)$) and efficient key dependent(active) diffusion layer. Our C-code implementation of RAINBOW is available that runs at about 9-10 MBytes/sec on a 133 MHz Pentium PC. We also presented the resistance against various(typical) cryptanalyses.

1 Introduction

The block cipher RAINBOW has the SPN structure and it processes on 128-bit data block with 128-bit seed key. But it has some extendible variants of key and encryption block sizes. Most of the block ciphers with SPN structure has two types of layers, so-called *diffusion layer* and *substitution layer* in each round. The former has two sub-layers, round key active part(in the most cases this is described simply by round-key XOR with input data) and bit-by-bit permutation part. The latter has some substitution boxes instead of key action. Our main goal for the construction of RAINBOW was focused on the very second part of diffusion layer and the self-reciprocal substitution layer. Concretely speaking we made the former be key active permutation and made the latter, substitution layer, be self-invertible and with flavored simple permutation, not using the algebraic inversion function. And all these processes were designed to have parallelism structure. Through those, RAINBOW would be expected to be highly resistant against the present various cryptanalyses and to be efficiently implementable.

2 description of RAINBOW

One round encryption of RAINBOW consists of three layers(see Fig.1), first two layers relate to the round key active diffusion and the other one relates to the nonlinear substitution with simple permutation. And the r times iteration of such layers contributes to the RAINBOW's encryption/decryption processes. The layers are named as $Green$($\mathcal{G}$-function), $Blue$($\mathcal{B}$-function), and Red($\mathcal{R}$-function), respectively.

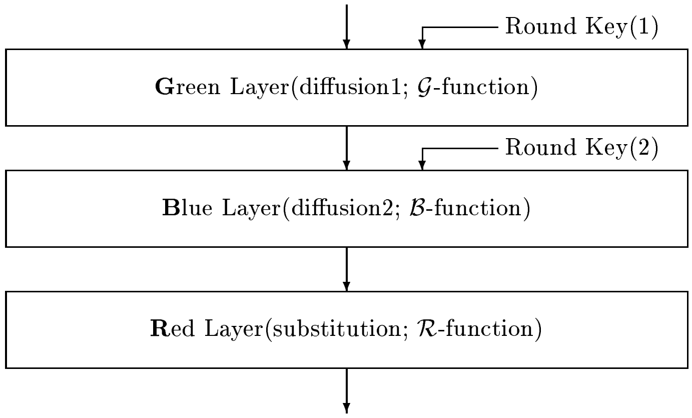

Fig. 1. One Round Encryption for RAINBOW

The detailed description for the above components of round encryption is presented in the following subsections.

2.1 $\mathcal{G}$-function

Let $X = [X_3, X_2, X_1, X_0]$ be a 128-bit round input data block, where X_i's are 32-bit subblocks and let $K = [K_3, K_2, K_1, K_0]$ be a 128-bit round key. Then the $\mathcal{G}_K$-function defined by K is simply described by the linear mapping that:

$$\mathcal{G}_K(X) = X \oplus K = [X_3 \oplus K_3, X_2 \oplus K_2, X_1 \oplus K_1, X_0 \oplus K_0],$$

where $\oplus$ denotes the bit by bit XOR operation. Of course this function is a self invertible function for a given K, i.e. $\mathcal{G}_K \circ \mathcal{G}_K(X) = X$.

2.2 $\mathcal{B}$-function

Let X and K be the same in the above. Then also this function has a simple description like as:

$$\mathcal{B}_K(X) = [\bar{X}_3, \bar{X}_2, \bar{X}_1, \bar{X}_0],$$

$$\bar{X}_0 = \bigoplus_{i=0}^{3}(X_i \wedge K_i)$$

$$\bar{X}_1 = \bigoplus_{i=0}^{3}(X_i \wedge K_{i+1})$$

$$\bar{X}_2 = \bigoplus_{i=0}^{3}(X_i \wedge K_{i+2})$$
$$\bar{X}_3 = \bigoplus_{i=0}^{3}(X_i \wedge K_{i+3})$$

where $\wedge$ denotes bit by bit logic AND operation and the subscripts are modulo-4 numbers. The present $\mathcal{B}_K$-function generated by K is not invertible in general. But we can give a specific condition for K to make the function $\mathcal{B}_K(*)$ into a self invertible function as one sees in the following theorem.

Theorem 1. *For a given* $K = [K_3, K_2, K_1, K_0]$ *such that*

$$\bigoplus_{i=0}^{3} K_i = (1, 1, \ldots, 1) := 1^{32},$$

the function $\mathcal{B}_K$ *defined by* K *as described in the above is a self invertible function.*

Proof. We have to show that

$$\mathcal{B}_K \circ \mathcal{B}_K(X) = X \tag{1}$$

Expanding the equation 2.1 and considering that $K_i \wedge K_i = K_i \wedge \bar{1} = K_i$, we know that it is sufficient to show that for each $i \neq j \in \{0, 1, 2, 3\}$,

$$\bigoplus_{k=0}^{3}(K_{i+k} \wedge K_{j+k}) = (0, 0, \ldots, 0) := 0^{32}, \tag{2}$$

where also the plus operator in the subscripts are modulo-4 addition. But here we show that only for the case of $j = i + 1$, the other cases also can be shown in the similar way.

$$\begin{aligned}
&(K_i \wedge K_{i+1}) \oplus (K_{i+1} \wedge K_{i+2}) \oplus (K_{i+2} \wedge K_{i+3}) \oplus (K_{i+3} \wedge K_i) \\
&= \{K_i \wedge (K_{i+1} \oplus K_{i+3})\} \oplus \{K_{i+2} \wedge (K_{i+1} \oplus K_{i+3})\} \\
&= (K_{i+1} \oplus K_{i+3}) \wedge (K_i \oplus K_{i+2}) \\
&= (K_{i+1} \oplus K_{i+3}) \wedge (K_{i+1} \oplus K_{i+3} \oplus 1^{32}) \\
&= \bar{0}
\end{aligned}$$

□

This function $\mathcal{B}$ plays its role for the key dependent data block diffusion just before the action of nonlinear self invertible function, $\mathcal{R}$ described in the next subsection.

2.3 $\mathcal{R}$-function

In the construction of $\mathcal{R}$, we also give the self invertible structure for the nonlinear function $\mathcal{R}$. Now let f be a bijective nonlinear function of $GF(2^8)$ to $GF(2^8)$. It is preferable to choose f so that it and its inverse f^{-1} have low differential uniformity and low linearity and so that have low complexity for their hardware implementation. In the below subsection we illustrate our choice for f. The function $\mathcal{R}$ consists of three types of component functions, $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, which all are defined in the following as functions of the sub-data block, $GF(2^{32})$ to $GF(2^{32})$. Let $x = (x_3, x_2, x_1, x_0)$, $x_i \in GF(2^8)$ and so $x \in GF(2^{32})$. Let's define

$$\begin{aligned}
\mathcal{P}_1(x) &= \begin{pmatrix} 0 & f & 0 & 0 \\ f^{-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & f \\ 0 & 0 & f^{-1} & 0 \end{pmatrix} \begin{pmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \end{pmatrix} \\
&= (f(x_2), f^{-1}(x_3), f(x_0), f^{-1}(x_1))^T && (3) \\
\mathcal{P}_2(x) &= \begin{pmatrix} 0 & 0 & f & 0 \\ 0 & 0 & 0 & f \\ f^{-1} & 0 & 0 & 0 \\ 0 & f^{-1} & 0 & 0 \end{pmatrix} \begin{pmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \end{pmatrix} \\
&= (f(x_1), f(x_0), f^{-1}(x_3), f^{-1}(x_2))^T && (4) \\
\mathcal{P}_3(x) &= \begin{pmatrix} 0 & 0 & 0 & f \\ 0 & 0 & f & 0 \\ 0 & f^{-1} & 0 & 0 \\ f^{-1} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \end{pmatrix} \\
&= (f(x_0), f(x_1), f^{-1}(x_2), f^{-1}(x_3))^T && (5)
\end{aligned}$$

Finally we define the function $\mathcal{R}$ of $GF(2^{128})$ to $GF(2^{128})$ as:

$$\begin{aligned}
\mathcal{R} &= [\mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_2, \mathcal{P}_1] \\
\mathcal{R}[X] &= \mathcal{R}[X_3, X_2, X_1, X_0] = [\mathcal{P}_2(X_3), \mathcal{P}_3(X_2), \mathcal{P}_2(X_1), \mathcal{P}_1(X_0)]
\end{aligned}$$

Theorem 2. *The above nonlinear function $\mathcal{R}$ is self invertible.*

Proof. It is sufficient to show that each $\mathcal{P}_i$ is self invertible. We only to show that for the case of $\mathcal{P}_1$ and the other cases can be shown in the similar manner.

$$\begin{aligned}
&\mathcal{P}_1 \circ \mathcal{P}_1(x_3, x_2, x_1, x_0) \\
&= \mathcal{P}_1(f(x_2), f^{-1}(x_3), f(x_0), f^{-1}(x_1)) \\
&= (f(f^{-1}(x_3)), f^{-1}(f(x_2)), f(f^{-1}(x_1)), f^{-1}(f(x_0))) \\
&= (x_3, x_2, x_1, x_0)
\end{aligned}$$

□

As said in the beginning of this section, for example, 7-round encryption and decryption of RAINBOW is described as:

$$\begin{aligned} \mathbf{RainbowEnc}_K(X) &= \mathcal{B} \circ \mathcal{G} \circ F \circ F \circ F \circ F \circ F \circ F \circ F(X), \qquad (6) \\ &\rightarrow F = \mathcal{R} \circ \mathcal{B} \circ \mathcal{G}, \end{aligned}$$

where the functions $\mathcal{G}$'s and $\mathcal{B}$'s are defined by its round keys derived from K by the key scheduling mentioned in the next section. Further in which the whole self-reciprocal(self-invertible) property of RAINBOW except the round key feeding can be achieved. Those functions $\mathcal{B}$ and $\mathcal{R}$ play very critical roles for RAINBOW.

2.4 Our Choice of f

We need to establish the bijective nonlinear function f to satisfy that if possible, it has low differential uniformity and low linearity, further to facilitate its hardware implementation. But this is a very hard problem. We took a easy approach and groped for comparatively effective f in the set of algebraic functions of $GF(2^8)$ to $GF(2^8)$. Concretely describing, let

$$f(x) = x^{37}, \quad x \in GF(2^8) \qquad (7)$$

$$f^{-1}(x) = x^{193}, \quad x \in GF(2^8). \qquad (8)$$

Then

$$\text{Differential Probability } DC(f) = 2^{-5.4}$$

$$\text{squared Linear Probability } LC^2(f) = 2^{-4},$$

further the nonlinear order of both f and f^{-1} is 3, since $x^{37} = x^{2^5+2^2+1}$ and $x^{193} = x^{2^7+2^6+1}$. The concrete implementation aspects of these functions is dealt in the section 5.

2.5 Key Scheduling

There are needed $2(R+1)$ round key blocks of which size is 16 bytes(128-bits) long for the RAINBOW's encryption or decryption process, that is, in each layer of equation 2.6, different types of key blocks are fed into those diffusion functions, $\mathcal{G}$ and $\mathcal{B}$. Here is presented the key scheduling algorithm for RAINBOW in pseudo-code. As one sees in the following algorithm, there are used only the two simple operations, bit-by-bit XOR and 4 types of rotations. Let $S = [S_3, S_2, S_1, S_0]$ be a 128-bit seed key and S_i's are its 32-bit subblock. In the following pseudo-code, the content 0xb7e15163 was chosen from the constant, so called *golden ratio* $\phi = 1.618033988749\ldots$, by the first 32 bits following the decimal point.

INPUT : seed key block; $S = [S_3, S_2, S_1, S_0]$, $|S_i| = 32 - bit$
OUTPUT(1) : encryption round key blocks; $K_e[2R+1][4]$

OUTPUT(2) : decryption round key blocks; $K_d[2R+1][4]$
$K_e[0] \leftarrow S$ /*that is, $K_e[0][j] \leftarrow S[j] = S_j,\ j = 0,1,2,3$ */
$i = 1$;
While $i < 2R+2$ do the following ($\leftarrow R$: number of rounds)
{ /*We encourage one to use $R \geq 7$*/
$K_e[i] \leftarrow K_e[i-1]$;
$K_e[i][0] \leftarrow (K_e[i][0] >>> 3) \oplus (K_e[i][1] >>> 5) \oplus (K_e[i][2] >>> 7)$
$\oplus(K_e[i][3] >>> 11) \oplus$ 0xb7e15163;
$K_e[i][1] \leftarrow (K_e[i][0] >>> 5) \oplus (K_e[i][1] >>> 7) \oplus (K_e[i][2] >>> 11)$
$\oplus(K_e[i][3] >>> 3) \oplus$ 0xb7e15163;
$K_e[i][2] \leftarrow (K_e[i][0] >>> 7) \oplus (K_e[i][1] >>> 11) \oplus (K_e[i][2] >>> 3)$
$\oplus(K_e[i][3] >>> 5) \oplus$ 0xb7e15163;
$K_e[i][3] \leftarrow (K_e[i][0] >>> 11) \oplus (K_e[i][1] >>> 3) \oplus (K_e[i][2] >>> 5)$
$\oplus K_e[i][3] >>> 7) \oplus$ 0xb7e15163;
$i = i+1$;
}
$j = 0$;
While $j < R+1$ do the following
{
$i = 2*j+1$;
$K_e[i][0] \leftarrow K_e[i][1] \oplus K_e[i][2] \oplus K_e[i][3] \oplus$ 0xffffffff;
$K_d[2*(R+1)-i] \leftarrow K_e[i]$;
$j = j+1$;
}
$j = 0$;
While $j < R+1$ do the following
{
$i = 2*j;\ \ i_1 = 2*(R-j);\ \ i_2 = i_1+1$;
$K_d[i][0] \leftarrow (K_e[i_1][0] \wedge K_e[i_2][0]) \oplus (K_e[i_1][1] \wedge K_e[i_2][1]) \oplus$
$(K_e[i_1][2] \wedge K_e[i_2][2]) \oplus (K_e[i_1][3] \wedge K_e[i_2][3])$;
$K_d[i][1] \leftarrow (K_e[i_1][0] \wedge K_e[i_2][1]) \oplus (K_e[i_1][1] \wedge K_e[i_2][2]) \oplus$
$(K_e[i_1][2] \wedge K_e[i_2][3]) \oplus (K_e[i_1][3] \wedge K_e[i_2][0])$;
$K_d[i][2] \leftarrow (K_e[i_1][0] \wedge K_e[i_2][2]) \oplus (K_e[i_1][1] \wedge K_e[i_2][3]) \oplus$
$(K_e[i_1][2] \wedge K_e[i_2][0]) \oplus (K_e[i_1][3] \wedge K_e[i_2][1])$;
$K_d[i][3] \leftarrow (K_e[i_1][0] \wedge K_e[i_2][3]) \oplus (K_e[i_1][1] \wedge K_e[i_2][0]) \oplus$
$(K_e[i_1][2] \wedge K_e[i_2][1]) \oplus (K_e[i_1][3] \wedge K_e[i_2][2])$;
$j = j+1$;
}

As one sees, this key scheduling algorithm is very simple and was designed for each bit values of the seed key to affect all the round key blocks and their sub blocks. And we can decrypt cipher texts CT using the same processing flow(equation 2.6) as in the the encrypting process with the decryption keys

$K_d[i]$'s evoluted in the above algorithm, i.e.

$$\mathbf{RainbowDec}_{K_d}(CT) = \mathcal{B} \circ \mathcal{G} \circ F \circ F \circ F \circ F \circ F \circ F \circ F(CT), \qquad (9)$$
$$\rightarrow F = \mathcal{R} \circ \mathcal{B} \circ \mathcal{G}.$$

Its validity was shown in the following theorem.

Theorem 3. *Let*

$$\begin{aligned} &K = [K_3, K_2, K_1, K_0] \\ &S = [S_3, S_2, S_1, S_0], \ \bigoplus_{i=0}^{3} S_i = (1, 1, \ldots, 1) \\ &\bar{K} = [\bar{K}_3, \bar{K}_2, \bar{K}_1, \bar{K}_0] \\ &\bar{K}_i = (K_0 \wedge S_i) \oplus (K_1 \wedge S_{i+1}) \oplus (K_2 \wedge S_{i+2}) \oplus (K_3 \wedge S_{i+3}) \\ &\qquad i = 0, 1, 2, 3, \ \textit{subscrypts are modulo-4 numbers} \end{aligned}$$

Then the following holds:

$$\mathcal{G}_K \circ \mathcal{B}_S = \mathcal{B}_S \circ \mathcal{G}_{\bar{K}}$$

Proof. Let $X = [X_3, X_2, X_1, X_0]$, $Y = [Y_3, Y_2, Y_1, Y_0]$ and $\mathcal{G}_K \circ \mathcal{B}_S(X) = Y$. Let's focus on the i^{th} subblock Y_i of Y. Noting that $\bigoplus_{j=0}^{4}(\bar{K}_j \wedge S_{i+j}) = K_i$ which is deduced from the proof of theorem2.1 we get the following

$$\begin{aligned} Y_i &= (X_0 \wedge S_i) \oplus (X_1 \wedge S_{i+1}) \oplus (X_2 \wedge S_{i+2}) \oplus (X_3 \wedge S_{i+3}) \oplus K_i \\ &= (X_0 \wedge S_i) \oplus (X_1 \wedge S_{i+1}) \oplus (X_2 \wedge S_{i+2}) \oplus (X_3 \wedge S_{i+3}) \oplus \bigoplus_{j=0}^{4}(\bar{K}_j \wedge S_{i+j}) \\ &= ((X_0 \oplus \bar{K}_0) \wedge S_i) \oplus ((X_1 \oplus \bar{K}_1) \wedge S_{i+1}) \oplus \\ &\quad ((X_2 \oplus \bar{K}_2) \wedge S_{i+2}) \oplus ((X_3 \oplus \bar{K}_3) \wedge S_{i+3}) \\ &= i^{th} \text{ component of } \mathcal{B}_S \circ \mathcal{G}_{\bar{K}}(X) \end{aligned}$$

This completes the proof. □

In this section we showed that RAINBOW has the structure that just its encryption process can be used as its decryption process(that is, following the same order of the functions $\mathcal{G}$, $\mathcal{B}$ and $\mathcal{R}$ with such above evoluted round keys. This efficiency reduces the size of software implemented code and the hardware implemented chip area.

3 Security of RAINBOW

3.1 DC and LC attacks

The practical analyses of differential cryptanalysis[4] and linear cryptanalysis[8] are heavily dependent on whether we can find some efficient differential characteristic or linear characteristic to make their attack feasible. But in our case of

RAINBOW the blue function layer of function $\mathcal{B}$ makes the effort to trace such efficient characteristic be meaningless, since its key dependent masking(it is assumed that all round keys are randomly chosen) makes it impossible for one to control the input differences to each S-box for DC or the input masking vectors to each S-box for LC. Even if, however, in some rough sense of average concept(for the randomly chosen round keys) there are 29 active S-boxes in every 3 rounds, the 7-round encryption has at least 58 active S-boxes, and so the differential characteristic and squared linear characteristic are $(2^{-5.4})^{58} = 2^{-313.2}$ and $(2^{-4})^{58} = 2^{-232}$ respectively. These are negligible amounts and these attacks are not available for RAINBOW.

3.2 Higher Order Differential Attack

It was shown in [2] that if an iterated cipher has the polynomial degree d of the ciphertext bits of the round next to the last as a function of the plaintext bits, the higher order differential attack[1] requires 2^{d+1} chosen plaintexts which will successfully recover the b bits of the last round key with average time complexity 2^{b+d}. But, as mentioned in section 2.4, the nonlinear order of one round is at least 3, and so the nonlinear order of the output bits of RAINBOW just after five rounds is 3^5. This amount exceeds the attack available maximum value 127 and the cipher would be expected to be secure again higher order differential attack.

3.3 Interpolation Attack

The interpolation attack[2] is considerable only when the whole encryption process can be described in some proper algebraic functions of data blocks and key blocks in a proper $GF(2^m)$. But, also for this case, the key dependent function $\mathcal{B}$ blocks such conversions into algebraic form over $GF(2^m)$ except over $GF(2)$. Even for the case of $GF(2)$ it's not available, since $\#|GF(2)| = 2$ is exorbitantly small for the number of plain/ciphertext pairs to be required in this attack. Hence RAINBOW would be expected to escape this attack.

4 Variants of Key and Encryption Block Sizes

4.1 Variants of Key Size

Let the seed key $S = [S_{n-1}, \ldots, S_1, S_0]$ consists of n word blocks, where 1 word block denotes the 32-bit block, i.e. the bit-size of S be $32n$, $4 \leq n \leq 8$. Then, in the key scheduling algorithm(pseudo-code) of section2.5, we use the first 4 words(128-bits) of S as the initial 128-bit seed key value as in the code and just before the first While-loop of the code we insert the following new code lines;

$K_e[1] \leftarrow K_e[0]$;
for $(j = 0; j < n - 4; j++)$ $K_e[1][j] \leftarrow K_e[1][j] \oplus S_{4+j}$;
$K_e[1][0] \leftarrow (K_e[1][0] >>> 3) \oplus (K_e[1][1] >>> 5) \oplus (K_e[i][2] >>> 7)$

$$
\begin{aligned}
&\quad\oplus(K_e[1][3] >\!>\!> 11) \oplus \mathrm{0xb7e15163};\\
K_e[1][1] &\leftarrow (K_e[1][0] >\!>\!> 5) \oplus (K_e[1][1] >\!>\!> 7)\\
&\quad\oplus(K_e[1][2] >\!>\!> 11) \oplus (K_e[i][3] >\!>\!> 3) \oplus \mathrm{0xb7e15163};\\
K_e[1][2] &\leftarrow (K_e[1][0] >\!>\!> 7) \oplus (K_e[1][1] >\!>\!> 11)\\
&\quad\oplus(K_e[1][2] >\!>\!> 3) \oplus (K_e[1][3] >\!>\!> 5) \oplus \mathrm{0xb7e15163};\\
K_e[1][3] &\leftarrow (K_e[1][0] >\!>\!> 11) \oplus (K_e[1][1] >\!>\!> 3)\\
&\quad\oplus(K_e[1][2] >\!>\!> 5) \oplus K_e[1][3] >\!>\!> 7) \oplus \mathrm{0xb7e15163};\\
i &= 2;
\end{aligned}
$$

Through this additional code the key scheduling algorithm gets to be available for several variants of seed key size with word by word increment between 128-bits and 256-bits.

4.2 Variants of Encryption Block Size

We think it is sufficient that one modifies the Blue-function $\mathcal{B}_K$ and the Red-function $\mathcal{R}_K$ to be available for the variants of sizes of input data X and round key K. This can be simply settled by the following considerations;

Let $|K| = |X| = N$ bits, $K = [K_3, K_2, K_1, K_0]$, $|K_i| = \frac{N}{4}$ bits, and $X = [X_3, X_2, X_1, X_0]$, $|X_i| = \frac{N}{4}$ bits. And we assume that, by a similar key scheduling algorithm with the present one, K was scheduled as $\bigoplus_{i=0}^{3} K_i = 1^{N/4}$. Then the resulting value of $\mathcal{B}_K(X) := \bar{X} = [\bar{X}_3, \bar{X}_2, \bar{X}_1, \bar{X}_0]$ is defined as the following by the same manner in the case of 128-bits:

$$
\begin{aligned}
\mathcal{B}_K(X) &= [\bar{X}_3, \bar{X}_2, \bar{X}_1, \bar{X}_0],\\
\bar{X}_0 &= \bigoplus_{i=0}^{3}(X_i \wedge K_i)\\
\bar{X}_1 &= \bigoplus_{i=0}^{3}(X_i \wedge K_{i+1})\\
\bar{X}_2 &= \bigoplus_{i=0}^{3}(X_i \wedge K_{i+2})\\
\bar{X}_3 &= \bigoplus_{i=0}^{3}(X_i \wedge K_{i+3}),
\end{aligned}
$$

and, if we let $\bar{X} = [Y_{n-1}, \ldots, Y_1, Y_0]$ and $|Y_i| = 32$ bits, the resulting value of $\mathcal{R}(\bar{X}) := Y$ is defined by repeating the functions $\mathcal{P}_i$'s of $GF(2^{32})$ to $GF(2^{32})$ constructed in section2.3 in the order of $\mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_2, \mathcal{P}_1$ from the right most word, as the following:

$$
Y = [\ldots, \mathcal{P}_2(Y_5), \mathcal{P}_1(Y_4), \mathcal{P}_2(Y_3), \mathcal{P}_3(Y_2), \mathcal{P}_2(Y_1), \mathcal{P}_1(Y_0)]
$$

Through this extensibility Rainbow gets to be available for several variants of block size with word by word increment.

5 Performance and Implementation Facilities

Our reference C-code implementation of RAINBOW runs at 9-10 MBytes/sec on a 400MHz Pentium PC with Window95 operating system with 32MByte RAM. But using effectively its parallelism it would be expected to be more optimized.

The first table in the following present the ECB-mode encryption speed for three types of key-block combinations. And those figures in milli-seconds were obtained with a raw resolution around 5ms and from these figures we can computed the RAINBOW's encryption/decryption times in clock cycles as in the following second table(note 1ms=133000 clocks).

Key/Block	Init.Cipher	Encrypt (1 Mbytes)	Decrypt (1 Mbytes)	Key Init. (1024times)
128/128	32 ms	390 ms	380 ms	50.0 ms
192/128	32 ms	390 ms	380 ms	51.2 ms
256/128	32 ms	380 ms	390 ms	51.2 ms

Key/Block	Init.Cipher	Encrypt (1 Block)	Decrypt (1 Block)	Init.1Key	Key Change
128/128	4256000 clks	791 clks	771 clks	6494 clks	0
192/128	4256000 clks	791 clks	771 clks	6650 clks	0
256/128	4256000 clks	771 clks	791 clks	6650 clks	0

As mentioned in the section2.4, we chose the S-box functions f and f^{-1} as x^{37} and x^{193} respectively and we make the RAINBOW's S-box, **RED**, by using the normal basis generated by the root of

$$p(x) = x^8 + x^7 + x^5 + x^3 + 1.$$

The concrete entries of the table, RED, were represented in appendix-A at the end of this manuscript. Here are presented two considerations on the hardware design of RAINBOW, one is based on the table(S-box) look-up method and another is based on the wholly circuit designed(for the functions f and f^{-1}) one, in which the explanation on the facilities of our choice for f and f^{-1} is presented.

5.1 Table Look-up Case

In this case, the S-box is stored in the memory devices, such as ROM, RAM, etc.. And so, the circuit design costs required for the implementations of the three functions, $\mathcal{G}$, $\mathcal{B}$, $\mathcal{R}$ are the followings;

- in $\mathcal{G}$-layer : 128 XOR gates and one gate delay;
- in $\mathcal{B}$-layer : 128×4 AND gates, $(32 \times 3) \times 4$ XOR gates and 3 gate delays;

- in $\mathcal{R}$-layer : 16 table look-up's;

So, in total, the implementation of the 7-round RAINBOW, in this case, requires $8 \times (128 + 384) = 4096 = 4K$ XOR gates, $8 \times 512 = 4096 = 4K$ AND gates, $16 \times 7 = 112$ table look-up's, and $4 \times 8 = 32$ gate delays and 7 table look-up delays.

5.2 Circuit Design for S-box

As previously described, $f = x^{37} = x^{2^5+2^2+1}$ and $f^{-1} = x^{193} = x^{2^7+2^6+1}$. If we use the normal basis of $GF(2^8)$ over $GF(2)$ to design the functions, their implementation costs are same and they can be designed by the two sequential field multiplications, since the terms, x^{2^i}'s can be achieved by only the right(or left) bit-by-bit rotations of x over the normal based representation. As well known, there is no optimal normal basis of $GF(2^8)$ over $GF(2)$ to facilitate the h/w design of field multiplication and so we investigated all the N-polynomials([7][6], this means the irreducible polynomials of which root generate a normal basis) of degree, 8, over $GF(2)$ to find such N-polynomial as to generate the second best normal basis of which complexity minimum.

There are just 16 N-polynomials of degree 8 over $GF(2)$ and we found that there exists only one N-polynomial of which multiplication table(matrix)'s complexity reaches the the minimum value, 21. The N-polynomial $p(x)$ is

$$p(x) = x^8 + x^7 + x^5 + x^3 + 1. \tag{10}$$

If we set $p(x)$ as the defining polynomial for $GF(2^8)$ over $GF(2)$, then its generating normal basis has the first field multiplication table(matrix), T_0(see [7]), is represented as

$$\begin{pmatrix} 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&1 \\ 0&0&0&0&1&0&1&0 \\ 1&0&0&0&0&1&1&1 \\ 0&0&1&0&0&1&0&0 \\ 0&0&0&1&1&0&0&0 \\ 0&1&1&1&0&0&0&1 \\ 0&1&0&1&0&0&1&1 \end{pmatrix}.$$

This table tells us the design cost of one field multiplication. From this we know that it, in total, requires $8 \times 29 = 232$ AND gates, $8 \times 20 = 160$ XOR gates, and 7 gate delay, in the case of bit-parallel design. Therefore the design of f requires two times of those complexities and so the whole $\mathcal{R}$-layer's needs $16 \times 464 = 7424 \approx 7K$ AND gates, $16 \times 320 = 5120 \approx 5K$ XOR gates, and 14 gate delays.

Collectively, the whole circuit design for the 7-round RAINBOW requires $53K$ AND gates, $39K$ XOR gates, and 130 gate delays. We think this is very effective and the performance induced from this seems to make RAINBOW be applicable to such areas of ATM, HDTV, B-ISDN, and Satellite. Further more the RAINBOW was designed only by using bit-by-bit XOR and logic AND operations and using 8×8 S-boxes which have simple and effective implementation characteristics in its VLSI circuit design structure. So it would be effectively applicable to 8-bit processors, too.

6 Conclusion

In this paper, we described the structure of the newly proposed block cipher, Rainbow and considered its effective characteristics and its design rational. As said previously, Rainbow was designed to satisfy our goal, i.e. the key active diffusion so that its layer, $\mathcal{B}$-layer, cut off such hazards from the typical and conventional attacks, differential cryptanalysis, linear cryptanalysis, higher order differential attack, and interpolation attack. Its low-cost VLSI design structure make it possible to be applicable to those application areas, ATM, HDTV, B-ISDN, Satellite and such as Smart Cards using 8-bit processor. The referencing C-code implementation of Rainbow which is compiled by visual C++ runs at 9-10 MBytes/s on Pentium 400MHz with window95 operating system.

References

1. X.J. Lai, *Higher order derivatives and differential cryptanalysis*, In R. Blahut, editor, *Communication and Cryptography, Two Sides of one tapestry.* Kluwer Academic Publishers, 1994. ISBN 0-7923-9469-0.
2. T. Jakobsen and L.R. Knudsen. *The Interpolation Attack on Block Ciphers.* Advances in Cryptology - Fast Software Encryption'97, Lecture Notes in Computer Science , Springer-Verlag pp.28-40, 1996.
3. J. Daemen, L. Knudsen and V. Rijmen, *The Block Cipher Square* Advances in Cryptology - Fast Software Encryption'97, Lecture Notes in Computer Science , Springer-Verlag pp.149-171, 1997.
4. E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard.* Springer-Verlag, 1993.
5. K. Nyberg. *S-Boxes and Round Functions with Controllable Linearity and Differential Uniformity.* Advances in Cryptology - Fast Software Encryption'94, Lecture Notes in Computer Science 1008, Springer-Verlag pp.111-130, 1994.
6. R. Lidl, H. Niederreiter, *Finite Fields.* Encyclopedia of Math. and its Application, #20.
7. Alfred J. Menezes, *Applications of Finite Fields*, Kluwer Academic Publishers, pp.83, 1993.
8. M. Matsui. *Linear Cryptanalysis Method for DES cipher*, Advances in Cryptology - EUROCRYPT'93, Lecture Notes in Computer Science 765, Springer-Verlag, pp.386-397, 1994
9. V. Rijmen, J. Daemen et al., *The Cipher SHARK*, Fast Software Encryption, LNCS 1039, Springer-Verlag, pp.99-112, 1996.

A Appendix

A.1 The S-box RED of Rainbow

Here are listed the tables of f and f^{-1} and the whole table of **RED**[512] is given by pasting these two tables in this order.

- Table of f.

0x00	0x0e	0x1c	0x08	0x38	0xe5	0x10	0x19
0x70	0x16	0xcb	0x42	0x20	0xe7	0x32	0xd4
0xe0	0xcc	0x2c	0x65	0x97	0xa7	0x84	0x1f
0x40	0x67	0xcf	0x78	0x64	0x2d	0xa9	0xbe
0xc1	0xc2	0x99	0xec	0x58	0xd1	0xca	0xfb
0x2f	0x8e	0x4f	0x6d	0x09	0x50	0x3e	0x2a
0x80	0x56	0xce	0x11	0x9f	0x0c	0xf0	0xa4
0xc8	0xdf	0x5a	0xb1	0x53	0x73	0x7d	0x6f
0x83	0x79	0x85	0xf9	0x33	0xe9	0xd9	0x4b
0xb0	0x74	0xa3	0x14	0x95	0x03	0xf7	0xdc
0x5e	0x7a	0x1d	0xc0	0x9e	0x55	0xda	0x26
0x12	0x6b	0xa0	0xd5	0x7c	0x98	0x54	0x72
0x01	0x48	0xac	0x0f	0x9d	0xad	0x22	0x36
0x3f	0x82	0x18	0xba	0xe1	0x57	0x49	0x2e
0x91	0xf1	0xbf	0x4a	0xb4	0x62	0x63	0xee
0xa6	0x51	0xe6	0x71	0xfa	0xc9	0xde	0x43
0x07	0x04	0xf2	0x8c	0x0b	0x21	0xf3	0x6a
0x66	0xb2	0xd3	0x8f	0xb3	0x3c	0x96	0x5f
0x61	0x76	0xe8	0xfd	0x47	0xb6	0x28	0x15
0x2b	0x88	0x06	0x52	0xef	0xd8	0xb9	0xb7
0xbc	0xfc	0xf4	0xa5	0x3a	0x0a	0x81	0x6e
0x3d	0x60	0xaa	0x13	0xb5	0xea	0x4c	0x39
0x24	0x87	0xd6	0x1b	0x41	0x5d	0xab	0x17
0xf8	0x25	0x31	0x77	0xa8	0xb8	0xe4	0xa1
0x02	0x46	0x90	0x35	0x59	0xc7	0x1e	0xaf
0x3b	0xfe	0x5b	0x8a	0x44	0x29	0x6c	0xdb
0x7e	0xd2	0x05	0x37	0x30	0x89	0x75	0x9c
0xc3	0x8d	0xae	0x8b	0x92	0xbb	0x5c	0xd0
0x23	0x9a	0xe3	0xd7	0x7f	0x45	0x94	0xed
0x69	0x9b	0xc4	0x4e	0xc6	0xc5	0xdd	0x68
0x4d	0xeb	0xa2	0xf6	0xcd	0x27	0xe2	0x34
0xf5	0x7b	0x93	0x1a	0xbd	0x0d	0x86	0xff

• Table of f^{-1}.

$0x00$	$0x60$	$0xc0$	$0x4d$	$0x81$	$0xd2$	$0x9a$	$0x80$
$0x03$	$0x2c$	$0xa5$	$0x84$	$0x35$	$0xfd$	$0x01$	$0x63$
$0x06$	$0x33$	$0x58$	$0xab$	$0x4b$	$0x97$	$0x09$	$0xb7$
$0x6a$	$0x07$	$0xfb$	$0xb3$	$0x02$	$0x52$	$0xc6$	$0x17$
$0x0c$	$0x85$	$0x66$	$0xe0$	$0xb0$	$0xb9$	$0x57$	$0xf5$
$0x96$	$0xcd$	$0x2f$	$0x98$	$0x12$	$0x1d$	$0x6f$	$0x28$
$0xd4$	$0xba$	$0x0e$	$0x44$	$0xf7$	$0xc3$	$0x67$	$0xd3$
$0x04$	$0xaf$	$0xa4$	$0xc8$	$0x8d$	$0xa8$	$0x2e$	$0x68$
$0x18$	$0xb4$	$0x0b$	$0x7f$	$0xcc$	$0xe5$	$0xc1$	$0x94$
$0x61$	$0x6e$	$0x73$	$0x47$	$0xae$	$0xf0$	$0xeb$	$0x2a$
$0x2d$	$0x79$	$0x9b$	$0x3c$	$0x5e$	$0x55$	$0x31$	$0x6d$
$0x24$	$0xc4$	$0x3a$	$0xca$	$0xde$	$0xb5$	$0x50$	$0x8f$
$0xa9$	$0x90$	$0x75$	$0x76$	$0x1c$	$0x13$	$0x88$	$0x19$
$0xef$	$0xe8$	$0x87$	$0x59$	$0xce$	$0x2b$	$0xa7$	$0x3f$
$0x08$	$0x7b$	$0x5f$	$0x3d$	$0x49$	$0xd6$	$0x91$	$0xbb$
$0x1b$	$0x41$	$0x51$	$0xf9$	$0x5c$	$0x3e$	$0xd0$	$0xe4$
$0x30$	$0xa6$	$0x69$	$0x40$	$0x16$	$0x42$	$0xfe$	$0xb1$
$0x99$	$0xd5$	$0xcb$	$0xdb$	$0x83$	$0xd9$	$0x29$	$0x8b$
$0xc2$	$0x70$	$0xdc$	$0xfa$	$0xe6$	$0x4c$	$0x8e$	$0x14$
$0x5d$	$0x22$	$0xe1$	$0xe9$	$0xd7$	$0x64$	$0x54$	$0x34$
$0x5a$	$0xbf$	$0xf2$	$0x4a$	$0x37$	$0xa3$	$0x78$	$0x15$
$0xbc$	$0x1e$	$0xaa$	$0xb6$	$0x62$	$0x65$	$0xda$	$0xc7$
$0x48$	$0x3b$	$0x89$	$0x8c$	$0x74$	$0xac$	$0x95$	$0x9f$
$0xbd$	$0x9e$	$0x6b$	$0xdd$	$0xa0$	$0xfc$	$0x1f$	$0x72$
$0x53$	$0x20$	$0x21$	$0xd8$	$0xea$	$0xed$	$0xec$	$0xc5$
$0x38$	$0x7d$	$0x26$	$0x0a$	$0x11$	$0xf4$	$0x32$	$0x1a$
$0xdf$	$0x25$	$0xd1$	$0x8a$	$0x0f$	$0x5b$	$0xb2$	$0xe3$
$0x9d$	$0x46$	$0x56$	$0xcf$	$0x4f$	$0xee$	$0x7e$	$0x39$
$0x10$	$0x6c$	$0xf6$	$0xe2$	$0xbe$	$0x05$	$0x7a$	$0x0d$
$0x92$	$0x45$	$0xad$	$0xf1$	$0x23$	$0xe7$	$0x77$	$0x9c$
$0x36$	$0x71$	$0x82$	$0x86$	$0xa2$	$0xf8$	$0xf3$	$0x4e$
$0xb8$	$0x43$	$0x7c$	$0x27$	$0xa1$	$0x93$	$0xc9$	$0xff$