Xenon^{TM} 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 Xenon^{TM} 1.0

PART-2: Design Concepts and Architecture for Xenon^{TM} 1.0

This is the second one of our two block cipher proposals named ‘Zodiac’ and ‘Xenon’.

Version 1.0

1. Design Philosophy for Xenon^{TM} 1.0

1.1 Design Concepts for Xenon^{TM} 1.0

In the design of Xenon, 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.
- **High Performance and Adaptive on Small memory device:** Our cipher has been designed to get high performance on general 32-bit CPU and designed to have no tabularized memory for round function such as S-boxes.

1.1.1. Simplicity

In an exterior view of Xenon, it adopts the Feistel-type network structure. The round function of Xenon was simply constructed by using only the basic 32-bit CPU operators such as exclusive-or (XOR), 32-bit adder, 32-bit shifter, and 32-bit multiplier. There were not used any pre-computed tables like S-box. The round function, however, was constructed to acquire low differential characteristic and low linear characteristic, and so it is very robust against differential cryptanalysis and linear cryptanalysis. The key schedule of Xenon is also very simple, it uses 128-bit linear feed-back shift register (LFSR) generated by a primitive trinomial over GF(2) of degree 128. But the LFSR can be implemented on word-by-word processing, so its key-setup can be done by high speed.

1.1.2. Provability

As a little bit of mention in the above section, the round-function structure of Xenon was made of the simply constructed Phi-function, and the function has very high differential uniformity (differential complexity) and high linear complexity in itself. So its 16 round iteration gives a very low total-differential-probability (or very low squared-linear probability) to the proposed cipher Xenon. Of course it is very hard (in fact is impossible) to make/extract a unified group operation covered over the purposed Phi-function of Xenon, and so the interpolation attack is suppressed. The detailed analysis is presented in section 3.1.

1.1.3. High Performance and Adaptive on small memory device

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

- **Direct CPU-operations without table look-up:** All operations of the round function of Xenon are XOR, 32-bit adder, 32-bit shifter, and 32-bit multiplier. But we tried to design its round function so that any S-box is not involved and the simply made round function has some robustness both on differential cryptanalysis and linear cryptanalysis. In fact, the Phi-function used in the Xenon round function has very high differential uniformity (differential complexity) and high linear complexity in itself. This is presented in section 3.1 in detail.

- **Confirmed differential and linear characteristics propagation:** We tried to supply some confirmed evidence on the differential characteristic and linear characteristic for the Phi-function that is used to construct the round function. Moreover we tried to find the minimum active Phi-function over 16 rounds iteration and as an experimental result we got the number would be greater or equal to 16. This tells us that the achievable maximum differential characteristic of Xenon is bounded by 2^{-304}.

In practice, Xenon can be built with only the instruction delay of 16*4 XORs , 16*2 Adders, and 16*2 Multipliers (the half of them are constant multipliers) in H/W implementation of the full 16-round encryption/decryption and it takes 391 clock-cycles for one-block (128-bits) encryption/decryption on Pentium machine.

1.2. The Key-Scheduling of Xenon^{TM}

Xenon-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 Xenon 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 Xenon can work "On the fly", deriving each block of round key, as it is needed, with as little required memory as possible.

- **All data bits of seed-key affect on all round-key bits:** Xenon's round-key derivation scheme from the input seed-key was designed to meet such goal that all bits of seed-key must affects on all round-key bits 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. This is guaranteed by the used LFSR's primitiveness.

- **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 Xenon 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 Xenon^{TM} 1.0

2.1. Xenon Building Blocks

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

2.1.1. Network Structure of Xenon^{TM}

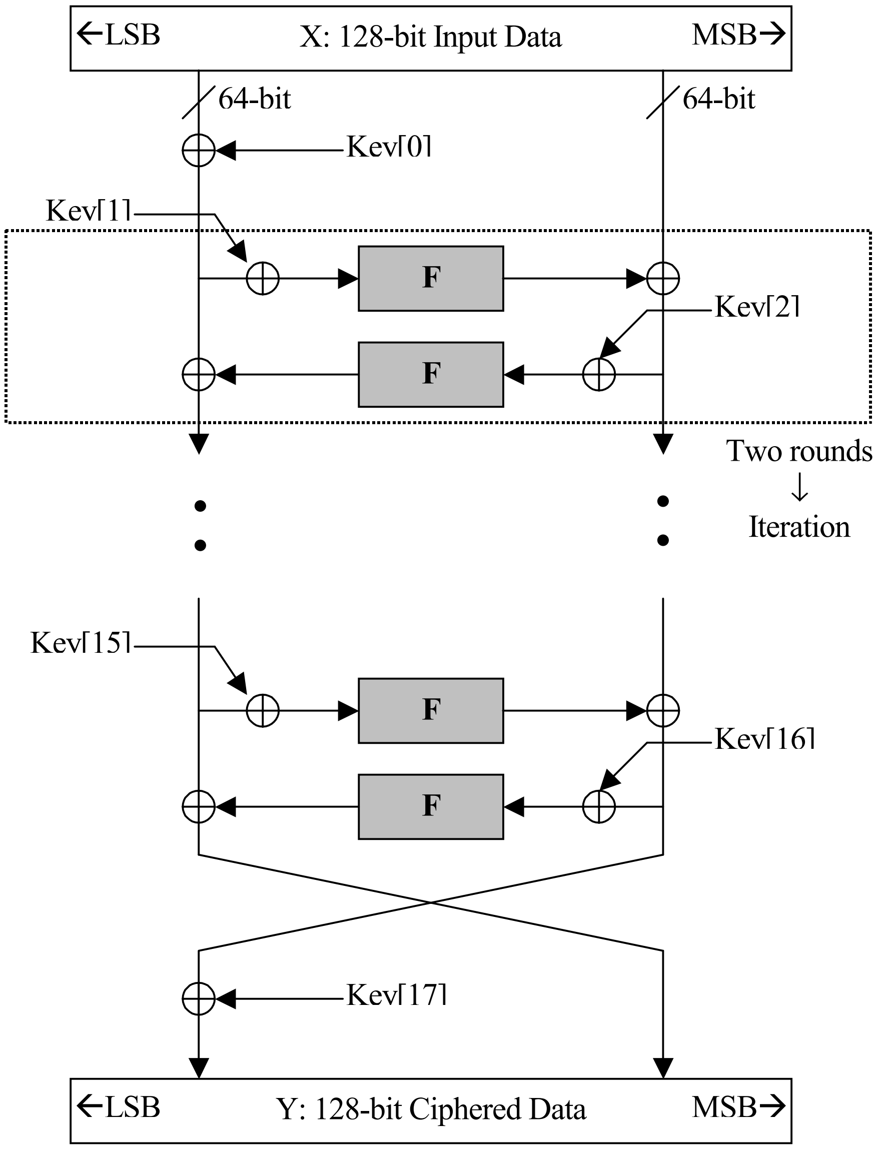


Figure 2.1: Feistel Network

Xenon 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.

2.1.2 The Round Function Structure

The round-function of Xenon has very simple structure and it is based on a non-linear Phi-function. The Phi-function consists of just word-by-word CPU basic instructions, such as 32-bit XOR, 32-bit adder, 32-bit shifter, and 32-bit multiplier.

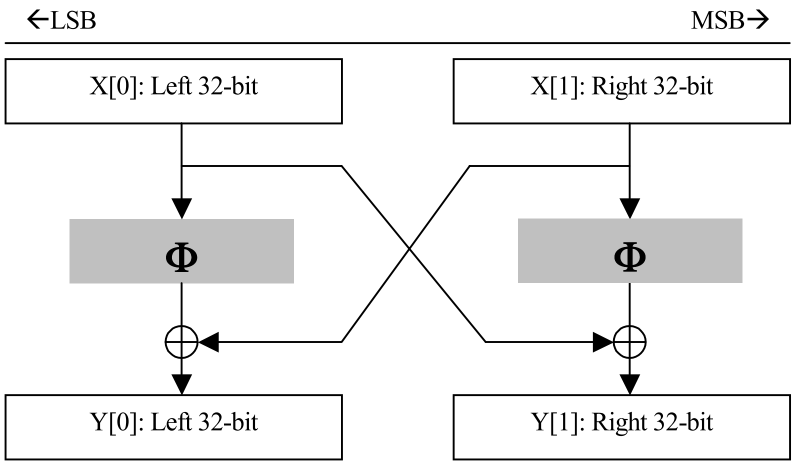


Figure 2.2: Round function of Xenon

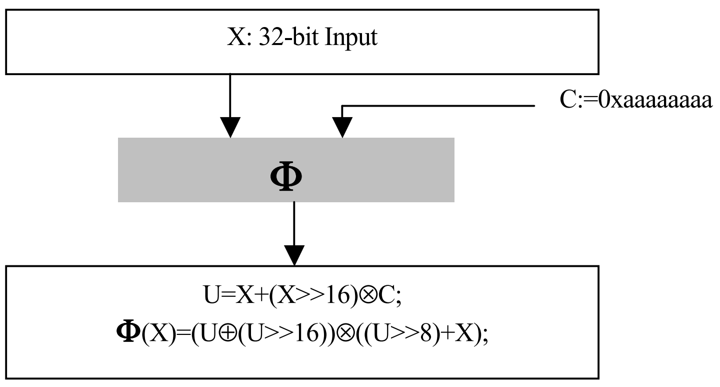


Figure 2.3: Φ-function of round function

In the lowest block of Fig.2.3, the add-operator means 32-add mod2^{32}, 32-bit multiplier means the usual 32-bit multiplier modular-2^{32}, and ⊕ means XOR operation. The shifter means data shifting to LSB direction.

2.2. Key Scheduling Algorithm of Xenon

The key-scheduling algorithm of Xenon was basically constructed from its round-function as you see in the Fig.2.4-(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.2.4-(1)),
- Step-2: round-key generating step with round-function (Fig.2.4-(2)),
- Step-3: role-changing step between the input-data-pad and key-pad (Fig.2.4-(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.2.4-(1)~(2), all data blocks are of 32-bit word.

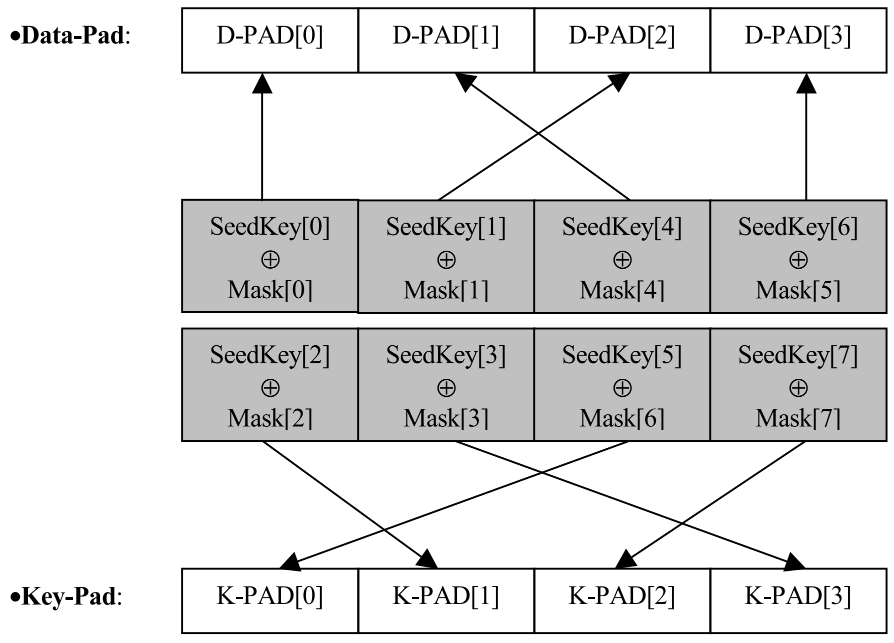
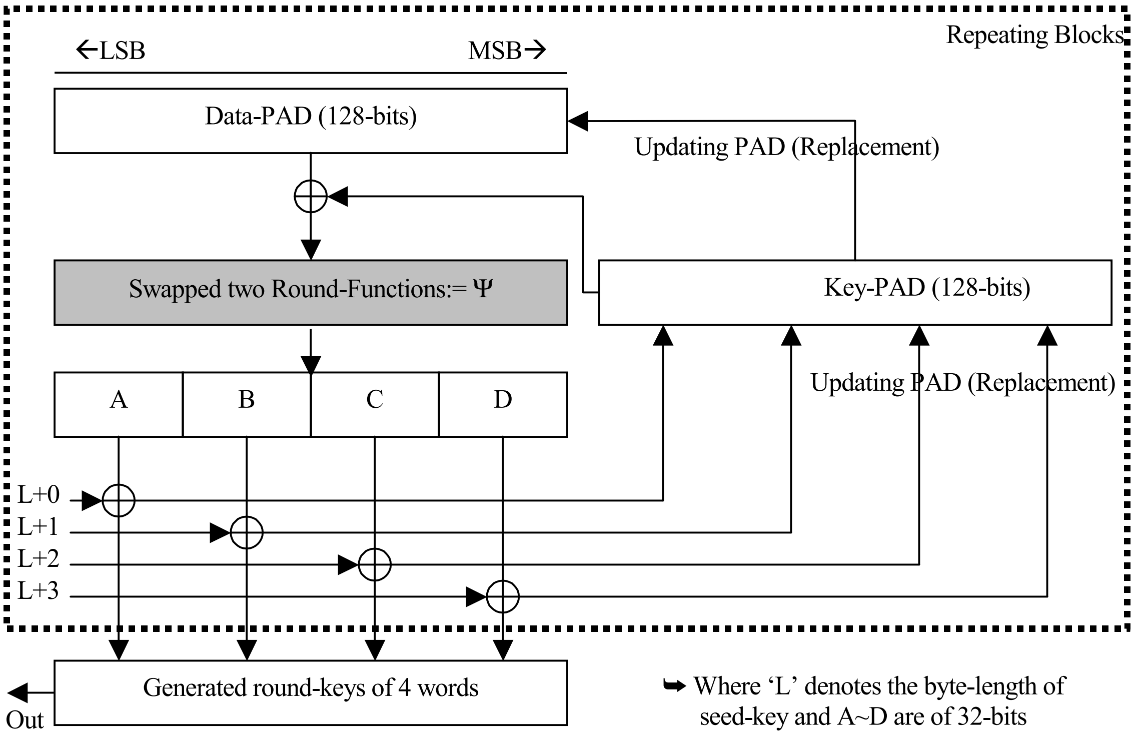


Figure 2.4-(1): Initial Seed-Key Loading on Data-Pads

The 32-bits masking constants, Mask[i]'s, appeared in Fig.2.4-(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.2.4-(2).



Repeating Blocks
←LSB
MSB→
Data-PAD (128-bits)
Updating PAD (Replacement)
Swapped two Round-Functions:= Ψ
Key-PAD (128-bits)
Updating PAD (Replacement)
A
B
C
D
L+0
L+1
L+2
L+3
Generated round-keys of 4 words
Out
➥ Where 'L' denotes the byte-length of seed-key and A~D are of 32-bits

Figure 2.4-(2): Round-key generation and Data-Updating steps

In that Fig.2.4-(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.2.4-(3)):

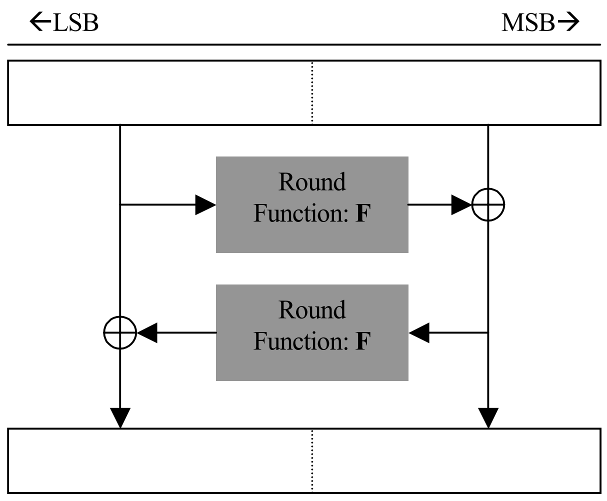


Figure 2.4-(3): The function Ψ

3. Security Consideration for Xenon^{TM}

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

3.1. Differential Cryptanalysis (in the differential characteristic sense)

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

With the notion that input words to the round function of Xenon affect on the adjacent input word or adjacent output word via the PHT-type permutation, this avalanche propagates to the next round, we know that the best minimization to decrease the number of active Phi-functions is to make all nontrivial input-difference words and their output-difference words 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 Phi-function, where the Phi-function 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 Phi-function is 16 with the 16 round Xenon.

Next we investigated the maximum differential characteristic of Phi-function and we got the following experimental results.

On GF(2^N)	N=8	N=12	N=16	N=18
$\mathrm{Max}_{(\alpha,\beta)}\mathrm{DC}(\Phi: \alpha\to\beta)$	$2^{-4.415}$	$2^{-8.00}$	$2^{-10.36}$	$2^{-11.83}$
$\mathrm{Max}_{(\alpha,0)}\mathrm{DC}(\Phi: \alpha\to 0)$	$2^{-5.00}$	$2^{-8.415}$	$2^{-12.20}$	$2^{-12.54}$
$\mathrm{Max}_{(\alpha,\alpha)}\mathrm{DC}(\Phi: \alpha\to\alpha)$	$2^{-5.00}$	$2^{-8.68}$	$2^{-11.83}$	$2^{-12.60}$
$\frac{N}{2}+X$:($\alpha\to\alpha$)	X=1	X=2.68	X=3.83	X=3.60

So we can conjectured that the maximum DC-characteristic of Phi-function is bounded by at most:

$$D(\Phi) < 2^{-(\frac{1}{2}\ln(X)+3)} \cong 2^{-19}.$$

Hence we get the total maximum DC-characteristic of 16 round Xenon is bounded by

$$D(Xenon) \le \left(2^{-19}\right)^{16} = 2^{-304}.$$

3.2. 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* $\phi_e^{(r)} + \phi_o^{(r)} = \prod_{i=1}^{r}((1-x_i)+x_i) = 1$ *and so* $2\phi_e^{(r)} - 1 = \phi_e^{(r)} - \phi_o^{(r)}$.

If r=2 then

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

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* $f_l \circ f_{l-1} \circ \ldots\ldots \circ f_2 \circ f_1$. *For the computation of total linear characteristic let* $\lambda, \omega \ne 0$

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:*

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

(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

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

This proves the theorem.

It has been pointed out that the number of active Phi-functions of 16 round Xenon cipher can be estimated as 16 in section 3.1. More over we know that, in the above theorem, the linear characteristic is bounded by

$$\Lambda(Xenon) \le \Lambda_{Xenon}{}^{16} + 2^{N-1}.$$

Where Λ_{Xenon} means the maximum linear characteristic of Phi-function of Xenon, and so the squared linear characteristic is bounded by

$$\Lambda^2(Xenon) \le \Lambda^2(\Phi)^{16} + 2^{-2(N-1)} \cong 2^{-304}.$$

3.3. Other Cryptanalysis

As previously mentioned, our cipher Xenon has the round function as generated from the Phi function and three different types of CPU-operations constructed the Phi-function. So one can not find any algebraic function that represents the Phi-function 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. Moreover we tried to make the key-schedule meet the *key avalanche property*.

4. Performance and Implementation Cost

4.1. Performance on Pentium Processor

On Pentium-III 450MHz, Xenon encrypts at 18.1 Mbytes/sec=144.8 Mbps/sec. The consumed total clock-cycles were estimated as the following and this result was made from performing the cycle-estimating codes to Xenon-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	Xenon	Mars	Serpent	TwoFish	Rijndale	RC6
128 bits						
Key-Setup	617 cycles	2066 cycles	1293 cycles	8560 cycles	227/1297 cycles	1705 cycles
128-bit BlockEnc	391 cycles	372 cycles	956 cycles	370 cycles	363 cycles	280 cycles
128-bit BlockDec	398 cycles	380 cycles	906 cycles	389 cycles	368 cycles	241 cycles
192 bits						
Key-Setup	613 cycles	2070 cycles	1304 cycles	11803 cycles	214/1513 cycles	2045 cycles
128-bit BlockEnc	391 cycles	372 cycles	969 cycles	370 cycles	435 cycles	280 cycles
128-bit BlockDec	398 cycles	380 cycles	906 cycles	389 cycles	429 cycles	241 cyclces
256 bits						
Key-Setup	617 cycles	2076 cycles	1298 cycles	15635 cycles	290/1831 cycles	1901 cycles
128-bit BlockEnc	391 cycles	372 cycles	952 cycles	370 cycles	500 cycles	280 cycles
128-bit BlockDec	398 cycles	380 cycles	906 cycles	389 cycles	496 cycles	241 cycles

4.2. Hardware Implementation Cost

Zodiac	Per one round			Delay per one round			Total Delay: Table-Lookup: 16*1, XOR:16*3 Sum=66 gate-delay	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	4	2	2		

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]	b1 cd 4b 11 15 46 44 50
Round-Key[1]	b4 e7 03 5c 15 5a 92 ca
Round-Key[2]	f0 a6 aa 8b 72 ab 5e d3
Round-Key[3]	d9 b3 7e 1b a8 13 63 00
Round-Key[4]	5f a1 15 13 14 1b 0b 51
Round-Key[5]	f8 fb 36 6c 2d fc c0 62
Round-Key[6]	bd ff 7c f2 94 7a 40 50
Round-Key[7]	31 24 cc cf d1 ea 9b db
Round-Key[8]	06 ba cb 2b 16 71 49 fb
Round-Key[9]	53 36 e1 92 e5 83 3d a1
Round-Key[10]	80 2f f4 5b f3 fc 59 8d
Round-Key[11]	b2 38 66 fe a0 4e 4f 00
Round-Key[12]	4d 10 0e 72 0b 05 08 2b
Round-Key[13]	e1 75 65 6b 30 a8 c6 70
Round-Key[14]	82 62 67 1f 90 b4 17 6b
Round-Key[15]	e9 af eb b1 a8 86 90 53
Round-Key[16]	43 dd 62 f3 e9 63 33 51
Round-Key[17]	8f 89 dc c7 57 11 88 87
Cipher Text	93 4b 1e dc dd 90 cc 06 e3 d5 80 4c 84 a9 cc 68

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]	b1 cd 4b 11 15 46 44 50
Round-Key[1]	b4 e7 03 5c 15 5a 92 ca
Round-Key[2]	f0 a6 aa 8b 72 ab 5e d3
Round-Key[3]	d9 b3 7e 1b a8 13 63 00
Round-Key[4]	5f a1 15 13 14 1b 0b 51
Round-Key[5]	f8 fb 36 6c 2d fc c0 62
Round-Key[6]	bd ff 7c f2 94 7a 40 50
Round-Key[7]	31 24 cc cf d1 ea 9b db
Round-Key[8]	06 ba cb 2b 16 71 49 fb
Round-Key[9]	53 36 e1 92 e5 83 3d a1
Round-Key[10]	80 2f f4 5b f3 fc 59 8d
Round-Key[11]	b2 38 66 fe a0 4e 4f 00
Round-Key[12]	4d 10 0e 72 0b 05 08 2b
Round-Key[13]	e1 75 65 6b 30 a8 c6 70
Round-Key[14]	82 62 67 1f 90 b4 17 6b
Round-Key[15]	e9 af eb b1 a8 86 90 53
Round-Key[16]	43 dd 62 f3 e9 63 33 51
Round-Key[17]	8f 89 dc c7 57 11 88 87
Cipher Text	09 f1 54 a0 61 d7 e0 29 13 2e 51 29 2c 28 9c 0f

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]	12 15 16 df 28 b6 00 bb
Round-Key[1]	08 ed 2f 7b 56 66 af 9d
Round-Key[2]	a6 b7 99 d0 d6 47 ad e1
Round-Key[3]	18 0a 34 4c 0f fd 02 8e
Round-Key[4]	e5 b4 73 98 91 2c da 43
Round-Key[5]	fa 6a cc 05 ee 2f 52 bb
Round-Key[6]	a0 1a b9 ea 8e 9e 92 27
Round-Key[7]	d9 d5 b0 4b a2 67 4c 7a
Round-Key[8]	ac d2 71 cb d6 d2 a5 a5
Round-Key[9]	fa 30 f5 66 32 fc 42 0e
Round-Key[10]	6c a1 ee a4 b9 9c ce 3f
Round-Key[11]	c5 e2 64 d2 63 ee f2 c9
Round-Key[12]	6e 78 3f df bc d2 14 59
Round-Key[13]	bb 12 42 ce 30 c5 b1 9a
Round-Key[14]	ef fd 4d 59 01 8e 44 d8
Round-Key[15]	63 3c 36 c8 b8 fe 81 8d
Round-Key[16]	b2 1b 08 cc 14 11 19 44
Round-Key[17]	9b 0d 81 86 52 8d aa 76
Cipher Text	1e 8f bc 8f 11 51 fc b9 16 b2 db ef 96 d3 6c 00

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]	12 15 16 df 28 b6 00 bb
Round-Key[1]	08 ed 2f 7b 56 66 af 9d
Round-Key[2]	a6 b7 99 d0 d6 47 ad e1
Round-Key[3]	18 0a 34 4c 0f fd 02 8e
Round-Key[4]	e5 b4 73 98 91 2c da 43
Round-Key[5]	fa 6a cc 05 ee 2f 52 bb
Round-Key[6]	a0 1a b9 ea 8e 9e 92 27
Round-Key[7]	d9 d5 b0 4b a2 67 4c 7a
Round-Key[8]	ac d2 71 cb d6 d2 a5 a5
Round-Key[9]	fa 30 f5 66 32 fc 42 0e
Round-Key[10]	6c a1 ee a4 b9 9c ce 3f
Round-Key[11]	c5 e2 64 d2 63 ee f2 c9
Round-Key[12]	6e 78 3f df bc d2 14 59
Round-Key[13]	bb 12 42 ce 30 c5 b1 9a
Round-Key[14]	ef fd 4d 59 01 8e 44 d8
Round-Key[15]	63 3c 36 c8 b8 fe 81 8d
Round-Key[16]	b2 1b 08 cc 14 11 19 44
Round-Key[17]	9b 0d 81 86 52 8d aa 76
Cipher Text	29 a7 c2 20 40 0f 55 1e 05 01 2d 3d 1d 31 1d 06

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]	5a 0f 43 12 01 7e 43 75
Round-Key[1]	2c f0 31 64 76 7f b5 86
Round-Key[2]	63 dd 38 4f 6e ed 3a e0
Round-Key[3]	39 3b 47 41 78 b5 3a 89
Round-Key[4]	14 a6 8e 52 98 4e 05 66
Round-Key[5]	08 c8 06 3a b4 60 c5 54
Round-Key[6]	d4 6a 9d ff 49 6f 59 94
Round-Key[7]	27 ec 02 69 a0 a9 ce 19
Round-Key[8]	71 9a 60 0b 15 6f 8a fb
Round-Key[9]	59 3e 2e 3e a2 c5 0a 31
Round-Key[10]	58 b8 54 a2 ee 6d d7 ac
Round-Key[11]	f3 68 21 75 a0 94 0f 66
Round-Key[12]	67 52 74 84 a3 44 36 7c
Round-Key[13]	3b ae 43 93 6b 34 27 32
Round-Key[14]	af a7 b6 90 31 84 55 df
Round-Key[15]	6f eb 56 26 3b 4b 93 5f
Round-Key[16]	5e 81 16 12 58 18 07 5d
Round-Key[17]	5e dc fe 88 eb 80 9d 6b
Cipher Text	2e ac f1 6a 1b 1f a3 83 41 eb 4d f3 16 45 76 17

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]	5a 0f 43 12 01 7e 43 75
Round-Key[1]	2c f0 31 64 76 7f b5 86
Round-Key[2]	63 dd 38 4f 6e ed 3a e0
Round-Key[3]	39 3b 47 41 78 b5 3a 89
Round-Key[4]	14 a6 8e 52 98 4e 05 66
Round-Key[5]	08 c8 06 3a b4 60 c5 54
Round-Key[6]	d4 6a 9d ff 49 6f 59 94
Round-Key[7]	27 ec 02 69 a0 a9 ce 19
Round-Key[8]	71 9a 60 0b 15 6f 8a fb
Round-Key[9]	59 3e 2e 3e a2 c5 0a 31
Round-Key[10]	58 b8 54 a2 ee 6d d7 ac
Round-Key[11]	f3 68 21 75 a0 94 0f 66
Round-Key[12]	67 52 74 84 a3 44 36 7c
Round-Key[13]	3b ae 43 93 6b 34 27 32
Round-Key[14]	af a7 b6 90 31 84 55 df
Round-Key[15]	6f eb 56 26 3b 4b 93 5f
Round-Key[16]	5e 81 16 12 58 18 07 5d
Round-Key[17]	5e dc fe 88 eb 80 9d 6b
Cipher Text	e2 dd b7 61 a7 d6 2b d8 06 2f 1d 48 51 ea 0a b7

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, $1 \le i \le 2^m$. 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 $\lfloor n/4 \rfloor, \ldots, \lfloor 3n/4 \rfloor$. 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 | $e_i > 5$, $i > 0$ }.

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

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 ith 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.

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

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

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, collect 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, collect 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-A

7. Intellectual Property Statement

We, SoftForum©, do not discriminate any use of and any user of this cipher, Xenon. 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©.

8. Maturity and Current Usage

Xenon^{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: Statistical Randomness Test Result

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

A.1. Test results for random plaintexts

Test	Level of significance	Round							
		2	4	6	8	10	12	14	16
Frequency	5%	95.17	94.99	94.8	94.98	95.11	95.21	94.98	95.09
	1%	98.91	99	98.87	99.06	99.01	99.01	98.92	98.93
	0.1%	99.95	99.93	99.91	99.95	99.97	99.96	99.96	99.94
Serial	5%	95.09	95.13	95.38	95.12	94.81	95.01	94.97	94.99
	1%	98.97	98.81	98.98	99.13	99.03	99.16	98.85	99.11
	0.1%	99.89	99.89	99.88	99.89	99.9	99.94	99.85	99.92
Auto correlation	5%	94.61	94.96	94.57	94.79	95.03	94.79	94.95	95.1
	1%	98.99	98.92	98.75	98.98	98.9	98.94	98.88	98.91
	0.1%	99.97	99.92	99.95	99.97	99.94	99.99	99.97	99.94
Change point	5%	89.64	90.16	90.21	90.04	90.07	90.11	89.77	90.43
	1%	97.95	97.94	98.2	98.21	98.1	98.04	98.14	98.14
	0.1%	99.83	99.73	99.89	99.85	99.8	99.8	99.77	99.74
Binary derivative	5%	95.2	95.06	95.04	94.73	95.19	94.6	95.01	94.69
	1%	99.04	99.05	98.98	98.97	98.84	99.08	98.94	99.01
	0.1%	99.94	99.92	99.94	99.97	99.93	99.97	99.93	99.96
Poker (4bit)	5%	95.44	95.39	94.85	94.94	94.71	95.11	94.69	94.72
	1%	99.02	99.1	99.11	98.91	98.92	98.92	98.97	98.9
	0.1%	99.89	99.93	99.9	99.93	99.91	99.86	99.9	99.91
Poker (8bit)	5%	94.6	95.4	94.9	95.2	95	95	95.8	94.8
	1%	98.5	98.9	99	99.2	99.2	99.3	99.1	98.5
	0.1%	100	99.9	99.7	99.9	100	99.9	100	99.8
Runs	5%	95.13	94.81	95.27	94.84	94.88	95.08	94.94	95.05
	1%	99	98.93	98.93	98.87	98.86	99.04	98.83	99.13
	0.1%	99.91	99.88	99.86	99.83	99.86	99.89	99.89	99.88

Runs distribution	5%	90.91	91.39	90.77	90.9	91.03	91.23	91.33	91.05
	1%	97.41	97.62	97.33	97.64	97.74	97.52	97.31	97.57
	0.1%	99.56	99.67	99.51	99.58	99.71	99.57	99.49	99.62
Linear complexity	5%	93.4	93.8	95.4	92.8	95.5	93.5	94.7	93.7
	1%	96.8	97.2	97.6	96.1	98.3	95.5	97.6	97.4
	0.1%	99.3	99.4	99.2	99.2	99.6	98.8	99.3	99.4
Universal	5%	95.2	94.6	95.4	95.4	94.8	95.6	95	95.9
	1%	98.9	98.6	99.1	99.1	98.2	99.2	98.9	98.9
	0.1%	100	99.9	99.9	99.7	99.7	99.8	99.9	99.9
Sequence complexity	Average	790.26	790.29	790.22	790.36	790.31	790.29	790.28	790.24
	Success	100	100	100	100	100	100	100	100

Level of significance	LC profile jump	LC profile jump height	Spectral	Coupon collector's	Gap	Collision
5%	96	97	99.9	94.78	93.47	100
1%	98	99	99.9	98.87	97.74	
0.1%	99	100	100	99.92	99.59	

A.2. Test results for plaintext/cipher text correlation

Test	Level of significance	Round							
		2	4	6	8	10	12	14	16
Frequency	5%	95.4	95.0	94.8	94.9	94.8	94.5	95.2	95.2
	1%	99.1	99.1	99.0	99.0	98.8	98.9	99.1	99.1
	0.1%	99.9	99.9	99.9	99.9	99.9	99.9	99.9	99.9
Serial	5%	95.1	95.0	95.2	94.7	95.0	95.0	95.2	95.0
	1%	98.9	99.0	98.9	98.9	99.0	98.8	99.1	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	94.8	95.0	95.0	94.9	95.1	94.5	95.0
	1%	98.9	98.9	99.1	99.0	99.0	99.0	98.9	99.1

	0.1%	99.9	99.9	100.0	99.9	99.9	99.9	99.9	99.9
Change point	5%	89.6	90.5	90.2	90.1	90.4	90.8	90.2	90.2
	1%	97.8	98.3	97.8	98.0	98.5	98.2	97.8	98.0
	0.1%	99.8	99.8	99.7	99.7	99.9	99.8	99.8	99.8
Binary derivative	5%	95.1	94.9	95.0	95.2	94.8	95.0	94.7	95.0
	1%	99.1	99.0	99.1	98.9	98.9	99.1	98.9	99.0
	0.1%	100.0	99.9	100.0	99.9	100.0	99.9	99.9	99.9
Poker (4bit)	5%	94.9	94.9	95.1	95.0	95.2	94.8	94.9	94.9
	1%	99.0	99.1	99.1	99.1	99.1	98.9	99.0	99.1
	0.1%	99.9	99.9	99.9	100.0	99.9	99.9	99.9	99.9
Poker (8bit)	5%	94.8	95.2	94.9	95.1	95.1	95.5	95.1	95.0
	1%	99.0	99.2	98.9	99.1	99.1	99.1	99.0	99.1
	0.1%	99.9	99.9	99.9	99.9	99.9	100.0	99.9	99.9
Runs	5%	95.2	95.1	95.1	94.5	95.2	95.2	95.0	95.1
	1%	99.1	99.0	99.0	99.0	99.0	99.0	99.1	99.1
	0.1%	99.9	99.9	99.9	99.9	99.9	100.0	99.9	99.9
Runs distribution	5%	91.4	91.0	90.6	91.0	90.7	91.5	91.1	91.0
	1%	97.6	97.6	97.3	97.8	97.4	97.9	97.6	97.6
	0.1%	99.7	99.6	99.5	99.7	99.6	99.6	99.6	99.7
Linear complexity	5%	93.7	93.9	94.0	93.7	93.7	93.5	93.6	94.2
	1%	96.9	96.9	96.9	97.1	97.0	96.8	96.8	97.0
	0.1%	99.2	99.2	99.2	99.3	99.2	99.3	99.2	99.4
Universal	5%	95.5	94.4	96.3	94.8	95.6	94.5	94.4	94.7
	1%	98.5	98.6	99.1	99.4	99.3	98.5	99.1	98.8
	0.1%	99.7	99.9	99.9	100.0	100.0	99.8	99.9	99.9
Sequence complexity	Average	790.3	790.2	790.3	790.3	790.3	790.3	790.3	790.2
	Success	100	100	100	100	100	100	100	100
Coupon	5%	94.0	94.1	94.0	94.9	95.2	94.8	95.7	94.8
	1%	99.3	98.3	98.8	98.8	98.9	98.8	99.0	98.5
	0.1%	100.0	100.0	100.0	99.6	99.7	99.7	99.9	99.9

Gap	5%	93.8	94.0	93.5	93.6	93.5	93.8	93.7	93.9
	1%	97.9	98.2	97.8	98.0	97.9	97.9	97.8	97.7
	0.1%	99.6	99.6	99.5	99.5	99.5	99.6	99.5	99.3
Collisio n		100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

A.3. Test results for low density plaintext

Test	Level of significance	Round							
		2	4	6	8	10	12	14	16
Frequency	5%	5.5	47.7	64.1	93.0	96.9	96.1	95.3	94.5
	1%	5.5	57.0	82.0	98.4	100.0	98.4	99.2	99.2
	0.1%	6.3	62.5	89.1	100.0	100.0	100.0	100.0	100.0
Serial	5%	0.0	37.5	78.9	96.1	95.3	94.5	95.3	93.0
	1%	0.0	46.1	85.9	98.4	98.4	100.0	100.0	97.7
	0.1%	0.0	57.0	91.4	100.0	100.0	100.0	100.0	100.0
Auto correlation	5%	10.9	85.9	96.9	97.7	94.5	92.2	94.5	98.4
	1%	14.1	95.3	99.2	100.0	99.2	98.4	97.7	100.0
	0.1%	15.6	99.2	100.0	100.0	100.0	98.4	100.0	100.0
Change point	5%	0.0	70.8	85.2	88.3	93.8	93.8	87.5	89.1
	1%	0.0	88.3	96.9	98.4	100.0	98.4	98.4	98.4
	0.1%	0.8	96.1	100.0	100.0	100.0	100.0	100.0	100.0
Binary derivative	5%	4.7	78.9	93.0	94.5	94.5	97.7	96.1	94.5
	1%	5.5	91.4	99.2	100.0	96.9	100.0	100.0	96.9
	0.1%	7.8	96.9	100.0	100.0	98.4	100.0	100.0	99.2
Poker (4bit)	5%	0.0	14.0	76.0	88.0	95.0	94.0	95.0	93.0
	1%	0.0	21.0	89.0	97.0	96.0	98.0	99.0	100.0
	0.1%	0.0	29.0	94.0	99.0	99.0	100.0	100.0	100.0
Poker (8bit)	5%	0.0	28.0	72.0	91.0	95.0	94.0	97.0	97.0
	1%	0.0	39.8	83.0	94.0	98.0	100.0	99.0	100.0
	0.1%	0.0	46.0	93.0	100.0	100.0	100.0	100.0	100.0
Runs	5%	3.9	82.8	93.0	98.4	93.0	94.5	97.7	96.9
	1%	4.7	93.0	98.4	99.2	100.0	100.0	100.0	99.2
	0.1%	7.8	98.4	99.2	100.0	100.0	100.0	100.0	100.0
Runs distribution	5%	0.0	59.4	89.1	88.3	94.5	93.0	98.4	95.3
	1%	0.8	72.7	96.1	96.9	99.2	99.2	100.0	97.7

	0.1%	0.8	83.6	99.2	99.2	100.0	100.0	100.0	100.0
Univers al	5%	0.0	54.0	96.0	96.0	97.0	95.0	93.0	97.0
	1%	0.0	75.0	99.0	99.0	100.0	100.0	97.0	98.0
	0.1%	0.0	93.0	100.0	100.0	100.0	100.0	100.0	100.0
Coupon Collecto r's	5%	0.0	97.0	96.0	95.0	94.0	93.0	98.0	96.0
	1%	1.0	99.0	100.0	99.0	100.0	99.9	99.0	100.0
	0.1%	2.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Gap	5%	0.0	92.0	98.0	99.0	92.0	95.0	96.0	96.0
	1%	0.0	99.0	100.0	100.0	99.0	100.0	100.0	100.0
	0.1%	0.0	99.0	100.0	100.0	99.0	100.0	100.0	100.0
Collisio n		0.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

A.4. Test results for high density plaintext

Test	Level of significance	Round							
		2	4	6	8	10	12	14	16
Frequency	5%	3.1	35.2	73.4	88.3	89.8	95.3	92.2	95.3
	1%	3.1	48.4	87.5	96.9	97.7	97.7	99.2	100.0
	0.1%	4.7	62.5	93.8	98.4	100.0	98.4	100.0	100.0
Serial	5%	0.0	42.2	79.7	96.1	95.3	90.6	90.6	98.4
	1%	0.0	51.6	88.3	98.4	99.2	100.0	98.4	100.0
	0.1%	0.0	66.4	96.1	99.2	100.0	100.0	99.2	100.0
Auto correlation	5%	12.5	92.2	93.0	93.8	92.2	95.3	97.7	96.9
	1%	14.8	97.7	99.2	97.7	99.2	99.2	99.2	100.0
	0.1%	19.5	100.0	99.2	100.0	100.0	100.0	100.0	100.0
Change point	5%	0.0	73.4	86.7	79.7	88.3	89.8	91.4	90.6
	1%	0.8	87.5	95.3	93.8	97.7	98.4	98.4	99.2
	0.1%	1.6	95.3	100.0	99.2	100.0	100.0	100.0	100.0
Binary derivative	5%	6.3	85.9	94.5	95.3	97.7	94.5	93.0	96.1
	1%	9.4	92.2	98.4	98.4	99.2	100.0	97.7	99.2
	0.1%	10.2	99.2	100.0	100.0	100.0	100.0	100.0	100.0
Poker (4bit)	5%	0.0	13.0	68.0	95.0	95.0	95.0	93.0	92.0
	1%	0.0	22.0	84.0	98.0	99.0	98.0	99.0	98.0
	0.1%	0.0	34.0	93.0	99.0	100.0	99.0	100.0	100.0
Poker (8bit)	5%	0.0	29.0	75.0	95.0	96.0	96.0	96.0	90.0
	1%	0.0	42.0	90.0	98.0	100.0	99.0	100.0	99.0
	0.1%	0.0	50.0	97.0	99.0	100.0	100.0	100.0	100.0
Runs	5%	1.6	82.8	93.0	93.0	93.0	93.0	93.0	96.1
	1%	2.3	92.2	96.9	99.2	98.4	98.4	100.0	100.0
	0.1%	3.1	97.7	100.0	99.2	100.0	100.0	100.0	100.0
Runs distribution	5%	0.0	53.9	89.1	90.6	93.8	92.2	96.9	92.2
	1%	0.8	67.2	96.1	96.9	98.4	100.0	100.0	97.7
	0.1%	0.8	78.1	98.4	98.4	100.0	100.0	100.0	100.0

Univers al	5%	0.0	63.0	93.0	91.0	97.0	94.0	97.0	96.0
	1%	0.0	82.0	99.0	98.0	100.0	100.0	99.0	100.0
	0.1%	0.0	93.0	100.0	99.0	100.0	100.0	100.0	100.0
Coupon Collecto r’s	5%	0.0	96.0	97.0	93.0	96.0	97.0	96.0	93.0
	1%	1.0	99.0	100.0	96.0	100.0	99.0	99.0	98.0
	0.1%	2.0	100.0	100.0	99.0	100.0	100.0	100.0	100.0
Gap	5%	0.0	93.0	95.0	94.0	94.0	97.0	92.0	90.0
	1%	0.0	97.0	99.0	98.0	100.0	99.0	99.0	99.0
	0.1%	0.0	100.0	100.0	99.0	100.0	99.0	99.0	99.0
Collisio n		0.0	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.