Zodiac™ 1.0

Architecture and Specification

September 2000

Submitted by Crypto-Laboratory of SoftForum

This proposed cipher had been designed and cryptanalyzed by Chang-Hyi Lee and has been developed by crypto-laboratory of Softforum©. The Contributors are; Chang-Hyi Lee, Kyung-Hwa Jun, Min-Suk Jung, Sang-Bae Park, and Jong-Deok Kim

Design Concepts and Architectures

For Zodiac™ 1.0

PART-1: Design Concepts and Architecture for Zodiac™ 1.0

This is the first one of our two block cipher proposals named 'Zodiac' and 'Xenon'.

Version 1.0

1. Design Philosophy for Zodiac™ 1.0

1.1 Design Concepts for Zodiac™ 1.0

In the design of Zodiac, we tried to stress the following principles:

- **Simplicity:** We tried not to include ad-hoc design elements based on any obscure reason. We do not believe that more complicated design could give the more secure structure.
- **Provability:** As possible as we can do, we tried to design the cipher as it has a provably secure structure especially resistant to the well known typical attacks such as differential cryptanalysis, linear cryptanalysis and interpolation attack.
- **Accommodation and Performance:** Our cipher has been designed to get high performance both on general 32-bit CPU and general 8-bit microprocessor (e.g. smart cards).

1.1.1. Simplicity

In an exterior view of Zodiac, it adopts the Feistel-type network structure. The round function of Zodiac was simply constructed by only using exclusive-or operations and S-box look-ups. There were used two types of 8-by-8 S-boxes in the Zodiac. The S-boxes were constructed from an algebraic inversion and modular-exponentiation and these two different originations make it possible to have the resistance against the well-known interpolation attack. Although its round function was built in a very simple manner, the whole architecture should have a robust data scrambling function in the sense that all sub-block bytes of input data affect on all byte blocks of output data in balanced form.

The simple design principal for Zodiac made it possible to analyze its security in algebraically measured form. Moreover its simplicity makes it possible to be easily implemented both on general 32-bit CPU and general 8-bit microprocessor (smart cards).

1.1.2. Provability

As a little bit of mention in the above section, the round-function structure of Zodiac was made of a simple algebraic design excepting S-boxes, and the scrutinized cryptanalysis for its security level was described in this document and it suppresses such the DC-attack in total-differential sense. The detailed description is presented in the section 3.1. And the use of the two different types of S-boxes protects any byte-by-byte interpolation attack, since one of the S-boxes was made from an algebraic inversion in the Galois-field GF(2^8) and the another one was made from an modular exponentiation in the modular number system.

1.1.3. Accommodation and Performance

The round function of Zodiac was constructed to meet the following principles for the effectiveness of the implementation:

- **Byte-wise oriented operation:** All operations of the round function of Zodiac are byte-wise. But we tried to design its round function so that the integration of all the operations on byte-wisely divided data blocks minimizes the total number of computation delays. So we could achieve the effectiveness both of the implementation cost and of the performance on the general 32-bit CPU and the general 8-bit microprocessor (e.g. smart card).

- **All exclusive-OR operation and the Use of Non-algebraic S-box:** We tried to use only the simple bit-wise exclusive-OR (XOR) operation in the exterior networking of Zodiac and to use the non-algebraic substitution function (S-box) to escape any algebraic attacks. Note that the XOR operation is the best data mixing operation in the sense of the balanced affection.

In practice, Zodiac can be built with only 3200 gates with building S-boxes as ROM-tables and 66 gate-delays in the hardware implementation of the full 16-round encryption/decryption and it takes 330 clock-cycles for one-block (128-bits) encryption/decryption on Pentium machine.

1.2 The Key-Scheduling of Zodiac™

Zodiac-encryption/decryption is processed by 16 rounds iteration of the same one round-function and it processes on 128-bit data block. So it needs sixteen 64-bit round-keys and additionally needs two 64-bit keys for the input masking and output masking of the

processing data. Hence there are fewer key bits provided as input to the cipher than are needed by the cipher, and so the key-scheduling algorithm is required to expand the input key bits to adaptive to the total size of needed key bits in full encryption/decryption process. We tried to design the key-scheduling algorithm to meet the following characteristics:

- **Fast key-setup:** For the encryption and decryption for a massive data, the key-scheduling time is a minor element, but for the encryption/decryption processing on a short data block, the key-setup can overwhelm encryption speed. The key-schedule of Zodiac was built in reasonably efficient manner for both software and hardware implementation.

- **Key avalanche criterion:** The key schedule should be designed to make the cipher satisfy the *key avalanche criterion*; when a single key bit is changed, each ciphertext bit changes with a probability of 1/2.

- **Key scheduling can work "On the fly" with data processing:** The key-schedule of Zodiac can work "On the fly", deriving each block of round key, as it is needed, with as little required memory as possible.

- **All data bytes of seed-key affect on all round-key bytes:** Zodiac's round-key derivation scheme from the input seed-key was designed to meet such goal that all bytes of seed-key must affects on all round-key bytes and so make it hard to extract an additional key-information from any information leakage of sub-data of key-block.

- **One round-key leakage does not imply its previous round-key leakage:** All derived round-keys are affected by all the seed-key bits in the so called 'one-way' manner, and so it does not reveal and give out any related information between one round-key and another unless one finds out the original seed-key bytes.

- **There is no weak-key:** The key-schedule does not generate any duplicate round-key pairs, and so it does not occur that encryption-key string is the same of decryption-key string.

- **Supports variable key length:** The current key schedule can support three types of seed-key lengths, 128-bits, 192-bits, and 256-bits.

Of course, since Zodiac uses and requires the 16 seed-key bytes at minimum for the input data to its key-schedule, it suppresses key exhaustive-searching attack.

2. Architecture for Zodiac™ 1.0

2.1. Zodiac Building Blocks

In this section, we describe the building blocks of Zodiac as the full network structure, construction of S-boxes, round-function F, and key-scheduling algorithm.

2.1.1. Network Structure of Zodiac™

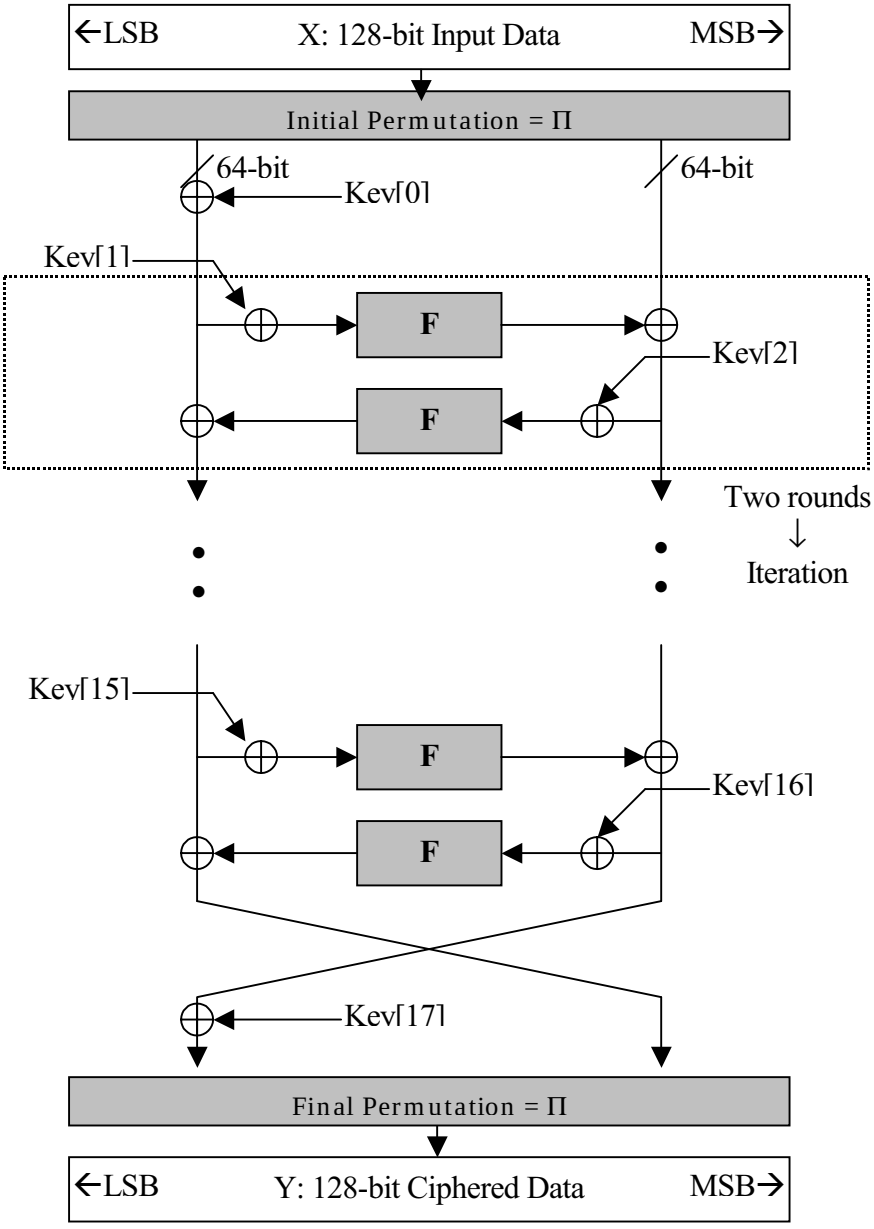
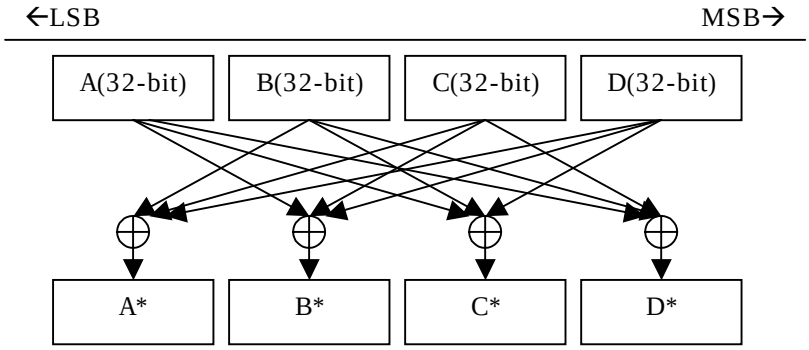


Figure 2.1: Feistel Network

Zodiac follows the conventional Feistel-network structure excepting the initial and final 64-bit data whitening by two 64-bit keys. These two additional data whitening does not break the symmetry of the Feistel-network structure and so the encryption and decryption processes are reciprocal to each other. The goal of the use of the final 64-bit whitening is to protect from revealing the final input data to the final round-function F. The detailed structure of the round-function F is presented in section 2.1.3.

The initial and final permutation, Π, appeared in Fig.2.1 is as the following:



Where, all words are of 32-bits-length. In another words, A*, B*, C*, D* can be presented by the following simple equations:

$$T = A \oplus B \oplus C \oplus D;$$

$$A^* = A \oplus T;$$

$$B^* = B \oplus T;$$

$$C^* = C \oplus T;$$

$$D^* = D \oplus T;$$

2.1.2. S-boxes of Zodiac

Two types of substitution functions, called S-boxes, construct the round-function of Zodiac and these two S-boxes are originated from the two different functions over GF(2^8). One of which is the algebraic inversion function in the finite field (binary) GF(2^8) and the other is the modular exponential operation in the modular number system. Let's call the first one by S1 and the second one by S2.

The two S-boxes S1 and S2 are generated by the following functions $h(X)$ and $g(X)$ respectively:

- $h(X) = h_0(h_0(X))$, Where $h_0(X) = (45^X \bmod 257) \bmod 256$
- $g(X) = (170 + X)^{-1}$ in GF(2^8), with irreducible polynomial, $X^8 + X^4 + X^3 + X + 1$

The detailed S-box tables for S1 and S2 are presented in appendix A-1. As previously mentioned, those two functions in the above were made up from different group structures and so they are not compatible and can't be imbedded into a same group in a compatible form. Moreover all the input bytes to these functions and their output bytes are permuted and mixed by the simple permutation networking of Zodiac's round-function of which detailed description is presented in section 2.1.3.

The maximum DC-probabilities of S1 and S2 are $2^{-4.7}$ and 2^{-6} respectively. And the maximum squared-LC probabilities of S1 and S2 are $2^{-4.4}$ and 2^{-6} respectively.

2.1.3. The Round Function

The round-function of Zodiac has very simple structure and all operations in it are based on 8-by-8 non-linear substitutions, i.e. S-box look-up, byte-wise networking and just XOR operation. The detailed structure is shown in the below Fig.2.2.

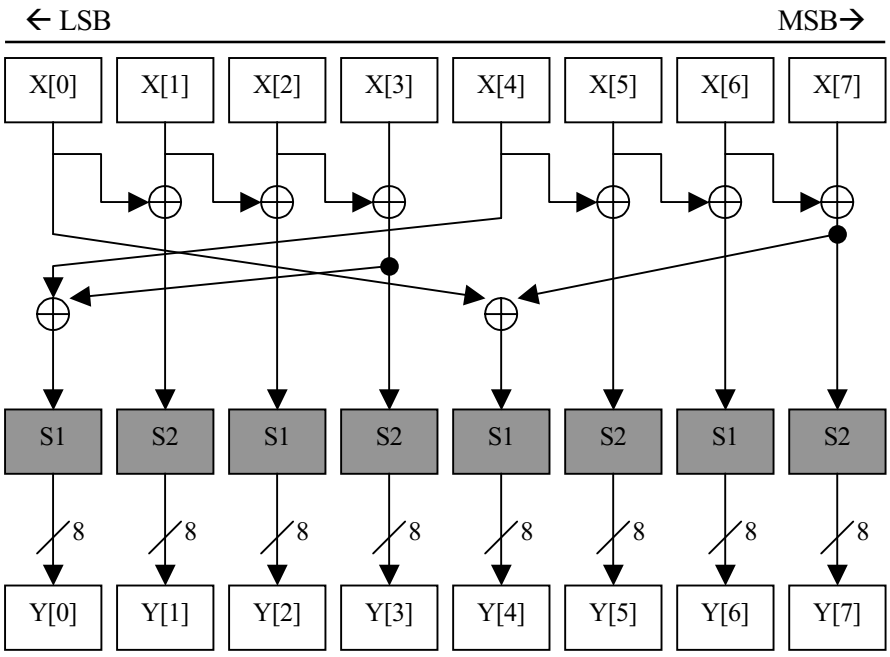


Figure 2.2: Round Function *F* of Zodiac

The simple data mixing permutation over the S-boxes in Fig.2.2 is bijective and moreover the permutation has just two gate-delays (or instruction-delays).

2.2. Key Scheduling Algorithm of Zodiac

The key-scheduling algorithm of Zodiac was basically constructed from its round-function as you see in the Fig.2.3-(2). We tried to design this key-schedule to meet the design goals presented in section 1.2 and to meet key-avalanche property, where both the simplicity and the effectiveness are severely reflected.

For the detailed key scheduling process, it consists of the three steps, say,

- Step-1: seed-key loading step (Fig.2.3-(1)),
- Step-2: round-key generating step with round-function (Fig.2.3-(2)),
- Step-3: role-changing step between the input-data-pad and key-pad (Fig.2.3-(2)).

As repeating (iterating) the steps-2 and step-3, all the round-key words are generated successively in 4 words by 4 words of 32-bits. In the following Fig.2.3-(1)~(2), all data blocks are of 32-bit word.

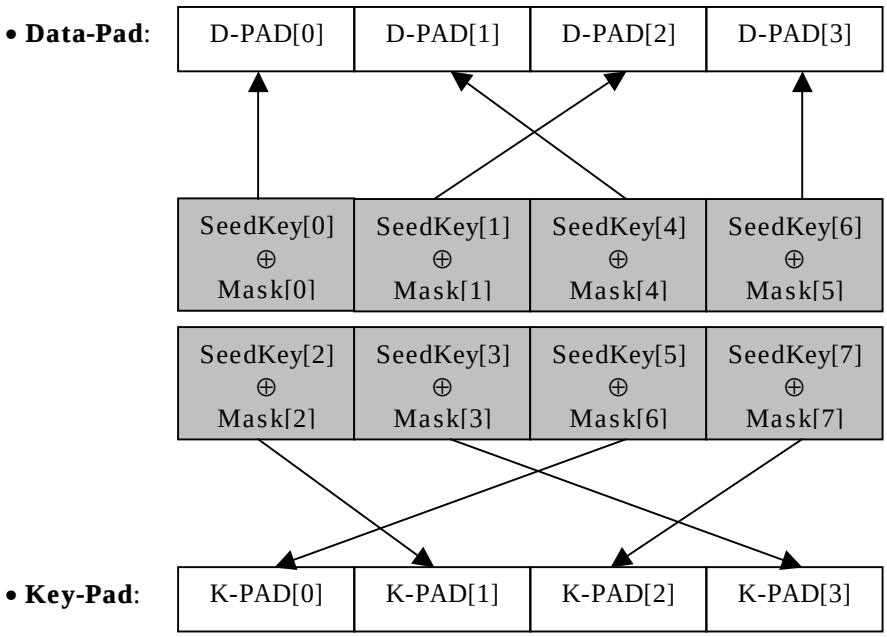


Figure 2.3-(1): Initial Seed-Key Loading on Data-Pads

The 32-bits masking constants, Mask[i]'s, appeared in Fig.2.3-(1) are listed in the following table. These constant values were picked out from the fractional part of the irrational number, $\frac{1}{\pi}$, in hexadecimal form.

Mask[0]	Mask[1]	Mask[2]	Mask[3]	Mask[4]	Mask[5]	Mask[6]	Mask[7]
bdba3bed	f36e6b11	cefb0d59	111ef1f1	72fc76bb	acb44526	9a26714f	37d81f7b

Now for the round-key generating step and data-pad updating step, see the following Fig.2.3-(2).

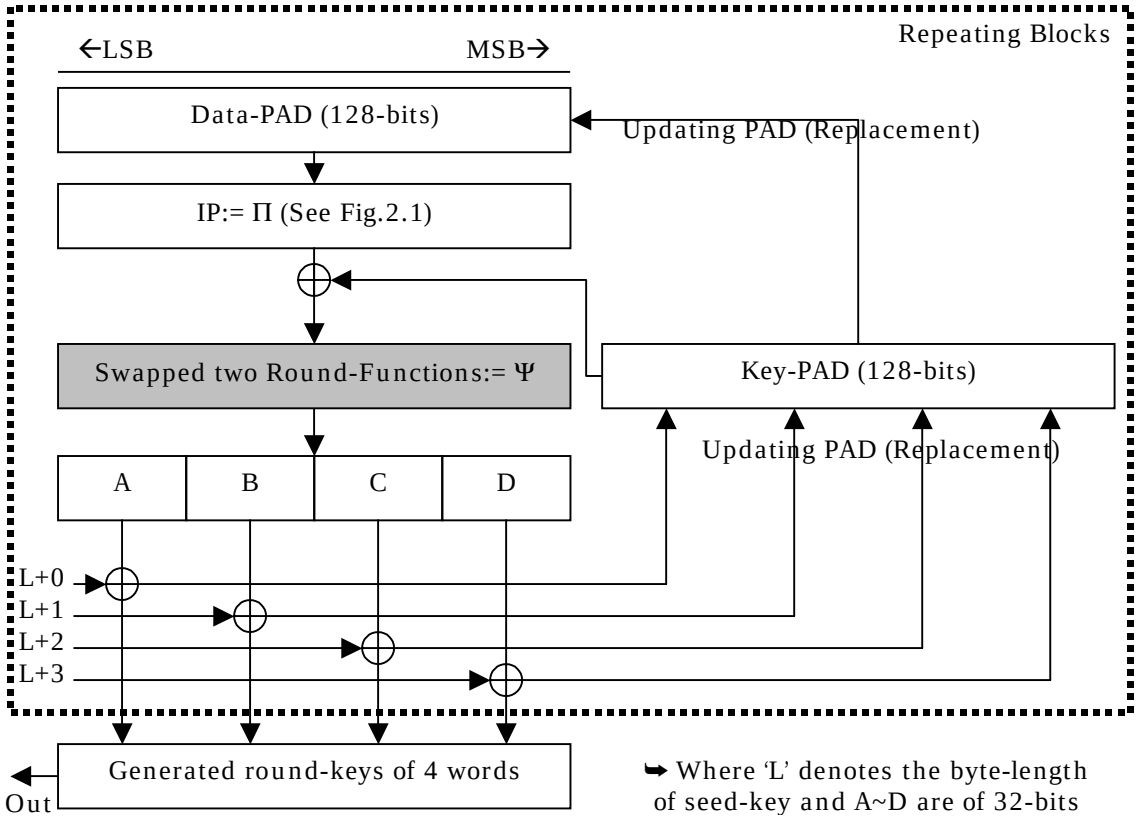


Figure 2.3-(2): Round-key generation and Data-Updating steps

In that Fig.2.3-(2), the shaded block (Ψ) denotes the function of consecutive two rounds in Fig.2.1 without round-key XORing. For the concrete description one can refer to the following diagram (Fig.2.3-(3)):

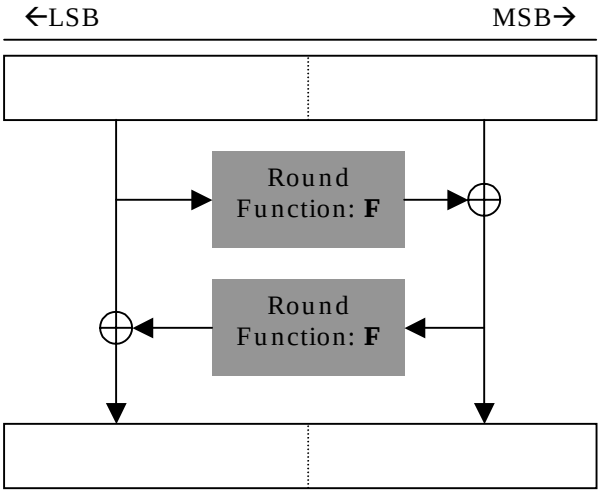


Figure 2.3-(3): The function Ψ

3. Security Consideration for Zodiac™

In this section, we support some technical evidence material for the security level of Zodiac in the typical aspects of differential cryptanalysis and linear cryptanalysis. And in addition, we briefly discuss on the interpolation attack and key-related attack.

3.1. Differential Cryptanalysis (in total differential sense for the provability)

To begin with, we introduce two main technical and theoretical results for dealing with DC-attack level to Zodiac. Throughout this section we consider only Feistel type block cipher in general. For more concrete decryption, our considering cipher gets 2N-bits plaintext as an input data, $X = (X^L, X^R)$, and throughput 2N-bits ciphertext $Y = (Y^L, Y^R)$ as its output data. Where the superscript 'L' and 'R' denote the N-bits left half of considering data and the N-bits right half of considering data respectively. Next we assume that the considering cipher has the following features:

- Its round function F is a function of N-bits input and N-bits output.
- F is constructed only by using n-bits-by-n-bits sub-functions called 'S-box' and a linear data mixing transformation, which is operated on n-bits-by-n-bits data block. Where we assume that S-box is bijective.
- In the above item, the S-box function has its maximum DC-characteristic $p_{\max}$ and finally the whole feature of the round function follows the below block diagram.

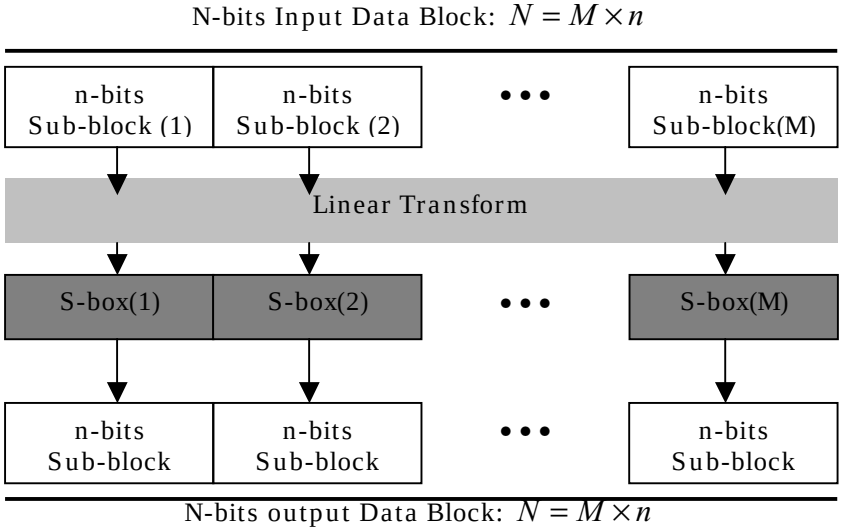


Fig.3.1: Our generic model of round function

Under the above assumptions, we start with some definitions for more convenient description.

Definition 3.1: *As an usual notation* $D_F(\Delta X \to \Delta Y)$ *of differential probability for the function* $F : \Omega \to \Omega$ *(where,* Ω *is a N-bit vector space) means the below:*

$$D_F(\Delta X \to \Delta Y) = \frac{\#\{X \mid F(X + \Delta X) + F(X) = \Delta Y\}}{2^N}.$$

We also use the same notation for the case of S-box, S*, instead of* F *just as* $D_S(\alpha \to \beta)$*, where* α *is the n-bit input difference to* S *and* β *is the n-bit output difference vector of* S*. And we denote by* $D(F)$ *the following maximum differential probability for* F *:*

$$D(F) = \max_{\Delta X \in \Omega} D_F(\Delta X \to \Delta Y).$$

Let C be the considering cipher which is a Feistel type and is built up by iterating the above one round function, F, up to R rounds. Now assume that a pair of a non-trivial input difference vector ΔX and an output difference vector ΔY of the cipher C is given as the below forms.

$$\Delta X = (\alpha_1, \alpha_2, ...\alpha_{2M})$$

$$\Delta Y = (\beta_1, \beta_2, ...\beta_{2M})$$

In addition, let's denote by δ_{ij} the output difference of the i^{th} round's j^{th} S-box, S_{ij}. Then, by equating the final summed-up difference vector (represented in n-bits sub differential data

blocks (vectors) with $(\beta_1, \beta_2, ...\beta_{2M})$, we get 2M equations $\{Eq.(k), k = 1,2,...,2M\}$ such that

$$Eq.(k) : f_k(\{\alpha_l\}, \{\delta_{ij}\}) = \beta_k .$$

Where $f_k(\{\alpha_l\}, \{\delta_{ij}\})$ denotes a linear combination of n-bits small difference vectors α_l's and δ_{ij}'s. Let ρ_{ij} be the n-bits input difference vector to S_{ij}, then ρ_{ij} is a linear combination of α_l's and δ_{ij}'s. Finally, if we denote by Ω the set of those componential difference vectors, then the total differential probability of the cipher C can be described as

$$D_C(\Delta X \to \Delta Y) = \sum_{(\delta_{ij}) \in \Omega^{MR}, \{Eq(k)\}} \prod_{i,j} D_{S_{ij}}(\rho_{ij} \to \delta_{ij}), \ \Delta X \neq 0 .$$

Note that all the intermediate difference vectors δ_{ij} inevitably appear in at least one equation among Eq(k)'s.

Definition 3.2: *Following the above contexts, if all the above equations, Eq(k), are linear on the component differential vectors δ_{ij}, we get a linear system of equation consisting of 2M equations Eq(k) of the variables δ_{ij} and we get also its coefficient matrix $\mathbf{M}$. Let's call it by 'fundamental differential matrix' or simply 'transient matrix'. A differential vectors string $\{(\alpha_l), (\delta_{ij}), (\beta_{ij})\}$ will be called an 'possible differential string' of C if*

$$D_C((\alpha_l) \to (\delta_{ij}) \to (\beta_{ij})) \neq 0$$

For example, consider a simple Feistel type cipher C of which round function F is identical to the S-box itself. See the following illustration:

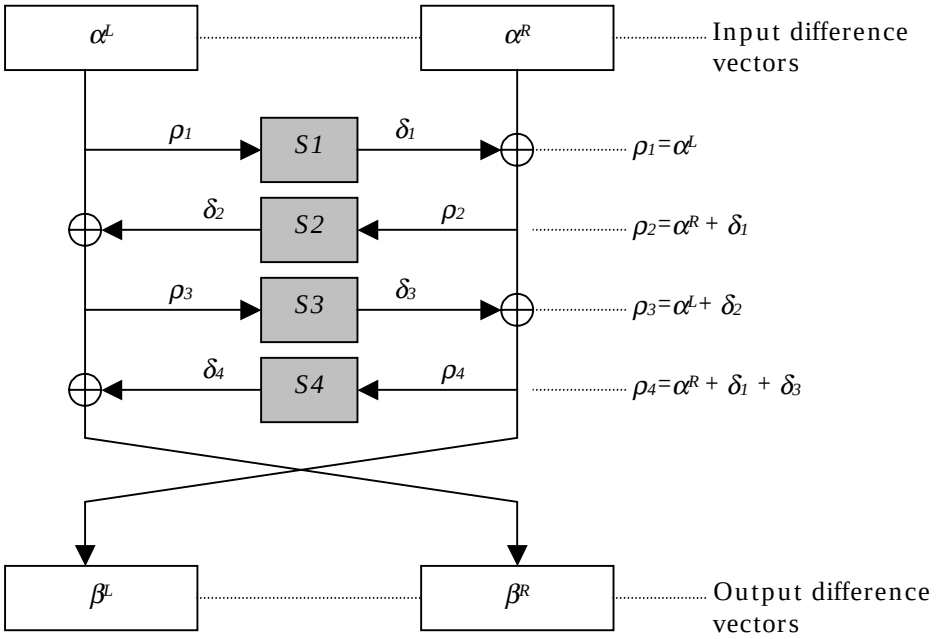


Fig.3.2: illustrated Feistel type block cipher

As you see in the Fig.3.2, we get the following two equations on the output difference variables of S-boxes:

$$\beta^L = \alpha^L + \delta_1 + \delta_3$$

$$\beta^R = \alpha^R + \delta_2 + \delta_4$$

Hence we have the following the fundamental differential matrix M of C:

	δ_1	δ_2	δ_3	δ_4
Eq(1)	1	0	1	0
Eq(2)	0	1	0	1

Note that if we fix $\delta_3 = 0$ then we have the following, newly confining equation,

$$Eq(\delta_3) : \alpha^R + \delta_2 = 0 \quad (\text{ by the bijectiveness of } S\ ).$$

And so we have the newly constructed (expanded) transient matrix M as the below:

	δ_1	δ_2	δ_4
Eq(1)	1	0	0
Eq(2)	0	1	1
Eq(δ_3)	0	1	0

Note that this modified transient matrix has its rank 3.

Definition 3.3: *Following the above contexts and definitions, given a non-trivial input and output difference pair, the fundamental differential matrix* M *of the cipher* C *will be called by 'primitive' if it satisfies the following conditions:*

- *Each input difference-vector to each S-box is represented by a linear combination of the output difference-vectors of S-boxes and the fixed input difference-vector of* C.
- *Given a pair of the input and output difference-vectors of* C, *the rank of the fundamental matrix* M *of* C *is 2M. Where M is the number of S-boxes used in each round as in the Fig.3.1.*
- *The round function* F *of* C *is bijective. Moreover for any newly generated transient matrix* M^* *which is derived from the fundamental matrix* M *by deleting some k columns of* M, *like fixing as* $\delta_{ij} = 0$ *and by adding the k related confining equations* $Eq(\delta)$ *to* M, *the below are satisfied:*

$$rank(\mathrm{M}^*) = 2M + 2k \quad (whenever\ \Delta X \neq 0).$$

Under the above contexts and definitions we get the following theorem.

Theorem 3.4: *If our considering cipher* C *has the fundamental differential matrix as primitive for each non-trivial input difference, then its maximal differential probability is asymptotically bounded by*

$$D(C) = \max_{(\Delta X, \Delta Y)} D_C(\Delta X \rightarrow \Delta Y)$$
$$\leq \sum_{0 \leq k \leq (RM-2M)/2} \binom{RM}{k} \left(\left(2^N\right)^{-2M-k} p_{av}^{-2k} \right)$$

Where M *is the number of S-boxes in one round function,* N *denotes the bit-size of the input data to the S-box, R is the number of iterated rounds and* p_{av} *denotes the average differential probability over all non-trivial differential characteristics of the S-boxes.*

Proof: *Let* (δ_{ij}) *be the output difference-vector string of S-boxes and* $\{(\alpha_l) \neq 0,\ (\beta_l)\}$ *are the fixed input difference-vector and output difference-vector strings, respectively, of the cipher* C *. Then*

$$D((\alpha_l) \to (\beta_l)) = \sum_{(\delta_{ij}),\{Eq(l)\}} \prod_{i,j} D_{S_{ij}}((\rho_{ij}) \to (\delta_{ij})) \ \ldots\ldots\ldots\ldots\ldots\ldots [eq3.1]$$

Note that the freedom-degree of the above equation on the difference-vectors (δ_{ij}) *is* RM *. Now we assume that the input difference- vector string* (α_l) *be a fixed non-trivial and we are to deal with the intermediate difference-vector string* (δ_{ij}) *and* (β_l) *as the variable difference-vector strings to be measured for maximizing the total differential probability in the eq.3.1. In addition, assume that that all differential probabilities of each S-box are almost uniformly distributed in the sense that the number of possible (i.e. non-degenerate) difference pairs corresponding to a given input difference-vector (non-trivial) to S-box is almost same over the non-trivial input difference-vectors except a few number of input difference-vector (say, the maximum differential probability). So, a given non-trivial input difference* α *to an S-box, the possible number of output difference-vectors can be estimated as* $\frac{1}{p_{av}}$*, and so the ratio of the number of possible input/output difference-pairs to the total number of difference-pairs for S-box can estimated as* $\left(2^N p_{av}\right)^{-1}$*. Note that diminishing (set by zero)* k_1 *components of* δ *with probability 1 decrease the freedom-degree of [eq.3.1]* $2k_1$*. In that case, the probability that the total difference-vector string* $((\alpha_l),(\delta_{ij}),(\beta_l))$ *is non-degenerated (i.e., having positive probability) is the following:*

$$\left(2^N p_{ac}\right)^{-2M-k_1}.$$

There exist some δ_{ij}*'s such that the action of diminishing them does not affect the increasing the rank of the transient differential matrix, since in fact, they can give some dependent confining equations on the default final round 2M equations. So if we denote by* k_2 *the number of such* δ_{ij}*'s, we know that the total freedom degree of the independent intermediate differential-vectors,* δ_{ij}*, is* $RM - 2M - 2k_1 - k_2$

Hence, summing up these all factors, we have the following deductive equation:

$$\begin{aligned} D((\alpha_l) \to (\beta_l)) &= \sum_{(\delta_{ij}),\{Eq(l)\}} \prod_{i,j} D_{S_{ij}}((\rho_{ij}) \to (\delta_{ij})) \\ &\leq \sum_{k_1,k_2} \binom{RM}{k} \left(\frac{1}{p_{av}}\right)^{RM-2M-2k_1-k_2} {p_{av}}^{RM-k_1-k_2} \left(2^N p_{av}\right)^{-(2M+k_1)} \\ &\cong \sum_{k_1, 0 \leq k_1 \leq (RM-2M)/2} \binom{RM}{k_1} {p_{av}}^{2M-k_1} \left(2^N p_{av}\right)^{-(2M+k_1)} \end{aligned}$$

Where subscript $k_1's$ range was given by the condition that the freedom degree should be greater than or equal to zero, i.e., $RM - 2M - 2k_1 - k_2 \geq 0$.

Now applying the above result to our cipher, Zodiac with 16-rounds, and substituting R=16, N=M=8 and $p_{av} = 2^{-7}$ as an example using the S-box of algebraic inversion. We are to know that its total differential probability asymptotically approaches to the bound 2^{-128}.

3.2. Differential Cryptanalysis (in the differential characteristic sense)

In this section, we'll investigate on the security level of Zodiac in the aspect of possible maximum differential characteristic by counting the minimum number of active S-boxes in 16 round Zodiac.

With the notion that input bytes to the round function of Zodiac affect on the adjacent input byte via the data mixing permutation and this avalanche propagates to the next round, we know that the best minimization to decrease the number of active S-boxes is to make all nontrivial input-difference bytes and their output-difference bytes be same. This tells us that if we denote the non-trivial input difference by '1' and zero-difference by '0', then we can simply count the appearance of '1' going into S-box, where the S-box is regarded as an identical linear function and all operations are binary XOR operations. This concept can be easily examined by a short program code. Using this simulation, we knew that in such above case the minimum number of active S-boxes is 40 with the 16 round Zodiac. This means 16 round Zodiac has the best differential characteristic as $\left(2^{-5.7}\right)^{40} = 2^{-228}$, where $2^{-5.7}$ is the average value of the two max DC-characteristics of S1 and S2.

3.3. Linear Cryptanalysis

Here we present another theorem related to the linear cryptanalysis for Feistel or SPN type block ciphers. Let's begin with the following lemma.

Lemma 3.5: *Let's define the two r-dimensional multi-variable functions $\phi_e^{(r)}$ and $\phi_o^{(r)}$ by the following:*

$$\phi_e^{(r)}(x_1, x_2, ..., x_r) := \sum_{\substack{a_i \in \{0,1\} \\ a_1 +, \bullet\bullet\bullet, + a_r = even}} \prod_{i=1}^{r} (1 - x_i)^{a_i} \, x_i^{a_i + 1 \bmod 2}$$

$$\phi_o^{(r)}(x_1, x_2, ..., x_r) := \sum_{\substack{a_i \in \{0,1\} \\ a_1 +, \bullet\bullet\bullet, + a_r = odd}} \prod_{i=1}^{r} (1 - x_i)^{a_i} \, x_i^{a_i + 1 \bmod 2}$$

$\phi_e^{(r)}$ is the sum of all possible products of even number of $(1-x_i)$'s and the remained x_j 's. Similarly $\phi_o^{(r)}$ is the sum of all possible products of odd number of $(1-x_i)$'s and the remained x_j 's.

Then we have

$$2\phi_e^{(r)} - 1 = \prod_{i=1}^{r}(2x_i - 1)$$

***Proof:** We show that by induction. First note that $\phi_e^{(r)} + \phi_o^{(r)} = \prod_{i=1}^{r}\left((1-x_i)+x_i\right) = 1$ and so $2\phi_e^{(r)} - 1 = \phi_e^{(r)} - \phi_o^{(r)}$.*

If r=2 then

$$\phi_e^{(2)} = x_1x_2 + (1-x_1)(1-x_2) \quad \textit{and}$$
$$2\phi_e^{(2)} - 1 = 4x_1x_2 - (x_1 + x_2) + 1 = (2x_1 - 1)(2x_2 - 1).$$

So the lemma holds for the case of r=2. Suppose that the lemma holds for all $2 \le r \le s$. Noting that

$$\phi_e^{(s+1)} = (1-x_{s+1})\phi_o^{(s)} + x_{s+1}\phi_e^{(s)},$$
$$\phi_o^{(s+1)} = (1-x_{s+1})\phi_e^{(s)} + x_{s+1}\phi_o^{(s)}.$$

Hence, we have

$$\begin{aligned}
&2\phi_e^{(s+1)} - 1 \\
&= \phi_e^{(s+1)} - \phi_o^{(s+1)} \\
&= (2x_{s+1} - 1)\phi_e^{(s)} - (2x_{s+1} - 1)\phi_o^{(s)} \\
&= (2x_{s+1} - 1)(\phi_e^{(s)} - \phi_o^{(s)}) \\
&= (2x_{s+1} - 1)(2\phi_e^{(s)} - 1) \\
&= \prod_{i=1}^{s+1}(2x_i - 1)
\end{aligned}$$

This completes the proof.

This implies the well known *piling-up-lemma* as you see in the following theorem.

***Theorem 3.6:** We use the same contexts until now. That is the considering cipher C is the R round block cipher of which block-size is 2N in bits. Let $\Lambda_0 = 2^{-N+1}$, Λ be the*

maximum linear characteristic of S-boxes, and the minimal number of active S-boxes is L. Then the total linear characteristic is asymptotically bounded by

$$\Lambda(C) \le \Lambda^L + \Lambda_0 .$$

Proof. *Let* f_1 *be the function of first two rounds,* f_2 *the function of the next two rounds, etc., and finally* f_l *be the function of last two rounds. Then the* $R(=2l)$ *rounds iterated cipher* C *is* $f_l \circ f_{l-1} \circ ..... \circ f_2 \circ f_1$. *For the computation of total linear characteristic let* $\lambda, \omega \neq 0$ *be the input and output masking vectors of* C *in* $GF(2^{2N})$ *respectively, that is we are to find the probabilistic range of* $\Lambda_{\lambda,\omega}(C)$. *Let* $\mu_i, i = 1,2,...,l-1$ *be any vectors in* $GF(2^{2N})$ *and then consider the following l iterated equations for a given input data* $X \in GF(2^{2N})$ *:*

$\lambda \bullet X = \mu_1 \bullet f_1(X)$ *with probability* p_1

$\mu_1 \bullet f_1(X) = \mu_2 \bullet f_2(X)$ *with probability* p_2, ...,

$\mu_{l-1} \bullet (f_{l-1} \circ ... \circ f_1)(X) = \omega \bullet (f_l \circ ... \circ f_1)(X)$, *with probability* p_l.

By the assumption of round independency, we know that the average probability for a randomly chosen $X \in GF(2^{2N})$ *satisfies:*

$$\lambda \bullet X = \omega \bullet (f_l \circ ... \circ f_1)(X) = \omega \bullet C(X)$$
$$= \phi_e^{(l)}(p_1, p_2, ..., p_l)$$

(See lemma 3.5). So if we let the Ψ *be the random variable to denote the number of* $X \in GF(2^{2N})$ *satisfying the above equation, then we know that* Ψ *approximately has the normal distribution*

$$\mathrm{N}(2^{2N}\phi_e^{(R)}, \sigma^2),\ \sigma = \sqrt{2^{2N}\phi_e^{(R)}(1-\phi_e^{(R)})} .$$

From this we get:

$$2^{2N}\phi_e^{(R)} - 2\sigma \le \Psi \le 2^{2N}\phi_e^{(R)} + 2\sigma$$

with probability 99.5%, where

$$\sigma = \sqrt{2^{2N}\phi_e^{(R)}(1-\phi_e^{(R)})} \le 2^N \frac{1}{2} = 2^{N-1} .$$

Hence, considering the formula $\Lambda_{\lambda,\omega}(C) = \left| 2\frac{\Psi}{2^{2N}} - 1 \right|$, *we get for each* λ, ω

$$\Lambda_{\lambda,\omega}(C) \le \left|2\phi_e^{(l)}(p_1, p_2, ..., p_l) - 1 + 4\sigma\right|$$
$$\le \left|2\phi_e^{(l)}(p_1, p_2, ..., p_l) - 1\right| + 4\sigma$$
$$\le \prod_{i=1}^{l}\left|2p_i - 1\right| + 2^{-N+1} \le \Lambda^L + \Lambda_0$$

This proves the theorem.

Hence, the total LC^2 is bounded by

$$(2^{-5.7})^{40} = 2^{-228}$$

Where $2^{-5.7}$ is the average value of the two max LC^2-characteristics of S1 and S2.

3.4. Other Cryptanalysis

As previously mentioned, our cipher Zodiac has two different types of S-boxes, which are constructed from different group operations, and so this characteristic can sufficiently suppress the interpolation attack.

In the aspect of key-related attack, the key-scheduling algorithm seems not to generate any weak and was designed to make all input seed-key bits affect all round-key bits.

4. Performance and Implementation Cost

4.1. Performance on Pentium Processor

On Pentium-III 450MHz, Zodiac encrypts at 14.7 Mbytes/sec=117.6 Mbps/sec. The consumed total clock-cycles were estimated as the following and this result was made from performing the cycle-estimating codes to Zodiac-Encrypter, which was adopted from

http://www.btinternet.com/~brian.gladman/cryptography_technology/aes2/index.html

*** The following table was made from performing the above simulator on Pentium 350 MHz*

*** Performance Comparison with AES Candidates*

Key-Length	Zodiac	Mars	Serpent	TwoFish	Rijndale	RC6
128 bits						
Key-Setup	705 cycles	2066 cycles	1293 cycles	8560 cycles	227/1297 cycles	1705 cycles
128-bit BlockEnc	480 cycles	372 cycles	956 cycles	370 cycles	363 cycles	280 cycles
128-bit BlockDec	488 cycles	380 cycles	906 cycles	389 cycles	368 cycles	241 cycles
192 bits						
Key-Setup	715 cycles	2070 cycles	1304 cycles	11803 cycles	214/1513 cycles	2045 cycles
128-bit BlockEnc	483 cycles	372 cycles	969 cycles	370 cycles	435 cycles	280 cycles
128-bit BlockDec	482 cycles	380 cycles	906 cycles	389 cycles	429 cycles	241 cyclces
256 bits						
Key-Setup	705 cycles	2076 cycles	1298 cycles	15635 cycles	290/1831 cycles	1901 cycles
128-bit BlockEnc	480 cycles	372 cycles	952 cycles	370 cycles	500 cycles	280 cycles
128-bit BlockDec	483 cycles	380 cycles	906 cycles	389 cycles	496 cycles	241 cycles

4.2. Hardware Implementation Cost

Zodiac takes 3200 gates and 66-gate delays for its hardware implementation.

Zodiac	Per one round			Delay per one round			Total Delay: Table-Lookup: 16*1, XOR:16*3 Sum=66 gate-delays	Total Gates: Include initial+ final masking 3200
	XOR	ADD	table	XOR	table			
	192	0	8	3	1			
Xenon	Per one round			Delay per one round			Total Delay: XOR:16*4 ADD:16*2 MUL:16*2	Total Gates:
	XOR	ADD	MUL	XOR	ADD	MUL		
	256	4	4	6	4	4		

The estimation values for Zodiac are taken up by simulating the VHDL circuit design developer.

In the following section 5, each test vector was listed and written in the order of 'from lsb-byte to msb-byte'.

5. Test Vectors

Seed Key (128 bit)	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
Plain Text	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Round-Key[0]	0a 27 1f 26 d8 c1 d0 77
Round-Key[1]	a4 dc b0 9e 45 4d 86 83
Round-Key[2]	07 b7 fd 26 8a da 29 91
Round-Key[3]	94 10 eb 1c 63 66 cc b5
Round-Key[4]	e0 5c 90 3d 74 9a 42 7a
Round-Key[5]	bb 3e c9 d4 ae 14 aa 7a
Round-Key[6]	b0 9d e6 4e eb ac df e4
Round-Key[7]	11 3a 2a 93 e9 dc c0 de
Round-Key[8]	72 4e 48 f0 66 54 84 53
Round-Key[9]	04 6f 19 22 05 05 91 6e
Round-Key[10]	e7 11 7c 25 05 3e c0 b4
Round-Key[11]	f7 64 e4 ef 49 44 ac c0
Round-Key[12]	d9 f3 e6 c4 59 8e 0a 28
Round-Key[13]	0c 29 78 93 b4 d5 ff 6f
Round-Key[14]	6b 3f 4c bf 3b 4b dc c6
Round-Key[15]	19 0f 51 4a 64 b5 05 69
Round-Key[16]	d9 0c ef 30 df 61 72 9c
Round-Key[17]	72 34 93 c9 04 f0 5d 58
Cipher Text	8e 0f ac 76 1a 29 b3 22 a3 16 ec 7b 30 8b 0d 5f

Seed Key	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
Plain Text	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
Round-Key[0]	0a 27 1f 26 d8 c1 d0 77
Round-Key[1]	a4 dc b0 9e 45 4d 86 83
Round-Key[2]	07 b7 fd 26 8a da 29 91
Round-Key[3]	94 10 eb 1c 63 66 cc b5
Round-Key[4]	e0 5c 90 3d 74 9a 42 7a
Round-Key[5]	bb 3e c9 d4 ae 14 aa 7a
Round-Key[6]	b0 9d e6 4e eb ac df e4
Round-Key[7]	11 3a 2a 93 e9 dc c0 de
Round-Key[8]	72 4e 48 f0 66 54 84 53
Round-Key[9]	04 6f 19 22 05 05 91 6e
Round-Key[10]	e7 11 7c 25 05 3e c0 b4
Round-Key[11]	f7 64 e4 ef 49 44 ac c0
Round-Key[12]	d9 f3 e6 c4 59 8e 0a 28
Round-Key[13]	0c 29 78 93 b4 d5 ff 6f
Round-Key[14]	6b 3f 4c bf 3b 4b dc c6
Round-Key[15]	19 0f 51 4a 64 b5 05 69
Round-Key[16]	d9 0c ef 30 df 61 72 9c
Round-Key[17]	72 34 93 c9 04 f0 5d 58
Cipher Text	f0 ad b8 0d 5e bf d0 20 10 a7 81 dc 16 93 f2 bd

Seed Key (192 bit)	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17
Plain Text	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Round-Key[0]	76 00 15 c7 f9 16 67 c7
Round-Key[1]	bc cd a2 8d af 5c 94 90
Round-Key[2]	61 d9 c1 af 8c 86 4e fd
Round-Key[3]	d1 38 36 0a 60 20 6d e1
Round-Key[4]	70 ae 55 a3 74 05 bd 8c
Round-Key[5]	1b f0 db 93 ac b7 3d dd
Round-Key[6]	19 d6 07 73 8f 93 7b e0
Round-Key[7]	1a 65 79 37 8d 83 df b4
Round-Key[8]	c0 d6 41 e2 c0 83 5d 08
Round-Key[9]	38 bf cc ac df f2 cb 33
Round-Key[10]	40 39 e3 dd 92 46 75 ab
Round-Key[11]	51 7e 01 fa 9a 6d 25 45
Round-Key[12]	08 19 18 da 7b 62 a4 e6
Round-Key[13]	57 f9 69 1d 4e a8 8c 0a
Round-Key[14]	1e 37 45 ac df e8 8d 6e
Round-Key[15]	2c 9a 6f 6a 49 03 de a3
Round-Key[16]	2e 56 ab 51 13 e0 a4 69
Round-Key[17]	52 e7 72 68 f7 5b 0b f2
Cipher Text	67 98 7a 6f a6 26 61 b3 a5 41 fa 04 93 3d 35 29

Seed Key (192 bit)	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17
Plain Text	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
Round-Key[0]	76 00 15 c7 f9 16 67 c7
Round-Key[1]	bc cd a2 8d af 5c 94 90
Round-Key[2]	61 d9 c1 af 8c 86 4e fd
Round-Key[3]	d1 38 36 0a 60 20 6d e1
Round-Key[4]	70 ae 55 a3 74 05 bd 8c
Round-Key[5]	1b f0 db 93 ac b7 3d dd
Round-Key[6]	19 d6 07 73 8f 93 7b e0
Round-Key[7]	1a 65 79 37 8d 83 df b4
Round-Key[8]	c0 d6 41 e2 c0 83 5d 08
Round-Key[9]	38 bf cc ac df f2 cb 33
Round-Key[10]	40 39 e3 dd 92 46 75 ab
Round-Key[11]	51 7e 01 fa 9a 6d 25 45
Round-Key[12]	08 19 18 da 7b 62 a4 e6
Round-Key[13]	57 f9 69 1d 4e a8 8c 0a
Round-Key[14]	1e 37 45 ac df e8 8d 6e
Round-Key[15]	2c 9a 6f 6a 49 03 de a3
Round-Key[16]	2e 56 ab 51 13 e0 a4 69
Round-Key[17]	52 e7 72 68 f7 5b 0b f2
Cipher Text	07 31 ce 35 8b 0c 76 dd a4 db 92 5f 05 86 b3 c5

Seed Key (256 bit)	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
Plain Text	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Round-Key[0]	28 c8 9a 95 b8 12 84 b7
Round-Key[1]	3e 2a bc 36 ed fc 9f bb
Round-Key[2]	cf 0c 05 03 10 c3 ff 90
Round-Key[3]	55 84 ed e1 da d1 43 fb
Round-Key[4]	35 5e 86 df 3d 4f ee 95
Round-Key[5]	4a 46 58 3d 63 d8 4c a6
Round-Key[6]	e2 44 14 cb 70 48 bb 05
Round-Key[7]	8e f1 b7 94 53 bb 33 07
Round-Key[8]	fa 0e 75 c8 3e 29 34 c1
Round-Key[9]	1f 1f 4d 06 f0 05 76 5f
Round-Key[10]	4e d6 bf 7d ac dd 36 0c
Round-Key[11]	c7 8b 12 49 fb 0f 57 77
Round-Key[12]	e4 85 50 ab 3a 55 61 b0
Round-Key[13]	4d d3 40 03 83 f3 9f 71
Round-Key[14]	61 22 80 fc 12 61 4c 23
Round-Key[15]	93 5a 7d 71 c2 7d b6 66
Round-Key[16]	f7 ee 30 97 8a 60 2a d3
Round-Key[17]	8b 8d 6b 09 ff f9 b3 13
Cipher Text	57 b2 2b 84 6e 85 41 55 71 23 42 a1 da 55 bc db

Seed Key (256 bit)	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
Plain Text	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
Round-Key[0]	28 c8 9a 95 b8 12 84 b7
Round-Key[1]	3e 2a bc 36 ed fc 9f bb
Round-Key[2]	cf 0c 05 03 10 c3 ff 90
Round-Key[3]	55 84 ed e1 da d1 43 fb
Round-Key[4]	35 5e 86 df 3d 4f ee 95
Round-Key[5]	4a 46 58 3d 63 d8 4c a6
Round-Key[6]	e2 44 14 cb 70 48 bb 05
Round-Key[7]	8e f1 b7 94 53 bb 33 07
Round-Key[8]	fa 0e 75 c8 3e 29 34 c1
Round-Key[9]	1f 1f 4d 06 f0 05 76 5f
Round-Key[10]	4e d6 bf 7d ac dd 36 0c
Round-Key[11]	c7 8b 12 49 fb 0f 57 77
Round-Key[12]	e4 85 50 ab 3a 55 61 b0
Round-Key[13]	4d d3 40 03 83 f3 9f 71
Round-Key[14]	61 22 80 fc 12 61 4c 23
Round-Key[15]	93 5a 7d 71 c2 7d b6 66
Round-Key[16]	f7 ee 30 97 8a 60 2a d3
Round-Key[17]	8b 8d 6b 09 ff f9 b3 13
Cipher Text	94 9d f4 4a 33 35 f8 5f e2 94 cf cc 15 ba 4c c0

6. Statistical Test

6.1. A Brief Description of Tests

6.1.1. Frequency Test

Given an n-bit stream, the purpose of this test is to check whether the number of ones and zeros are approximately uniformly distributed. Let n_0, n_1 be the number of zeros and ones respectively. If the length of the binary stream is greater than or equal to 10, then the statistic Z follows the standard normal distribution.

$$Z = 2\sqrt{n}\left(\frac{n_1}{n} - \frac{1}{2}\right)$$

6.1.2. Serial Test

This test is to determine if the numbers of occurrences of 00, 01, 10, and 11 are approximately the same. The notations n_{00}, n_{01}, n_{10}, n_{11}, n_0, n_1 represent the number of 00, 01, 10, 11, 0, 1 in the stream respectively. Select the overlapping subsequences in the stream. If the length of binary stream is not less than 21, then the statistic X approximately follows the chi-square distribution with freedom degree of 2

$$X = \frac{4}{(n-1)}\left(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2\right) - \frac{2}{n}\left(n_0^2 + n_1^2\right) + 1$$

6.1.3. Poker Test

Given an n-bit binary stream, the poker test determines if the sequences of length m each appear approximately the same number of times in the binary stream, as would be expected for a random sequence. The length m of subsequences must be $\lfloor n/m \rfloor > 5*2^m$. We partition the binary stream into k non-overlapping m-bit subsequences. Let n_i be the number of occurrences of the i^{th} type of sequence of length m, $1 \leq i \leq 2^m$. The statistic X used approximately follows a chi-square distribution with freedom degree of $2^m - 1$

$$X = \frac{2^m}{k}\left(\sum_{i=1}^{2^m} n_i^2\right) - k$$

6.1.4. Runs Distribution Test

A run of length i in a binary stream is a set of i consecutive ones or zeros. Especially a run of ones (or zeros) is called a block (or gap). It is expected that if the stream is random, then the number of occurrences of runs of various length are approximately uniform. The expected number of blocks (or gaps) of length i in a random n-bit sequence is $e_i = (n-i+3)/2^{i+2}$. Let k be the largest integer i for which $e_i \leq 5$. Let b_i, g_i be the number of blocks and gaps, respectively, of length i in a binary stream for each i, $1 \leq i \leq k$. The statistic X approximately follows the chi-square distribution with freedom degree of 2k – 2.

$$X = \sum_{i=1}^{k} \frac{(b_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(g_i - e_i)^2}{e_i}$$

6.1.5. Runs Test

This test is to determine if the conditional probability of each subsequent element, s_i for i = 2, … , n, is dependent on the previous element only. A change indicates the start of a new run of a different value and thus increments the number of runs for the stream. Hence the total number of runs in the stream equals one more than the total number of changes. Possibly, the number of runs ranges from 1 to n. The null hypothesis for the test is $\Pr(s_i = 1 \mid s_{i-1} = 0) = \Pr(s_i = 0 \mid s_{i-1} = 1) = 1/2$. The statistic Z follows the standard normal distribution.

$$Z = \frac{2R - n - 1}{\sqrt{n-1}}$$

6.1.6. Autocorrelation Test

Given an n-bit binary stream $S = \{s_i\}$, the purpose of this test is to determine the correlation between the original stream and d bit non-cyclic shifted stream $S_d = \{s_{i+d}\}$. Let the shift amount, d, be the fixed integer between 1 and n/2. Specially it recommended to select one of the numbers $\lfloor n/4 \rfloor$, … , $\lfloor 3n/4 \rfloor$. If n-d is greater than and equal to 10, the statistic Z approximately follows the standard normal distribution where M = min (d, n-d).

$$A(d) = \sum_{i=0}^{i=n-d-1} s_i \oplus s_{i+d} \qquad Z = \frac{2}{\sqrt{M}}\left(A(d) - \frac{M}{2} \right)$$

6.1.7. Binary Derivative Test

The binary derivative is a new stream generated from the initial stream. The first binary derivative $d_1(s)$ is the binary stream of length n-1 formed by the XORing of overlapping pairs of values in the initial stream s. The k^{th} binary derivative stream of length n-k, is the result of executing the above operation k times successively. This test is to determine if the number of ones or zeros of the k^{th} binary derivative stream is approximately uniform. Let p_k be the proportion of ones in the k^{th} binary derivative stream. The statistic Z follows the standard normal distribution.

$$Z = 2\sqrt{n-k}\left(p_k - \frac{1}{2}\right)$$

6.1.8. Change Point Test

The change point test determines the bit position in the stream at which the change in the proportion of ones before and after differs the most. At each bit position t in the stream the proportion of ones to that point is compared to the proportion of ones in the remaining stream. The bit where the maximum change occurs is called the change point. If the binary stream is random, then there is no change in the proportion of ones throughout. Given n bit binary stream, let S[n] be the number of ones in the entire binary stream. Let S[t] be the number of ones in the first t-bit subsequence. The statistic to this test is U[t] = nS[t] – tS[n], t =1, 2, …, n-1. If M denotes the maximum absolute value of U[t] i.e., M = Max{|U[t]|}, then tail area probability α is as follows and is applied to the one-tailed test. For the two-tailed test, 2α is used.

$$\alpha = e^{-\frac{2 \cdot M^2}{nS[n](n-S[n])}}$$

6.1.9. Sequence Complexity Test

The sequence complexity is the number of new patterns encountered when the stream is observed successively. If a number of new patterns exist, it is difficult to distinguish between the given stream and a random stream. That is, if the sequence complexity of the binary stream is small, then there are many repetitive patterns in the stream and the period of the stream is small. Thus the stream is not random. If the sequence complexity is greater than or equal to the threshold value, n/log_n, the binary sequence passes this test.

6.1.10. Linear Complexity Test

This test is to determine the minimum amount of information needed in order to reconstruct the entire stream. The minimum length of LFSR (Linear Feedback Shift Registers) to construct the binary stream S, is called the linear complexity. Given a binary stream S with linear complexity L, we are able to reconstruct the entire stream with 2L consecutive bits stream. Hence L must be large so that it is not possible to reconstruct the entire binary stream easily. The recurrence relation for LFSR is a linear function under addition modulo 2, and may be expressed as follows, where $\oplus$ is the addition modulo 2, a_i in $\{0,1\}$, i = 1, …, L, t > L.

$$s_t = a_1 s_{t-1} \oplus a_2 s_{t-2} \oplus \cdots \oplus a_L s_{t-L}$$

If the linear complexity is small, it is possible to reconstruct the entire binary stream using the recurrence relation, given the 2L consecutive bit stream. And so the binary stream is not random. This test is passed if the linear complexity approximate to n/2. For an n-bit random sequence, the expected value is $E(L) \approx n/2+\alpha$, where α is 2/9 if n is even parity, else 5/18 and a variance is $Var(L) \approx 86/81$. The statistic Z approximately follows the standard normal distribution.

$$Z = \sqrt{\frac{81}{86}}\left(L - \frac{n}{2} - \alpha \right)$$

Even if a stream are highly patterned or contains large subsequences that are highly patterned, the stream may actually be regarded as random using the linear complexity test. For example, given a stream of length n for which both halves of the stream consist of n/2 – 1 values of one element followed by one value of the other element, would be classified as random based on the linear complexity test. This stream is obviously patterned and would fail all the previous tests in this document. To overcome this problem we are concerned about the linear complexity profile and execute two tests related to linear complexity profile.

6.1.11. Linear Complexity Profile Jump Test

Given an n bit binary stream s, let s(i) be the first i bit subsequence of s. $L_{s(i)}$ denotes the linear complexities of s(i), then the values of $L_{s(i)}$ are defined to be the linear complexity profile of s and should follow approximately the i/2 line. When $L_{s(i)} > L_{s(i-1)}$, then jump occurs. We find the number of jumps, F in the linear complexity. For large n, F is approximately normally distributed with $\mu = n/4$, $\sigma^2 = n/8$. The statistic Z for the number of jumps F follows the standard normal distribution. Since a small number of jumps would indicate a stream within which patterns may exist, one-tailed test (lower tail) is applied.

$$Z = \sqrt{\frac{8}{n}}\left(F - \frac{n}{4} \right)$$

6.1.12. Linear Complexity Profile Jump Height Test

If a stream passes the linear complexity profile jump test, then the distribution of jump heights may be investigated. The height of jump height is the change in linear complexity, $L_{s(i)} - L_{s(i-1)}$ when a jump occurs. Let o_i be the number of jumps with height i. The total sum of the numbers of jumps is F. If a random sequence is given, the expected value of the number of jumps with height i, e_i is $F/2^i$. If a chi-squared goodness-of-fit test is applied to the observed, the statistic X is as follows and follows a chi-square distribution with k-1 degrees of freedom, where $k = \max\{ i \mid e_i > 5,\ i > 0 \}$.

$$X = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$$

6.1.13. Spectral Test

This test is based on an evaluation of the power spectrum of a finite string. Given an n-bit binary stream $\{s_k\}$, k=0,…,n-1, values 0, 1 of all s_k are changed to 1 and –1 respectively. Then DFT (discrete Fourier transform) of $\{ s_k\}$ is computed. If a random sequence is uniform distributed, the expected value m_r and a variance v_r of r^{th} power of I_m are expressed in the equations depending on r and n. And so using these values, the statistic X_r is computed and it approximately follows the standard normal distribution.

$$I_m = \frac{1}{n}\left| \sum_{k=0}^{n-1} \left(s_k e^{-i(2\pi/n)km} \right) \right|^2, m = 0, \cdots, n-1$$

$$m_r = r!\left[1 - \frac{r(r-1)}{2n} \right] \qquad v_r = (n/2-1)[m_{2r} - (1+r^2)(m_r)^2]$$

$$X_r = \frac{1}{v_r} \sum_{k=1}^{n/2-1} [I_k^r - m_r]$$

6.1.14. Universal Test

This test is based on the basic idea that it is impossible to significantly compress a random stream without loss of information. Thus if a sample sequence can be significantly compressed, the sequence is regarded as nonrandom. In this test, instead of actually compressing a sequence, the universal test computes the quantity related to the length of the compressed sequence. In order to effectively test, this test has a drawback that it require much longer sample sequence. At first select a parameter L between 6 and 16. Then partition a given binary stream s into non-overlapping L-bit blocks and discard the leftovers. For each i, $1 \le i \le Q+K$, let b_i be the integer whose binary representation is the i^{th} block. The blocks are scanned in order. A table T is obtained so that at each stage T[j] is the position of the last occurrence of the block corresponding to integer j, $0 \le j \le 2^L-1$. The first Q blocks of s are used to initialize table T; Q should be chosen to be at least $10*2^L$ in order to have a high likelihood that each of the 2^L L-bit blocks occurs at least once in the first Q blocks. The remaining K blocks are used to define the statistic X.

$$X = \frac{1}{K}\sum_{i=Q+1}^{Q+K}\log_2(i - T[b_i]), Q+1 \le i \le Q+K$$

If we select the parameters Q and K such that $Q \ge 10*2^L, K \ge 1000*2^L$, then $Q+K \ge 1010*2^L$. The mean value μ and the variance σ^2 of X are as follows. A variance σ^2 of statistic X is computed using values of σ_1^2 as follows. The normalized statistic Z for X follows the standard normal distribution.

$\sigma^2 = c(L, K)^2 * \sigma_1^2 / K$, $c(L, K) \approx 0.7 - (0.8/L) + (1.6+(12.8/L))*K^{-4/L}$

$$Z = \frac{(X - \mu)}{\sigma}$$

L	μ	σ_1^2
1	0.7326495	0.690
2	1.5374383	1.338
3	2.4016068	1.901
4	3.3112247	2.358
5	4.2534266	2.705
6	5.1277052	2.954
7	6.1962507	3.125
8	7.1836656	3.238

6.1.15. Gap Test

This test is used to examine the length of gaps between occurrences of U_j, in a certain range. If α and β are two real numbers with $0 \le \alpha \le \beta \le 1$, we want to consider the lengths of consecutive subsequences $U_j, U_{j+1,\ldots,} U_{j+r}$ in which U_{j+r} lies between α and β but the other U's do not. The consecutive numbers between α and β are called gap. In this test, we examine the length of gap. The statistic is the number of occurrence of length of gap. Let p be the probability that $\alpha \le U_j < \beta$. The probability for each number of occurrence is $p_r = p(1 - p)^r$. For some integer t if the number of gaps of length is greater than t, then increment the number of occurrence of length t. and so $p_t = (1 - p)^t$. The statistic follows a chi-square distribution with t degrees of freedom.

6.1.16. Coupon Collector's Test

For the sequence $Y_0, Y_1, \ldots$, with $0 \le Y_j \le d$, this test counts the lengths of n consecutive coupon collectors segments. For this sequence, if the every number from 0 to d-1 occurs in some subsequence of $\{Y_j\}$, then it is called the complete set. The statistic is the number of occurrence of minimum length complete set. Since at least d entries in the subsequence needed in order to be the complete set, we examine the number of occurrence between d and t where t is the user-defined maximum. If the sequence length exceeds t, then it increments the number of occurrences of t. For each length, the probability is as follows.

$$P_r = \frac{d!}{d^r} S(r-1, d-1), d \le r \le r \qquad P_t = 1 - \frac{d!}{d^{t-1}} S(t-1, d)$$

Where S(n, k) is the Stirling number. For each length, the number of occurrences follows a chi-square distribution with t – d degrees of freedom.

6.1.17. Collision Test

This test can be used when the number of categories is much larger than the number of observations. Suppose we have m urns and we throw n balls at random into those urns, where m is much greater than n. Most of the balls will land in urns that were previously empty, but if a ball falls into an urn that already contains at least one ball, we say that a collision has occurred. The collision-test counts the number of collisions and computes the probability for this number. If the probability is less than 1, then sequence passes this test.

6.2. Collecting Sample Data

6.2.1. Random Plaintext

In order to examine the randomness of ciphertext based on random plaintext and random 128-bit keys, a number of sequences were constructed. According to the computational complexity of each test and test speed, we determine the length and the number of sample sequences. For frequency, serial, auto-correlation (shift amount = 2,560), change, 1st binary derivative, 4-bit poker, runs, runs distribution, sequence complexity, linear complexity tests, collect 10,000 10,240-bit samples and test for each even round. For coupon collector's test, gap test, collision tests, gather 10,000 10,240-bit sample sequences and test for the full round. For 8-bit poker test, collect one thousand of 10,240-bit sample sequences, and test for each even round. For the spectral test(r=7), collect one thousand of 8,192-bit sample sequences. And for LC profile-jump, and LC profile-jump-height test, collects one thousand of 5,120-bit samples. Finally, for universal test (L=8, Q= 20*256), one thousand of 517,120-bit sample sequence. Each sequence was a result of the concatenation of a number of ciphertext blocks using a number of random plaintexts and random 128-bit keys in the ECB mode.

6.2.2. Plaintext/Ciphertext Correlation

In order to examine the correlation of plaintext/ciphertext pairs, a number of sequences were constructed. Given a random 128-bit key and a number of random plaintext blocks, a binary sequence was constructed concatenating derived blocks where a derived block is the result of applying the XOR operator on the plaintext block and its corresponding ciphertext block computed in the ECB mode. According to the computational complexity of each test and test speed, we determine the length and the number of sample sequences. For frequency, serial, auto-correlation (shift amount = 2,560), change, 1st binary derivative, 4-bit poker, runs, runs distribution, sequence complexity, linear complexity tests, collect ten thousands of 10,240-bit samples and test for each even round. For coupon collector's test, gap test, collision tests, gather ten thousands of 10,240-bit sample sequences and the test for the full round. For 8-bit poker test, collect one thousand of 10,240-bit sample sequences, and test for each even round. For the spectral test(r=7), collect 1,000 8192-bit sample sequences. And for LC profile-jump, and LC profile-jump height test, collects one thousand of 5,120-bit samples. Finally, for the universal test (L=8, Q= 20*256), one thousand of 517,120-bit sample sequences. Each sequence was a result of the concatenation of a number of ciphertext blocks using a number of random plaintexts and random 128-bit keys in the ECB mode.

6.2.3. Low Density Plaintext

The data set was created based on low-density blocks used as plaintext. The data set consisted of 128 sequences. Each sequence consisted of 8,257 ciphertext blocks computed in the ECB mode. These ciphertext blocks were formed from one all zero plaintext block, 128 plaintext blocks of a single one and 127 zeroes (the one appearing in each of the possible 128

bit positions), and 8,128 plaintext blocks of two ones and 126 zeroes (the two ones appearing in each combination of two bit positions within the 128-bit positions).

6.2.4. High Density Plaintext

The data set was created based on high-density blocks used as plaintext. The data set consisted of 128 sequences. Each sequence consisted of 8,257 ciphertext blocks computed in the ECB mode. These ciphertext blocks were formed from one all ones plaintext block, 128 plaintext blocks of a single zero and 127 ones (the one appearing in each of the possible 128 bit positions), and 8,128 plaintext blocks of two zeroes and 126 ones (the two zeroes appearing in each combination of two bit positions within the 128-bit positions).

6.3. Test Result: see Appendix-C

7. Zodiac S-box input and output difference pairs

Let a 128-bit block $((\alpha_0,\cdots,\alpha_7),(\overline{\alpha_0},\cdots,\overline{\alpha_7}))$ be an input of the encryption function which consists of 16 rounds. Consider the corresponding 128-bit output $((\gamma_0,\cdots,\gamma_7),(\overline{\gamma_0},\cdots,\overline{\gamma_7}))$ of the encryption function and the 64-bit output $(\beta_0^i,\cdots,\beta_7^i)$ of S-boxes in each round.i. Then we will find the equation with γ_j, $\overline{\gamma_j}$, β_j^i and α_j, and calculate input equations corresponding with the S-box outputs $(\beta_0^i,\cdots,\beta_7^i)$, where i is denoted by the round number, and $0,\cdots,7$ are denoted by each number of byte-blocks.

At first, consider the equation about γ_j and $\overline{\gamma_j}$. In the first round, $(\alpha_0,\cdots,\alpha_7)$ is entered in the S-boxes, and in the second round, the value xored $(\overline{\alpha_0},\cdots,\overline{\alpha_7})$ and the first round output is entered into the second round. In the third round, the value xored the second round output and the first round input is the third round input, and so on. Continuing this process, finally, we can get the following equations.

$$\gamma_j = \overline{\alpha}_j \oplus_{k=1}^{8} \beta_j^{2k-1}$$

$$\overline{\gamma_j} = \alpha_j \oplus_{k=1}^{8} \beta_j^{2k}$$

Now, observe the relation between S-box inputs and outputs. The input of the first round output β_j^i is the follows. If $j=0$, the S-box input is $\alpha_2 \oplus \alpha_3 \oplus \alpha_4$, and if $j=4$, the S-box input is $\alpha_0 \oplus \alpha_6 \oplus \alpha_7$, otherwise, the S-box input is $\alpha_{j-1} \oplus \alpha_j$. In the second round, the value xored $\overline{\alpha_j}$ and the first round output is the S-box inputs. If $j=0$, the S-box input is $(\overline{\alpha_2} \oplus \beta_2^1) \oplus (\overline{\alpha_3} \oplus \beta_3^1) \oplus (\overline{\alpha_4} \oplus \beta_4^1)$, and if $j=4$, the S-box input is $(\overline{\alpha_0} \oplus \beta_0^1) \oplus (\overline{\alpha_6} \oplus \beta_6^1) \oplus (\overline{\alpha_7} \oplus \beta_7^1)$, otherwise, the S-box input is $(\alpha_{j-1} \oplus \beta_{j-1}^1) \oplus (\alpha_j \oplus \beta_j^1)$. In the third round, the value xored the first round input and second round output is the third round input. So, the inputs of S-box are represented by α_j and β_j^2. If $j=0$, the S-box input is $(\alpha_2 \oplus \beta_2^2) \oplus (\alpha_3 \oplus \beta_3^2) \oplus (\alpha_4 \oplus \beta_4^2)$, and if $j=4$, the S-box input is $(\alpha_0 \oplus \beta_0^2) \oplus (\alpha_6 \oplus \beta_6^2) \oplus (\alpha_7 \oplus \beta_7^2)$, otherwise, the S-box input is $(\alpha_{j-1} \oplus \beta_{j-1}^2) \oplus (\alpha_j \oplus \beta_j^2)$. The fourth round S-box inputs are represented by the value xored the second round inputs and the third round S-box outputs β_j^3, that is, they are represented by $\overline{\alpha_j}$, β_j^1 and β_j^3. Continuing this, we obtain the following results. In the even round, the S-box inputs are represented by $\overline{\alpha_j}$ and the previous odd round S-box outputs, and in the odd round, the S-box inputs are represented by α_j and the previous even round S-box outputs. The follows are the results.

RESULT: REPRESENTATION

If i is even,

$$\beta_j^{\,i} \quad \leftrightarrow \quad \begin{cases} (\alpha_2 \oplus_{k=1}^{l} \beta_2^{2k-1}) \oplus (\alpha_3 \oplus_{k=1}^{l} \beta_3^{2k-1}) \oplus (\alpha_4 \oplus_{k=1}^{l} \beta_4^{2k-1}) & j = 0 \\ (\alpha_0 \oplus_{k=1}^{l} \beta_0^{2k-1}) \oplus (\alpha_6 \oplus_{k=1}^{l} \beta_6^{2k-1}) \oplus (\alpha_7 \oplus_{k=1}^{l} \beta_7^{2k-1}) & j = 4 \\ (\alpha_{j-1} \oplus_{k=1}^{l} \beta_{j-1}^{2k-1}) \oplus (\alpha_j \oplus_{k=1}^{l} \beta_j^{2k-1}) & j \neq 0 \quad 4 \end{cases},$$

and if i is odd,

$$\beta_j^{\,i} \quad \leftrightarrow \quad \begin{cases} (\overline{\alpha_2} \oplus_{k=1}^{l} \beta_2^{2k}) \oplus (\overline{\alpha_3} \oplus_{k=1}^{l} \beta_3^{2k}) \oplus (\overline{\alpha_4} \oplus_{k=1}^{l} \beta_4^{2k}) & j = 0 \\ (\overline{\alpha_0} \oplus_{k=1}^{l} \beta_0^{2k}) \oplus (\overline{\alpha_6} \oplus_{k=1}^{l} \beta_6^{2k}) \oplus (\overline{\alpha_7} \oplus_{k=1}^{l} \beta_7^{2k}) & j = 4 \\ (\overline{\alpha_{j-1}} \oplus_{k=1}^{l} \beta_{j-1}^{2k}) \oplus (\overline{\alpha_j} \oplus_{k=1}^{l} \beta_j^{2k}) & j \neq 0, \quad 4 \end{cases},$$

where $l = \left\lfloor \frac{i}{2} \right\rfloor$ and $j = 0, \cdots, 7$, i is denoted by round number, that is, $i = 1, \cdots, 16$.

We can make a 144×128 matrix that represents this result. See the Appendix-B.

8. Intellectual Property Statement

We, SoftForum©, do not discriminate any use of and any user of this cipher, Zodiac. SoftForum© does release the cipher in a reasonable and non-discriminatory fashion with license-free and royalty-free.

The only one thing to do in the use of the cipher is to make clear its ownership and disclose the cipher's name and the supporter's name, SoftForum©.

9. Maturity and Current Usage

Zodiac^{TM} has been built into our crypto-library for several applications including secure B2B transaction and secure banking system in Korean commercial domain. Moreover it's been planed to embed it into mp3 contents protection system and several contents protection & delivering system with co-working other watermarking technology suppliers.

Moreover we are planning its hardware implementation for the use in VPN-Gateway and it is currently used in our prototype product of PC-protection software, named 'XecurePC'.

Appendix-A: S-box Tables of Zodiac

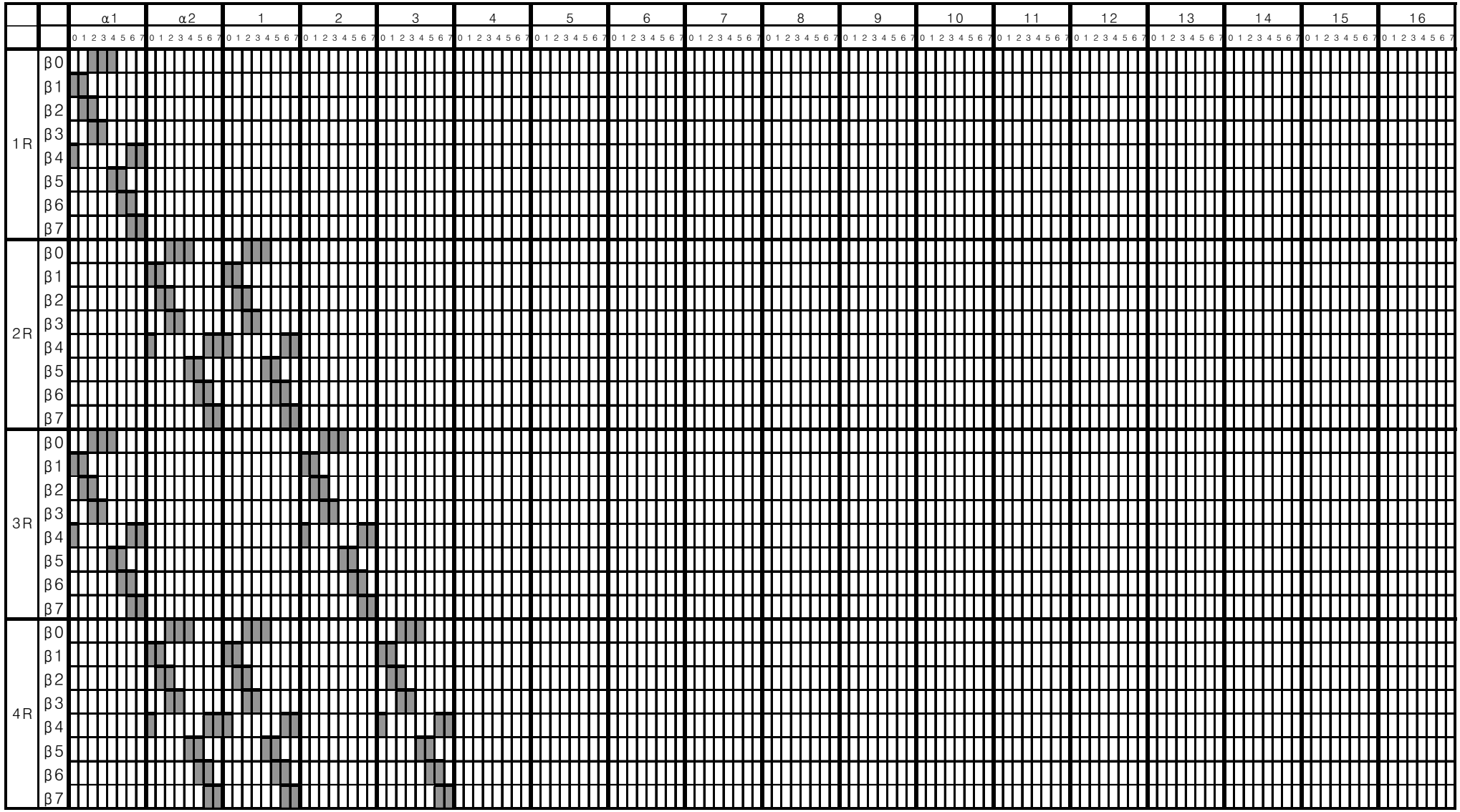
A.1. Sbox-1

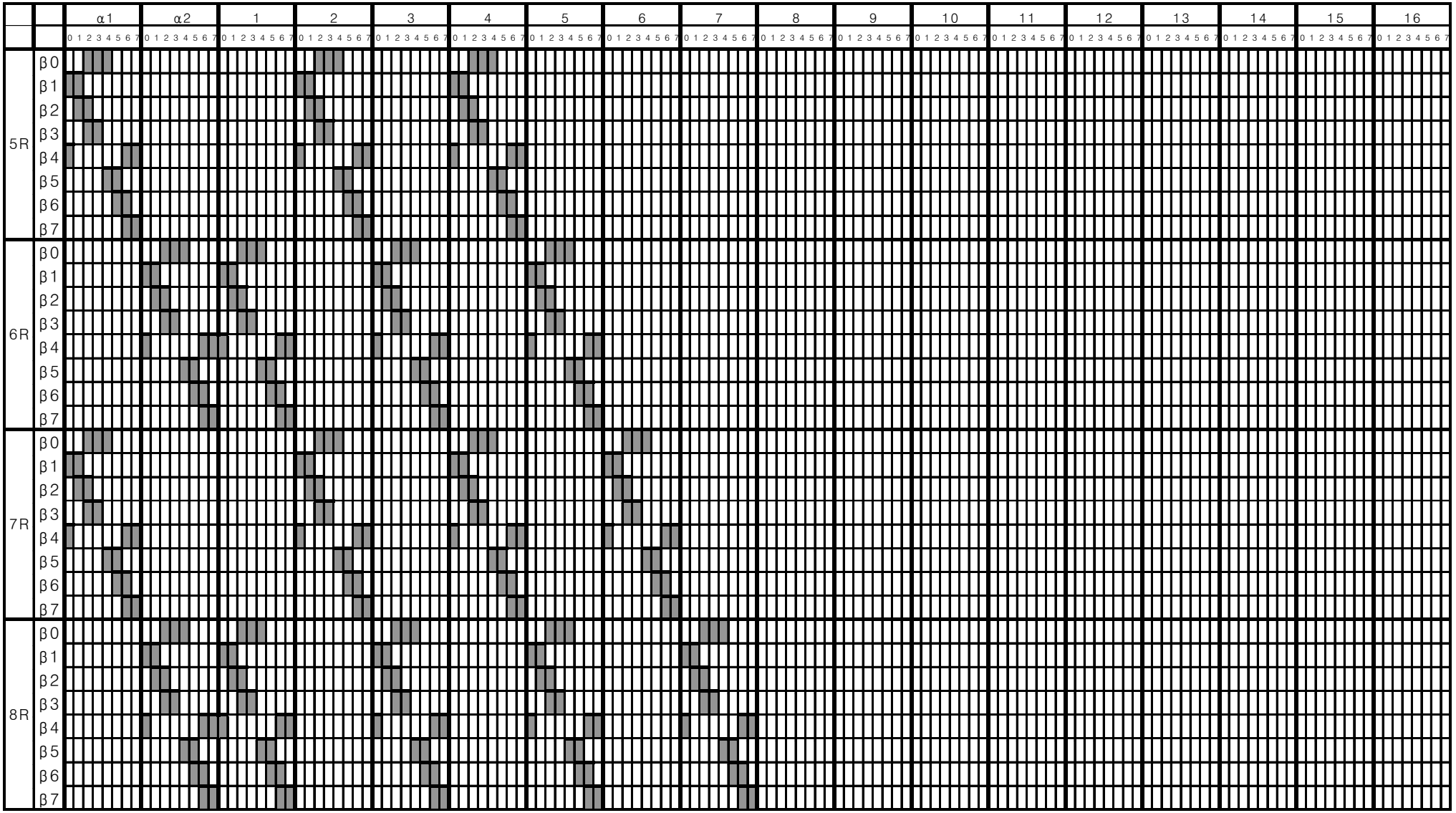
x	S1(x)	x	S1(x)	x	S1(x)	x	S1(x)	x	S1(x)	x	S1(x)	x	S1(x)	x	S1(x)
0	0x2d	1	0xf3	2	0x7c	3	0x6d	4	0x9d	5	0xb5	6	0x26	7	0x74
8	0xf2	9	0x93	10	0x53	11	0xb0	12	0xf0	13	0x11	14	0xed	15	0x83
16	0x78	17	0xb6	18	0x03	19	0x16	20	0x73	21	0x3b	22	0x1e	23	0x8e
24	0x70	25	0xbd	26	0x86	27	0x1b	28	0x47	29	0x7e	30	0x24	31	0x56
32	0xf1	33	0x77	34	0x88	35	0x46	36	0x97	37	0xb1	38	0xba	39	0xa3
40	0xb7	41	0x10	42	0x0a	43	0xc5	44	0x37	45	0xb3	46	0xc9	47	0x5a
48	0x28	49	0xac	50	0x64	51	0xa5	52	0xec	53	0xab	54	0xaa	55	0xc6
56	0x67	57	0x95	58	0x58	59	0x0d	60	0xf8	61	0x9a	62	0xf6	63	0x6e
64	0x66	65	0xdc	66	0x05	67	0x3d	68	0xd3	69	0x8a	70	0xc3	71	0xd8
72	0x89	73	0x6a	74	0xe9	75	0x36	76	0x49	77	0x43	78	0xbf	79	0xeb
80	0xd4	81	0x96	82	0x9b	83	0x68	84	0xa0	85	0x65	86	0x5d	87	0x57
88	0x92	89	0x1f	90	0xd5	91	0x71	92	0x5c	93	0xbb	94	0x22	95	0xc1
96	0xbe	97	0x7b	98	0xbc	99	0x99	100	0x63	101	0x94	102	0x5f	103	0x2a
104	0x61	105	0xb8	106	0x34	107	0x32	108	0x19	109	0xfd	110	0xfb	111	0x17
112	0x40	113	0xe6	114	0x51	115	0x1d	116	0x41	117	0x44	118	0x8f	119	0x29
120	0xdd	121	0x04	122	0x80	123	0xde	124	0xe7	125	0x31	126	0xd6	127	0x7f
128	0x01	129	0xa2	130	0xf7	131	0x39	132	0xda	133	0x6f	134	0x23	135	0xca
136	0xfe	137	0x3a	138	0xd0	139	0x1c	140	0xd1	141	0x30	142	0x3e	143	0x12
144	0xa1	145	0xcd	146	0x0f	147	0xe0	148	0xa8	149	0xaf	150	0x82	151	0x59
152	0x2c	153	0xf5	154	0x7d	155	0xad	156	0xb2	157	0xef	158	0xc2	159	0x87
160	0xce	161	0x75	162	0x06	163	0x13	164	0x02	165	0x90	166	0x4f	167	0x2e
168	0x72	169	0x33	170	0x85	171	0xc0	172	0x8d	173	0xcf	174	0xa9	175	0x81
176	0xe2	177	0xc4	178	0x27	179	0x2f	180	0x6c	181	0x7a	182	0x9f	183	0x52
184	0xe1	185	0x15	186	0x38	187	0x2b	188	0xfc	189	0x20	190	0x42	191	0xc7
192	0x08	193	0xe4	194	0x09	195	0x55	196	0x5e	197	0x8c	198	0x14	199	0x76
200	0x60	201	0xff	202	0xdf	203	0xd7	204	0x98	205	0xfa	206	0x0b	207	0x21
208	0x00	209	0x1a	210	0xf9	211	0xa6	212	0xb9	213	0xe8	214	0x9e	215	0x62
216	0x4c	217	0xd9	218	0x91	219	0x50	220	0xd2	221	0xee	222	0x18	223	0xb4
224	0x07	225	0x84	226	0xea	227	0x5b	228	0xa4	229	0xc8	230	0x0e	231	0xcb
232	0x48	233	0x69	234	0x4b	235	0x4e	236	0x9c	237	0x35	238	0x79	239	0x45
240	0x4d	241	0x54	242	0xe5	243	0x25	244	0x3c	245	0x0c	246	0x4a	247	0x8b
248	0x3f	249	0xcc	250	0xa7	251	0xdb	252	0x6b	253	0xae	254	0xf4	255	0xe3

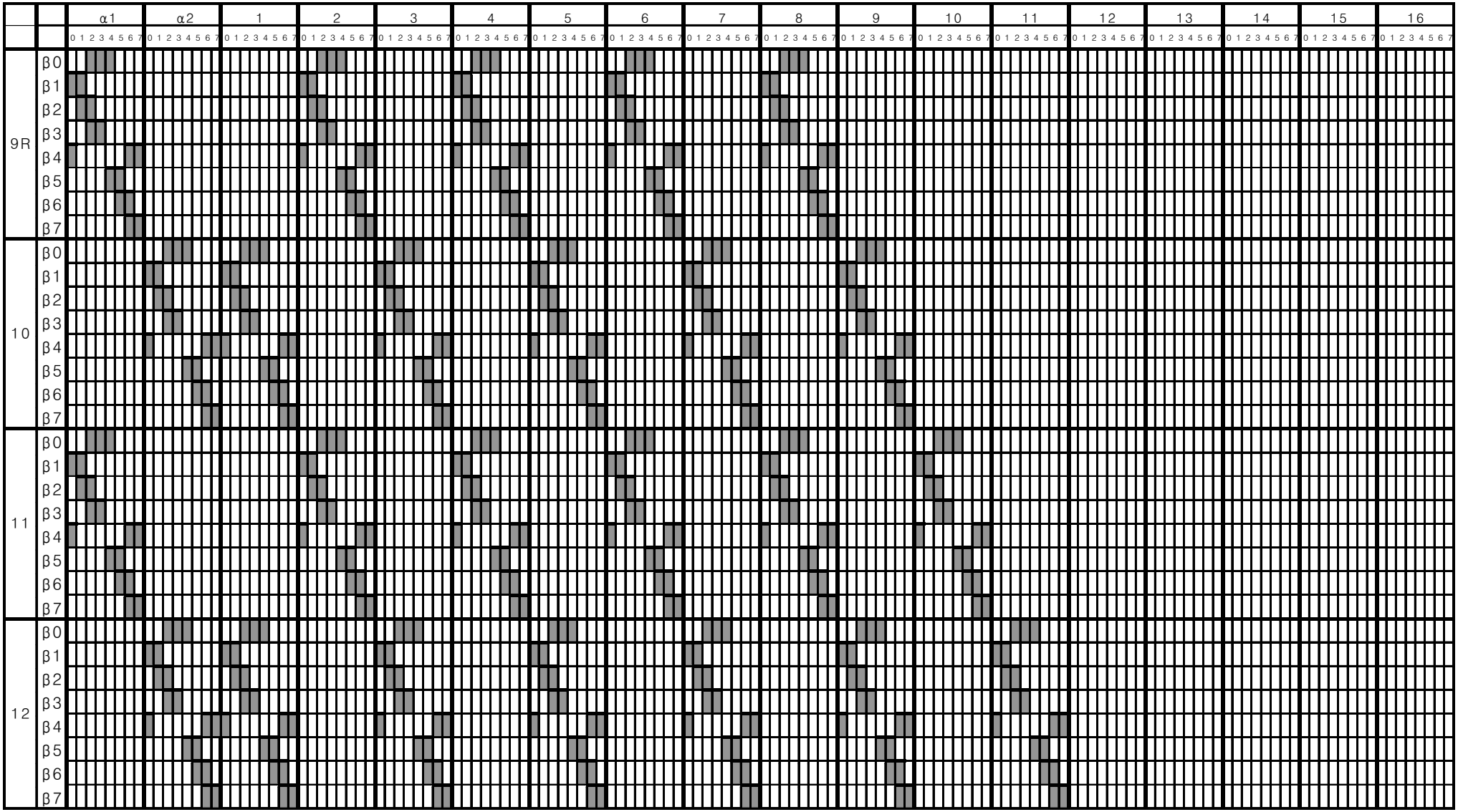
C.1 Sbox-2

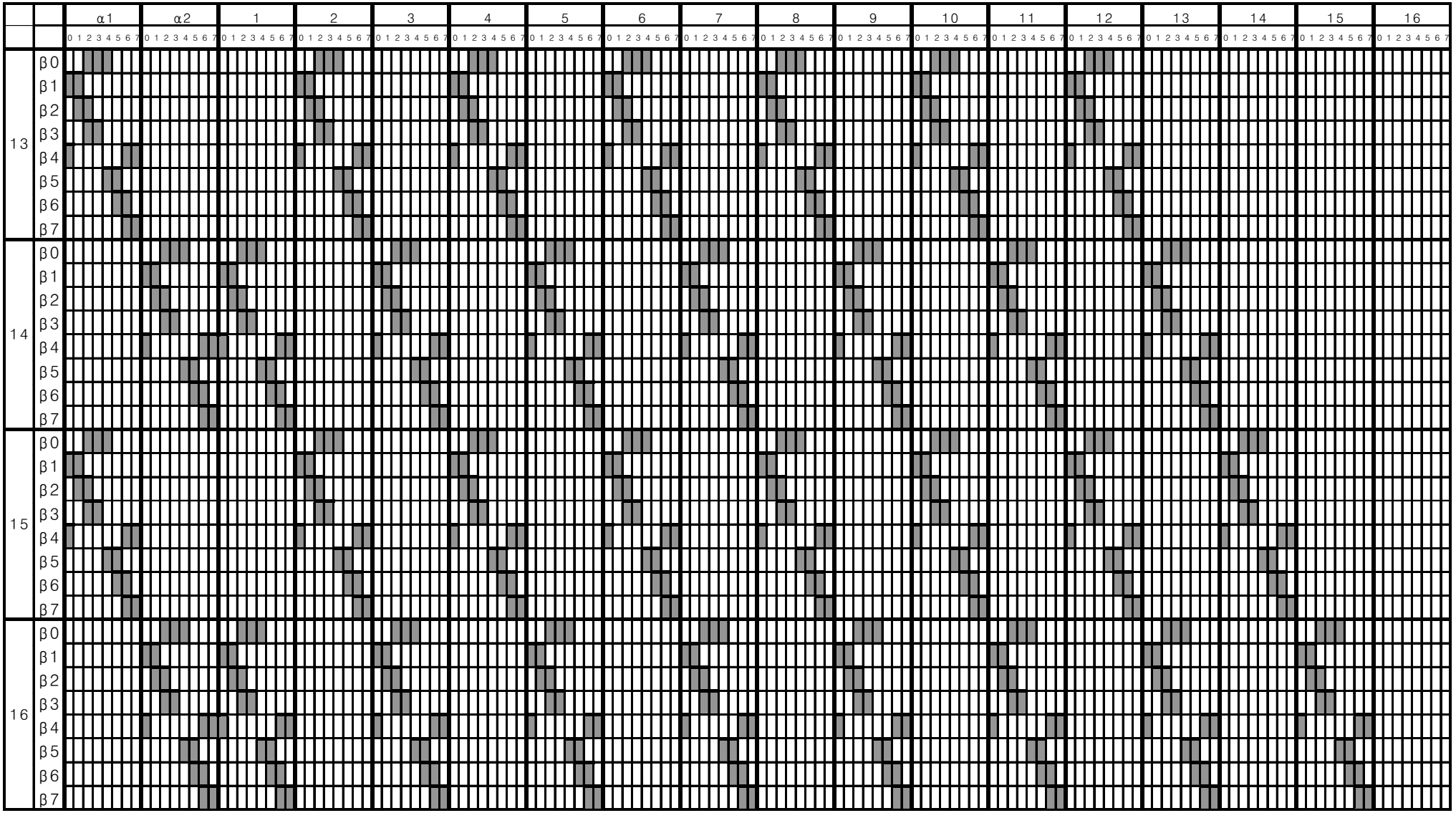
x	S2(x)	x	S2(x)	x	S2(x)	x	S2(x)	x	S2(x)	x	S2(x)	x	S2(x)	x	S2(x)
0	0x12	1	0x4a	2	0x26	3	0xc8	4	0xd2	5	0x62	6	0xce	7	0xe7
8	0x2e	9	0xc3	10	0xfb	11	0x7c	12	0x65	13	0x48	14	0x8f	15	0xb8
16	0x76	17	0x3d	18	0xa5	19	0x8e	20	0x86	21	0x57	22	0xbd	23	0xbc
24	0x1f	25	0xef	26	0x0c	27	0xe0	28	0x78	29	0x71	30	0x11	31	0x75
32	0x95	33	0xd9	34	0x9b	35	0x9e	36	0xb9	37	0xa4	38	0xf7	39	0x02
40	0x7f	41	0x80	42	0x83	43	0x7e	44	0xbe	45	0x56	46	0x96	47	0x73
48	0x9f	49	0x88	50	0x2a	51	0x14	52	0x89	53	0x9a	54	0xf9	55	0xdc
56	0x32	57	0x6d	58	0xde	59	0x6a	60	0x84	61	0x72	62	0xd8	63	0x8a
64	0xd7	65	0xe3	66	0x08	67	0x4e	68	0x1e	69	0xb3	70	0x5d	71	0x50
72	0xd6	73	0xeb	74	0xb1	75	0x0d	76	0xcf	77	0xad	78	0xc6	79	0x0e
80	0x7d	81	0xa0	82	0xdd	83	0x9c	84	0x41	85	0x1c	86	0xcd	87	0x1a
88	0x38	89	0x34	90	0x5b	91	0x23	92	0x03	93	0x8c	94	0x68	95	0x46
96	0x53	97	0x04	98	0xa9	99	0x27	100	0xac	101	0xe6	102	0x1b	103	0xfc
104	0x2f	105	0xa3	106	0x0b	107	0x28	108	0xe4	109	0x0f	110	0xda	111	0xd4
112	0xc4	113	0xd5	114	0x94	115	0x8b	116	0x90	117	0x6b	118	0x9d	119	0xf8
120	0xae	121	0x63	122	0x7a	123	0x07	124	0xe2	125	0xea	126	0xc5	127	0xdb
128	0x98	129	0x15	130	0xc1	131	0x0a	132	0xa2	133	0xc2	134	0x30	135	0x44
136	0x5a	137	0xf1	138	0x3a	139	0x6e	140	0xa8	141	0xc9	142	0x55	143	0x4d
144	0x20	145	0x6f	146	0xf2	147	0x35	148	0x59	149	0x19	150	0x77	151	0xbb
152	0x92	153	0x6c	154	0x2c	155	0x45	156	0x66	157	0x42	158	0xf3	159	0x39
160	0x29	161	0xc0	162	0xe8	163	0x4f	164	0xe5	165	0xc7	166	0xb0	167	0xe1
168	0x8d	169	0xf6	170	0x00	171	0x01	172	0x7b	173	0xd1	174	0xcb	175	0x52
176	0xfd	177	0xcc	178	0x58	179	0x3f	180	0xee	181	0xb2	182	0xff	183	0x40
184	0xaa	185	0x4b	186	0x74	187	0xb4	188	0x60	189	0x5f	190	0x99	191	0x2b
192	0x91	193	0xdf	194	0xf4	195	0x47	196	0x21	197	0x3b	198	0x33	199	0x93
200	0xaf	201	0xd3	202	0x16	203	0x5e	204	0x36	205	0x43	206	0x49	207	0xa6
208	0xd0	209	0x06	210	0xb6	211	0x70	212	0x81	213	0x82	214	0xa1	215	0xfa
216	0x97	217	0x85	218	0x79	219	0xb7	220	0xba	221	0x3c	222	0x10	223	0xb5
224	0xab	225	0x13	226	0xa7	227	0x64	228	0xe9	229	0x09	230	0x54	231	0x25
232	0x37	233	0x67	234	0x1d	235	0xfe	236	0xf5	237	0x69	238	0x2d	239	0x31
240	0x22	241	0xf0	242	0x18	243	0x3e	244	0x61	245	0x17	246	0x51	247	0xec
248	0x05	249	0xca	250	0xed	251	0x5c	252	0x87	253	0xbf	254	0x4c	255	0x24

Appendix-B: Transient Differential Matrix


α1 α2 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
0 1 2 3 4 5 6 7
1R 2R 3R 4R
β0 β1 β2 β3 β4 β5 β6 β7


α1 α2 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
0 1 2 3 4 5 6 7
5R
6R
7R
8R
β0 β1 β2 β3 β4 β5 β6 β7


α1 α2 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
0 1 2 3 4 5 6 7
9R
10
11
12
β0 β1 β2 β3 β4 β5 β6 β7


α1 α2 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
0 1 2 3 4 5 6 7
13 14 15 16
β0 β1 β2 β3 β4 β5 β6 β7

Appendix-C: Statistical Randomness Test Result

The following tables represent the success ratio in percentage for each test.

C.1 Test results for random plaintexts

Test	Level of significance	Round							
		2	4	6	8	10	12	14	16
Frequency	5%	95.2	95.0	95.1	95.1	95.4	95.1	94.8	95.1
	1%	99.0	99.1	99.2	98.9	99.2	98.9	99.0	99.0
	0.1%	99.9	99.9	99.9	99.9	99.9	99.9	99.9	99.9
Serial	5%	94.5	95.2	94.9	94.9	95.0	95.5	95.0	95.1
	1%	98.7	99.0	99.1	99.0	99.1	99.2	98.9	99.0
	0.1%	99.9	99.9	99.9	99.9	99.9	99.9	99.9	100.0
Auto correlation	5%	94.8	95.5	95.3	95.0	94.9	95.0	94.6	95.2
	1%	98.9	99.0	98.9	99.9	99.0	99.0	99.0	99.0
	0.1%	99.9	99.9	99.9	99.9	99.9	99.9	99.9	99.9
Change point	5%	90.1	90.0	90.5	90.7	89.9	89.9	90.4	91.0
	1%	98.0	98.0	98.0	98.2	98.0	98.1	98.1	98.2
	0.1%	99.8	99.9	99.8	99.8	99.8	99.8	99.8	99.8
Binary derivative	5%	95.1	94.6	95.2	94.9	94.5	95.1	94.8	94.5
	1%	99.0	99.0	99.0	98.9	98.7	99.1	99.1	99.0
	0.1%	99.9	100.0	100.0	99.9	99.9	99.9	99.9	100.0
Poker (4bit)	5%	95.2	95.2	95.2	95.4	94.9	95.0	95.1	94.7
	1%	99.0	99.1	99.1	99.1	98.9	99.1	99.1	99.0
	0.1%	99.9	99.9	99.9	99.9	99.9	99.9	99.9	99.9
Poker (8bit)	5%	95.2	95.0	94.9	94.7	94.9	94.7	95.3	95.4
	1%	99.0	99.0	99.0	98.8	99.1	98.9	99.0	99.0
	0.1%	99.9	99.9	99.9	99.9	99.9	99.9	99.9	99.9
Runs	5%	95.0	94.8	95.1	94.9	94.8	95.3	95.0	94.8
	1%	99.0	98.9	99.0	99.0	98.9	99.1	99.1	99.0
	0.1%	99.9	99.9	99.9	99.9	99.9	99.9	99.9	99.9
Runs distribution	5%	91.1	90.8	91.1	90.6	90.9	90.9	91.0	91.6
	1%	97.8	97.5	97.5	97.2	97.5	97.5	97.6	97.6
	0.1%	99.6	99.7	99.7	99.6	99.5	99.6	99.6	99.6
Linear complexity	5%	93.7	93.4	93.7	93.5	93.5	93.5	93.8	94.0
	1%	96.6	96.6	96.7	96.8	96.6	96.7	97.1	97.0
	0.1%	99.2	99.1	99.3	99.3	99.1	99.1	99.3	99.2
Universal	5%	95.2	94.7	94.3	93.5	94.4	95.5	94.3	94.7
	1%	98.6	99.0	99.1	98.8	98.8	98.7	99.0	99.2
	0.1%	99.9	99.9	100.0	100.0	99.7	99.8	99.8	100.0
Sequence complexity	Average	790.3	790.3	790.3	790.3	790.3	790.3	790.3	790.3
	Success	100	100	100	100	100	100	100	100
Coupon collector's	5%	95.4	95.5	94.9	95.0	96.4	95.6	94.9	95.3
	1%	98.8	98.9	98.6	99.1	99.2	99.5	99.3	99.0299.9
	0.1%	99.7	99.8	100.0	100.0	100.0	100.0	100.0	
Gap	5%	93.8	93.9	93.9	93.9	93.7	93.5	93.5	94.0
	1%	98.0	97.8	98.1	97.8	97.9	97.9	98.1	97.9
	0.1%	99.4	99.5	99.5	99.5	99.6	99.5	99.6	995
Collision		100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

C.2 Test results for plaintext/cipher text correlation

Test	Level of significance	Round							
		2	4	6	8	10	12	14	16
Frequency	5%	94.9	95.1	95.0	95.7	95.1	95.1	95.1	95.2
	1%	99.1	98.9	99.0	99.2	99.2	98.9	98.9	99.0
	0.1%	99.9	99.9	99.9	99.9	99.9	99.9	99.8	99.9
Serial	5%	95.0	94.9	95.1	94.8	94.5	95.0	95.1	95.1
	1%	99.2	99.0	99.1	99.1	98.9	99.1	99.0	99.0
	0.1%	99.9	99.0	99.9	99.9	99.9	99.9	99.9	99.9
Auto correlation	5%	95.0	95.1	94.7	94.9	95.2	94.8	94.8	95.0
	1%	99.0	99.0	98.9	99.0	99.0	99.0	98.9	99.0
	0.1%	99.9	99.9	99.9	99.9	99.9	99.9	100.0	99.9
Change point	5%	90.1	90.0	89.9	90.0	89.9	90.2	90.4	90.8
	1%	98.2	97.9	97.9	98.2	98.2	98.1	98.1	98.4
	0.1%	99.9	99.8	99.8	99.8	99.8	99.8	99.8	99.8
Binary derivative	5%	95.2	95.7	95.0	94.7	95.3	95.2	94.9	94.8
	1%	99.2	99.1	99.9	99.0	99.1	98.9	98.8	98.8
	0.1%	100.0	99.9	99.9	99.9	100.0	99.9	99.9	100.0
Poker (4bit)	5%	94.7	94.9	94.8	94.8	94.9	95.1	95.3	94.6
	1%	98.9	98.9	98.9	99.1	98.8	99.0	99.0	99.1
	0.1%	99.9	99.9	99.9	99.9	99.9	99.8	100.0	99.9
Poker (8bit)	5%	95.1	95.2	95.1	95.2	95.1	95.0	95.2	94.8
	1%	99.1	99.1	98.8	99.1	99.0	99.0	99.2	98.9
	0.1%	99.9	100.0	99.9	99.9	99.9	99.9	99.9	99.9
Runs	5%	94.8	94.9	95.3	95.1	95.0	95.6	95.3	95.0
	1%	98.9	99.0	99.2	99.1	99.0	99.1	99.0	98.9
	0.1%	99.9	99.9	100.0	100.0	99.9	99.9	99.9	99.9
Runs distribution	5%	91.2	90.7	91.1	91.5	90.8	91.1	91.0	91.1
	1%	97.8	97.4	97.4	97.6	97.4	97.5	97.4	97.6
	0.1%	99.7	99.6	99.6	99.5	99.6	99.7	99.6	99.6
Linear complexity	5%	93.4	94.1	94.1	94.1	93.8	94.0	93.6	93.7
	1%	96.7	96.9	97.0	97.1	979.0	97.1	96.8	96.8
	0.1%	99.1	99.3	99.3	99.3	99.1	99.2	99.2	99.2
Universal	5%	95.4	94.8	94.6	95.1	94.7	95.4	95.2	93.6
	1%	99.0	99.2	99.0	98.8	98.6	99.0	99.1	98.6
	0.1%	99.8	99.9	100.0	100.0	99.7	100.0	99.9	100.0
Sequence complexity	Average	790.3	790.3	790.3	790.3	790.2	790.3	790.3	790.3
	Success	100	100	100	100	100	100	100	100
Coupon collector's	5%	94.6	94.4	95.1	96.0	94.3	95.3	95.1	94.5
	1%	99.0	99.1	99.2	99.5	98.7	99.4	98.3	98.9
	0.1%	99.9	99.9	100.0	99.9	99.9	100.0	99.8	100.0
Gap	5%	94.1	93.5	93.6	94.0	93.4	93.7	93.1	94.0
	1%	98.4	97.8	98.1	98.1	98.1	97.9	97.8	98.1
	0.1%	99.7	99.5	99.6	99.5	99.6	99.5	99.5	99.6
Collision		100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

C.3 Test results for low density plaintext

Test	Level of significance	Round							
		2	4	6	8	10	12	14	16
Frequency	5%	21.1	97.7	98.4	95.3	96.1	93.0	93.0	97.7
	1%	31.1	100.0	100.0	98.4	100.0	99.2	96.9	98.4
	0.1%	39.3	100.0	100.0	100.0	100.0	100.0	98.4	100.0
Serial	5%	7.8	98.4	94.5	91.4	91.4	95.3	93.0	94.5
	1%	10.9	100.0	99.2	99.2	97.7	100.0	99.2	98.4
	0.1%	14.8	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Auto correlation	5%	73.4	93.8	93.0	96.9	94.5	94.5	96.9	93.8
	1%	86.7	98.4	99.2	99.2	99.2	99.2	99.2	97.7
	0.1%	95.3	100.0	100.0	100.0	100.0	100.0	100.0	99.2
Change point	5%	3.9	93.0	90.6	93.8	93.8	87.5	89.8	89.1
	1%	14.1	99.2	99.2	98.4	97.7	96.1	97.7	97.7
	0.1%	28.1	100.0	100.0	100.0	100.0	100.0	100.0	99.2
Binary derivative	5%	25.8	93.8	92.2	95.3	96.1	94.5	93.8	97.7
	1%	29.7	96.9	98.4	99.2	99.2	98.4	98.4	100.0
	0.1%	38.3	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Poker (4bit)	5%	0.0	92.0	92.0	97.0	96.0	95.0	94.0	96.0
	1%	0.0	97.0	97.0	99.0	98.0	100.0	99.0	100.0
	0.1%	0.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Poker (8bit)	5%	0.0	96.0	95.0	93.0	91.0	97.0	994.0	96.0
	1%	0.0	100.0	99.0	96.0	95.0	100.0	99.0	100.0
	0.1%	0.0	100.0	100.0	100.0	99.0	100.0	100.0	100.0
Runs	5%	23.4	89.8	94.5	93.8	98.4	95.3	96.1	95.3
	1%	29.7	97.7	100.0	100.0	100.0	98.4	98.4	99.2
	0.1%	32.8	99.2	100.0	100.0	100.0	100.0	100.0	100.0
Runs distribution	5%	0.0	93.0	96.9	94.5	94.5	94.5	95.3	95.3
	1%	0.8	99.2	99.2	100.0	97.7	97.7	97.7	99.2
	0.1%	0.8	100.0	100.0	100.0	99.2	98.4	100.0	100.0
Universal	5%	0.0	92.0	92.0	95.0	98.0	93.0	97.0	92.0
	1%	0.0	99.0	97.0	99.0	100.0	100.0	100.0	99.0
	0.1%	0.0	99.0	99.0	100.0	100.0	100.0	100.0	100.0
Coupon collector's	5%	59.0	94.0	93.0	95.0	95.0	97.0	96.0	98.0
	1%	72.0	100.0	100.0	100.0	100.0	100.0	99.0	100.0
	0.1%	77.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Gap	5%	5.0	93.0	94.0	92.0	95.0	95.0	95.0	97.0
	1%	7.0	100.0	99.0	100.0	99.0	99.0	98.0	100.0
	0.1%	17.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Collision		0.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

C.4 Test results for high density plaintext

Test	Level of significance	Round							
		2	4	6	8	10	12	14	16
Frequency	5%	23.4	92.2	93.0	96.1	94.5	97.7	93.8	96.9
	1%	28.1	97.7	97.7	98.4	98.4	100.0	99.2	100.0
	0.1%	35.9	99.2	100.0	100.0	100.0	100.0	99.2	100.0
Serial	5%	3.9	92.2	93.8	94.5	94.5	96.1	93.0	93.8
	1%	6.3	99.2	99.2	100.0	98.4	99.2	97.7	100.0
	0.1%	10.9	100.0	100.0	100.0	100.0	100.0	98.4	100.0
Auto correlation	5%	79.7	93.8	93.8	96.1	96.9	95.3	93.8	97.7
	1%	88.3	99.2	99.2	99.2	99.2	98.4	98.4	99.2
	0.1%	98.4	100.0	99.2	100.0	100.0	99.2	99.2	100.0
Change point	5%	5.5	93.8	92.2	86.7	90.6	87.5	90.6	88.3
	1%	14.1	100.0	99.2	97.7	97.7	99.2	97.7	97.7
	0.1%	27.3	100.0	100.0	100.0	100.0	100.0	100.0	99.2
Binary derivative	5%	30.5	94.5	94.5	94.5	93.8	96.1	95.3	97.7
	1%	42.2	100.0	97.7	99.2	98.4	98.4	99.2	98.4
	0.1%	50.8	100.0	100.0	100.0	100.0	99.2	100.0	100.0
Poker (4bit)	5%	0.0	92.0	96.0	95.0	94.0	94.0	97.0	98.0
	1%	0.0	97.0	100.0	99.0	98.0	98.0	98.0	99.0
	0.1%	0.0	100.0	100.0	100.0	100.0	100.0	99.0	100.0
Poker (8bit)	5%	0.0	98.0	94.0	96.0	97.0	95.0	100.0	99.0
	1%	0.0	100.0	100.0	99.0	99.0	99.0	100.0	100.0
	0.1%	0.0	100.0	100.0	100.0	100.0	99.0	100.0	100.0
Runs	5%	24.2	96.1	96.1	96.9	94.5	94.5	96.9	91.4
	1%	28.9	100.0	97.7	99.2	99.2	98.4	99.2	98.4
	0.1%	32.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Runs distribution	5%	0.0	95.3	93.0	89.8	94.5	96.1	93.0	90.6
	1%	0.8	99.2	99.2	96.1	100.0	99.2	98.4	96.9
	0.1%	0.8	100.0	100.0	100.0	100.0	100.0	99.2	100.0
Universal	5%	0.0	95.0	93.0	94.0	94.0	96.0	95.0	98.0
	1%	0.0	100.0	98.0	99.0	98.0	100.0	99.0	99.0
	0.1%	0.0	100.0	99.0	100.0	100.0	100.0	100.0	100.0
Coupon Collector's	5%	63.0	96.0	94.0	95.0	95.0	95.0	92.0	93.0
	1%	72.0	98.0	98.0	97.0	97.0	99.0	99.0	100.0
	0.1%	77.0	100.0	100.0	99.0	100.0	100.0	100.0	100.0
Gap	5%	10.0	96.0	95.0	95.0	92.0	96.0	97.0	95.0
	1%	15.0	98.0	99.0	99.0	97.0	99.0	100.0	99.0
	0.1%	24.0	100.0	100.0	100.0	98.0	100.0	100.0	100.0
Collision		7.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

REFERENCE

[1] 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 (1994), pp.111-130.

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

[3] Xuejia Lai. On the Design and Security of Block Ciphers, ETH Series in Information Processing vol.1, Hartung-Gorre Verlag Konstanz.

[4] K. Nyberg, Perfect Nonlinear S-boxes, Advances in Cryptology – EUROCRYPT'91, Lecture Notes in Computer Science 547, Springer-Verlag 1991, pp.378-385.

[5] C. Carlet. Partial Bent Functions, Advances in Cryptology – CRYPTO'92, Lecture Notes in Computer Science, Springer-Verlag, 1993.

[6] K. Nyberg. Differentially Uniform mappings for Cryptography, Advances in Cryptology – EUROCRYPT'93, Lecture Notes in Computer Science 765, Springer-Verlag 1994, pp.55-64.

[7] L.R. Knudsen. Truncated and higher order differentials, In B. Preneel, editor, Fast Software Encryption – Sencon International Workshop, Leuven, Belgium, LNCS 1008, pp.196-211. Springer-Verlag, 1995.

[8] W. Meier and O. Staffelbach. Nonlinearity Criteria for Cryptographic Functions, Advances in Cryptology – EUROCRYPT'89, Lecture Notes in Computer Science, Springer-Verlag 1990, pp.549-562.

[9] T. Jakobsen and L. R. Knudsen. The Interpolation Attack on Block Ciphers, Preproceedings of Fast Software Encryption. pp.28-40. Springer-Verlag, 1997.

[10] E. Biham and A. Shamir, Differential cryptanalysis of the data encryption standard, Springer-Verlag, 1993.

[11] C. M. Adams. Constructing symmetric ciphers using the CAST design procedure. Designs, Codes and Cryptography, 12(3): 283-316, November 1997.

[12] X. Lai, J. Massey and S. Murphy, Markov ciphers and differential cryptanalysis, Proceedings of EUROCRYPT'91, pp.17-38.

[13] M. Naor and O. Reingold, On the construction of pseudo-random permutations: Luby-Rackoff Revisited, Proceedings of the 29'th ACM Symposium on Theory of Computings, 1997, pp.189-199.

[14] J. Daemen, L. Knudsen, and V. Rijmen, The Block Cipher Square, Fast Software Encryption, 4th International Workshop Proceedings, Springer-Verlag, 1997, pp.149-165.

[15] J. Kelsey, B. Schneier, and D. Wagner, Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES, Advances in Cryptology – CRYPTO'96, Proceedings, Springer-Verlag, 1996, pp.237-251.

[16] X. Lai. Higher Order Derivations and Differential Cryptanalysis, Communications and Cryptography: Two Sides of One Tapestry, Kluwer Academic Publishers, 1994. pp.227-233.

[17] B. Preneel, V. Rijmen, A. Bosselaers, Recent Developments in the Design of Conventional Cryptographic Algorithms, State of the Art and Evolution of Computer Security and Industrial Cryptography, Lecture Notes in Computer Science, Springer-Verlag, 1998.

[18] L. Brown, M. Kwan, J. Pieprzyk, and J. Seberry. Improving Resistance to Differential Cryptanalysis and the Redesign of LOKI. Advances in Cryptology – ASIACRYPT'91, Volume 739 of Lecture Notes in Computer Science, pp.36-50, Springer-Verlag, 1993.

[19] L. R. Knudsen. A Key-Schedule Weakness in SAFER K-64. Advances in Cryptology – CRYPTO'95, Volume 963 of Lecture Notes in Computer Science, pp. 274-286. Springer-Verlag, 1995.

[20] K. Nyberg. Linear Approximation of Block Ciphers. Advances in Cryptology – EUROCRYPT'94, Volume 950 of Lecture Notes in Computer Science, pp. 439-444. Springer-Verlag, 1995.