Simple yet Assured Tree Encryption (SATE)

A new secure yet simple block-cipher algorithm that uses Fistel Tree

Andre Susanto (13512028)
Department of Informatics Engineering
Institut Teknologi Bandung
Indonesia
13512028@std.stei.itb.ac.id

***Abstract*—Simplicity is one of the most important factors in selecting a cryptographic algorithm. It is so important that the AES algorithm winner, which is Rijndael, is less secure than its rivals. Simplicity is important characteristic as algorithms that have this property can be applied to wide area of applications cheaply. SATE, which stands for Simple yet Assured Tree Encryption, is a very simple encryption algorithm that is inspired by tree structure. The usage of tree structure can be implemented easily and cheaply using standard shift registers, yet it will give lots of complexity to the cipher text as it has lots of branches. Experiments show that SATE is a very strong cryptography algorithm with strong confusion-diffusion characteristic.**

Keywords—block cipher; simple cipher; tree cipher; fistel; fistel tree.

I. Introduction

Cryptography is a science of protecting valuable information so that only the authorized ones can read and modify the information. It has been used since Caesar era to protect his messages to his generals. It is also used during world wars to secure military communications. Nowadays, cryptography is one of the most important sciences in the world since its wide applications in computer security. As a consequence, people are racing to develop faster and more secure encryption algorithm.

There are two major considerations in choosing encryption algorithms [2]. The first one is (of course) the algorithms' security. That means the algorithms must be practically unbreakable in their projected lifespan. The second thing to be taken into account is the algorithms' efficiency, means they should be fast and simple so that we can easily implement them in hardware forms. These two substantial considerations also played important role on the selection of the Advanced Encryption Standard (AES).

On January 2, 1997, NIST made a formal call to cryptographic community around the world to submit their ciphers as candidates for the new encryption standard, which was the AES. The selection process was very complicated as the standard would set the winning algorithm to be implemented in wide area of applications across borders. There were three evaluation criteria for the algorithms: 1) Security, 2) Cost, and 3) Implementation characteristics [1].

According to an analysis that was done by Schneier et al, Rijndael algorithm, which is the chosen AES algorithm, was not the strongest algorithm among other five candidates of AES. In fact, it was just second to the weakest algorithm, which was RC6. On the other hand, there were MARS, Twofish, and Serpent which overwhelmed Rijndael algorithm. Figure 1 shows us details of safety factors that were calculated by Schneier et al.

Algorithm	Safety Factor
MARS	1.90
RC6	1.18
Rijndael	1.11 ~ 1.56
Serpent	3.56
Twofish	2.67

Fig. 1. AES candidates safety factors

Although algorithms security is very important, NIST choice of Rijndael made algorithm security is not a sole ruler in cryptographic world, yet cost and implementation characteristics are considered more important as long as we have a "secure enough" encryption algorithm.

In this paper, the writer explained a new block-cipher algorithm that is called Simple yet Assured Tree Encryption Algoritm (SATE). This algorithm used a combination between Fistel network structure and tree promotion algorithm to encrypt and decrypt information. This technique made writer's algorithm strong yet simple enough to be implemented as hardware or to be applied on low resource computing machines.

II. Theories

A. *Tree*

Tree is a data structure that consists of a root and its children (nodes, and leafs) that simulates tree structure. A tree cannot have a cycle.

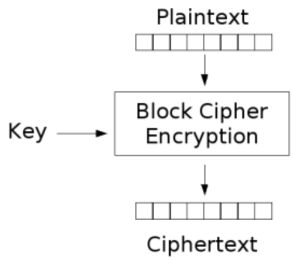
B. Block Ciphers

Block ciphers are encryption algorithms (*E* for encrypt and *D* for decrypt) that do their operations in unit of blocks. Basically, they divide plain texts into *P* (chunk of plain text) in which size they operate and encrypt them into *C* (chunk of cipher text), or:

$$C = E(P)$$

$$P = D(C)$$

Picture 1 shows us block ciphers encryption process. From the picture, we know that plain text came in a block. It then entered the block cipher and came out as a block of cipher text which had the same size as the input text.



Pic 1. Block cipher illustration (McCombe, 2007)

C. Block Cipher Operations

Block ciphers can be operated in various ways. There are four common block ciphers operation, which are:

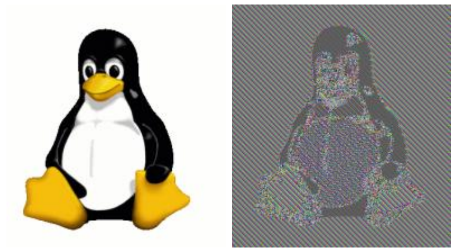
1. Electronic Code Book (ECB)

   By using this operation mode, for every *E*, *P*, and *C*, we will always have:

$$C_1 = E(P_1)$$

$$C_X = E(P_X)$$

   Therefore, if we have a plain text `01010` and *E (P)* is `10001`, we will always get the same result (`10001`) for every `01010` that we have. That is why this operation mode is called Electronic Code Book, as we can actually use a “code book” to determine *E (P)*. However, this operation method is considered a weak method. A statistical attack can be applied to block ciphers that operate in this mode. Another kind of attack by manipulating certain blocks also becomes one of this operation mode’s problems.

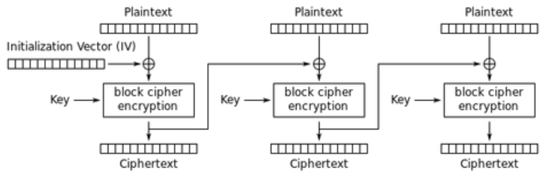


Pic 2. ECB Simple XOR (left: original)

   Picture 2 shows us simple xor encryption that operated in ECB mode. By looking at the picture, we know that this operation mode is not strong enough to protect our valuable information.

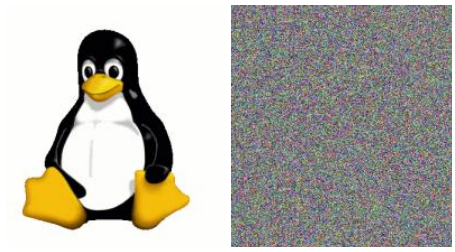
2. Cipher Block Chaining (CBC)

   As been mentioned by its name, this operation mode chains every *E* and *D* so that each block has dependency of the previous one. Picture 3 shows us how CBC mode chains every *E* and *D*.



Pic 3. CBC illustration (McCombe, 2007)

   There are several advantages by using this operation method, the most important are: 1) for every E, P, and C, we will not always have $C_X = E(P_X)$, means statistical analysis are tough to be applied. 2) Every modification to the cipher block will cause data error that could be detected easily, thus making attack by modifying certain blocks in cipher text is tough.

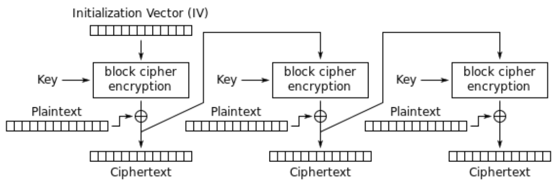


Pic 4. CBC Simple XOR (left: original)

   Picture 4 shows us simple xor encryption that operated in CBC mode. The result was way better. It has more randomness than the result of ECB.

3. Cipher Feedback (CFB)

   One main problem of using block cipher is when the information came in size that was not big enough to fit the block size. As the ciphers operate in unit of block, they could not do the encryption and had to wait for other information to fit the block size. CFB solves this problem by allowing ciphers to operate in smaller unit than their block size.

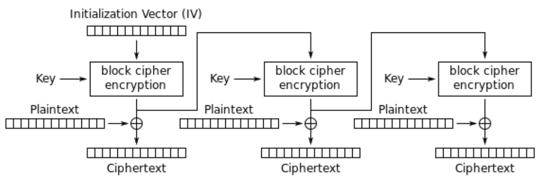


Pic 5. CFB illustration (McCombe, 2007)

Picture 5 shows us how CFB allow block ciphers operate in smaller units than their block size. CFB used Initialization Vector (IV) that would be encrypted by *E*. After that, CFB xored the *C'* from previous process with the plain text. The result, which was *C*, was used as a feedback to the next *E*.

4. Output Feedback (OFB)

   Output Feedback and Cipher Feedback basically use same method. The only difference is located at the feedback that is used by the next *E*. When CFB uses *C* as feedback to the next E, OFB uses *C'*, which is the result of *E(IV)*.



Pic 6. OFB illustration (McCombe, 2007)

Picture 6 shows us the how OFB works. The main differences between CFB and OFB can be seen if we compare Picture 5 and Picture 6.

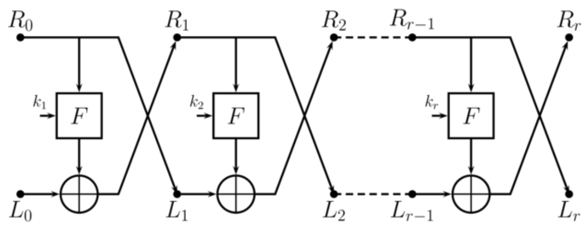
D. Fistel Network

Fistel Network is a reversible symmetric structure that is widely used in cryptographic community as it is used in DES, GOST, Lucifer, Blowfish, and other encryption algorithms. To construct a Fistel Network we can construct:

$$L_{i+1} = R_i$$

$$R_{i+1} = L_i \oplus F(R_i, K_i)$$

The advantage of using this structure is that *F* function does not have to be invertible. As a consequence, we can construct the *F* function as complex as we want to.



Pic 7. Fistel Structure

Picture 7 shows us how Fistel Network works. In the picture, input is divided into two parts, L and R.

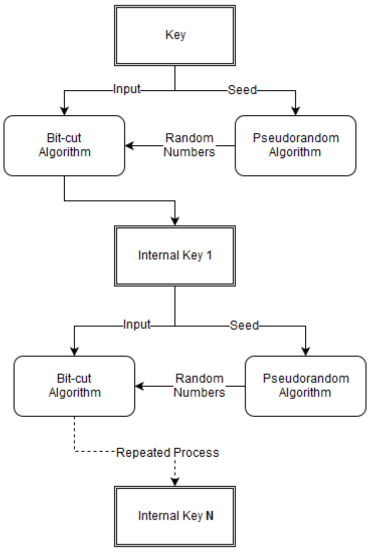
III. The Proposed Algorithm

A. Overview of Algorithm

The main purpose of SATE is providing simple way yet secure so that this algorithm can be implemented everywhere without worries of computing limitations. In order to achieve that, the writer uses tree structure which is simple to implement even in hardware by using shift registers. Although it is simple, it can cause a very big diffusion as it has branches to pick, and a little modification can lead to a very different result. The usage of Fistel Network complements the usage of tree by providing ability to use super complex function in each node that is guaranteed to be reversible (cipher text ←→ plain text). As for block size and key size, SATE uses a key that has same size as the block size. SATE supports the usage of several block size, such as 64, 128, 256, and 512 bits.

B. Internal Key Generation

Sate uses 15 internal keys that are placed in every nodes of the tree structure. In order to generate those 15 internal keys, SATE uses a pseudorandom generator and a bit-cut algorithm. The pseudorandom algorithm generates a sequence of random numbers from a seed (which is calculated from the previous generated key or original key if it is the first generation). Later, the generated numbers used to cut and shuffle the bits (as in shuffling cards).



Pic 8. Key Generation Algorithm

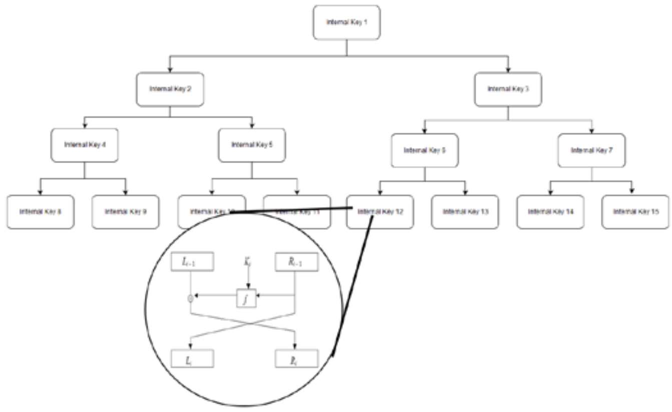
The shuffle algorithm that is used is as follows:

```
iterate i: 1 to 15:
  if (i = 0):
    random = Pseudorandom(key)
    internalKey[i] = key
  else:
    random = Pseudorandom (internalKey[i – 1])
    internalKey[i] = internalKey[i - 1]

  repeat (15 + random.nextInt() ) times:
    c1, c2 = cutBitAt ( random.nextInt() , internalKey[i])
    internalKey[i] = joinBit (c1, c2)
```

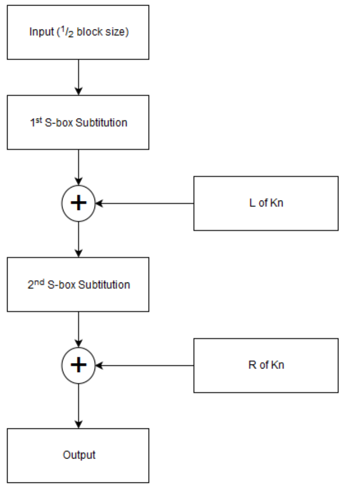
C. *Fistel-Tree Structure*

SATE uses a Fistel-Tree structure with 15 nodes. Each nodes contains a Fistel structure that uses an internal key that corresponds with its node id. To encrypt a block, the corresponding block will go from root to each leaf. As SATE uses 15 nodes, therefore we have 8 leafs. As a result, the block will go from root to leaf for 8 times (from root to leaf 1, root to leaf 2, and so on). Picture 9 shows us the detailed view of the Fistel-Tree structure.



Pic 9. Fistel Tree Structure

As for the F function, SATE uses two S-box substitutions and two xors. The first xor is done by using left part of internal key while the other one uses the right part of the internal key. Picture 10 shows a clear flow of F function that is used by SATE.



Pic 10. F Function

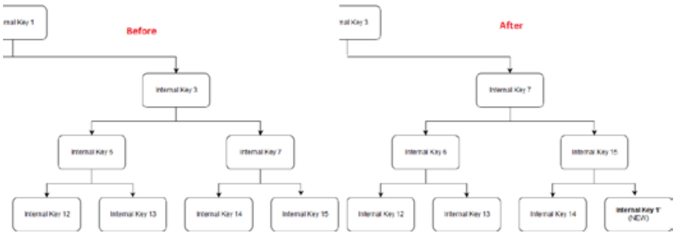
SATE uses the same S-box as Rijndael's. The substitutions process is also being done in the same way as Rijndael's. Picture 11 shows us the S-box that is used by SATE.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
x	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Pic 11. S-Box

D. *Leaf Promotion*

After a block is encrypted, SATE will modify its Fistel-Tree by promoting a leaf and placing a new leaf to fill the abandoned post. By using this method, the frequently used root of the tree is always replaced in every leaf promotion, thus making bigger diffusion impact on each modification. Picture 12 shows us the tree before and after the promotion process.



Pic 12. Leaf Promotion

The leaf promotion algorithm that is used is as follows:

newLeaf = **copy** (tree.root.bits)

i = *getNLastBits*(cipherText, 1)

if (i = 1):

  circularBitRotateRight(newLeaf, 2)

else:

  circularBitRotateLeft(newLeaf, 1)

n = *getNLastBits*(cipherText, 3)

promote (tree.leaf[n]) //recursive function also promote leaf's parent

tree.leaf[n] = newLeaf

IV. Experiments and Analysis

A. *Internal Key Generator*

The internal key generator module generates 15 internal keys from a key. The following shows us one of the results of internal key generation process that is done by SATE.

Original Key	0D 0F 0F 13 11 13 13 17 15 17 17 1B 19 1B 1B 1E
Internal Keys	1 0F 13 11 13 13 17 15 17 17 1B 19 1B 1B 1E 0D 0F
	2 11 13 13 17 15 17 17 1B 19 1B 1B 1E 0D 0F 0F 13
	3 13 17 15 17 17 1B 19 1B 1B 1E 0D 0F 0F 13 11 13
	4 15 17 17 1B 19 1B 1B 1E 0D 0F 0F 13 11 13 13 17
	5 17 1B 19 1B 1B 1E 0D 0F 0F 13 11 13 13 17 15 17
	6 19 1B 1B 1E 0D 0F 0F 13 11 13 13 17 15 17 17 1B
	7 1B 1E 0D 0F 0F 13 11 13 13 17 15 17 17 1B 19 1B
	8 0D 0F 0F 13 11 13 13 17 15 17 17 1B 19 1B 1B 1E
	9 0F 13 11 13 13 17 15 17 17 1B 19 1B 1B 1E 0D 0F
	10 11 13 13 17 15 17 17 1B 19 1B 1B 1E 0D 0F 0F 13
	11 13 17 15 17 17 1B 19 1B 1B 1E 0D 0F 0F 13 11 13
	12 15 17 17 1B 19 1B 1B 1E 0D 0F 0F 13 11 13 13 17
	13 17 1B 19 1B 1B 1E 0D 0F 0F 13 11 13 13 17 15 17
	14 19 1B 1B 1E 0D 0F 0F 13 11 13 13 17 15 17 17 1B
	15 1B 1E 0D 0F 0F 13 11 13 13 17 15 17 17 1B 19 1B

The writes did the experiments by using various types of keys and get very good results. The methods that the writer used did not generate any repeated internal keys. It also generated random-like internal keys which are good to protect valuable information.

B. *ECB Mode*

In this experiment we want to prove that our SATE algorithm still has a good diffusion rate to a small key change. To prove that, the writer set an experiment involving a text and two keys whose difference is only 1 bit. The first setup is as follow:

Key 1	0D 0F 0F 13 11 13 13 17 15 17 17 1B 19 1B 1B **1F**
Plain Text	01 4D D0 24 30 2C 10 8B 7F 39 16 1E 58 C8 C6 BC FA 88 F5 04 78 E4 D2 E8 49 67 7B 83 FD 05 C2 98 F8 4A 52 67 B0 83 F3 C9 EA E4 22 8A F3 C2 80 3A 19 2B AC 88 B6 82 AF E8 2A 84 22 AA 77 7F E3 79 36 0C B1 A1 D5 41 4E 8B 28 E6 9C 27 F1 80 1E 5B 55 C8 DC 5B F2 6C 5E 45 CB 63 3B C3 DC 87 C4 1E 84 2E F7 07 32 5C 2E 16 22 7D 35 D8 85 4C 6D 76 97 CF 54 22 38 CC B9 E3 3B 76 4C 57 20 B6 97 8C FE A3 6C CE 51 D8 0A CF 0E A9 A1 E8 74 BF 7C 58 25 83 6B CF AF BB CB F1 A3 7F 97 DE 7A 6D 2A B6 01 73 B7 8E A0 98 CA 90 B6 72 87 29 CA FC 7D E5 46 E0 A9 CD 72 56 45 63 A4 BC 35 BC 07 2E 8E CA 4D 3F E8 F0 78 E8 1C 7D 31 29 A1 8B 17 C1 45 7D 6C F9 20 30 28 D0 80 96 78 F7 AE 90 CC 96 96 A2 25 62 C4 34 24 4C BD BB 64 39 10 1A 66 D0 B0 A8 CE 7B 42 89 C4 92 E2 D8 2B 47 BE BF 9B CC AC 74
Cipher Text	00 4D D0 24 30 2C 10 8B 7F 39 16 1E 58 C8 C6 BC AA D8 05 F4 88 B4 82 B8 58 74 48 B0 CC 14 D3 89 FA 48 90 A1 7E ED 9D CF FF F3 15 B9 C2 D7 95 2F 99 AB EC CC FA 6B 46 69 BE 12 94 18 C5 E8 74 EC 77 4F 32 A6 D8 E9 E6 4B 6C A0 DA E5 33 47 D9 9E 71 EE 7A 7D D8 E3 D1 E4 EB 41 59 25 32 6C 2F BF 84 6C 35 45 34 FF 8D 93 8B D7 DF B6 F3 3E 1F 5E 83 99 AA 5C 02 7B 0E 72 B6 B8 CA 55 7A E0 C1 80 4A 55 32 10 8B 8E 5C FF A5 41 01 C8 4C 8B 48 76 F7 13 13 33 57 CF BF A3 09 96 37 FF 43 59 1E 98 52 62 4D F0 DA 6D 3F 43 9D 1A A4 8B 70 48 C9 4B 75 51 F3 13 68 C3 D0 D0 CE 15 D5 DD FE DB 7B 25 16 A6 9A 16 5A 45 E1 A6 FB A0 21 8A 8E 54 50 32 5E 08 7A FE 32 45 45 25 B2 7E AE 91 55 03 83 6D 57

	D3 5E 3A FE 18 B9 49 A6 B9 19 12 E6 5E BE 6C 6C 1A 28 F7 6E 96 B6 7A 99 B7 C7 E7 4B 92 72 60
Block size	128 bits
Input Size	2048 bits = 256 bytes = 16 blocks

Then, the writer made a small change to the key. The following is the result:

Key 2	0D 0F 0F 13 11 13 13 17 15 17 17 1B 19 1B 1B **1E**
Plain Text	01 4D D0 24 30 2C 10 8B 7F 39 16 1E 58 C8 C6 BC FA 88 F5 04 78 E4 D2 E8 49 67 7B 83 FD 05 C2 98 F8 4A 52 67 B0 83 F3 C9 EA E4 22 8A F3 C2 80 3A 19 2B AC 88 B6 82 AF E8 2A 84 22 AA 77 7F E3 79 36 0C B1 A1 D5 41 4E 8B 28 E6 9C 27 F1 80 1E 5B 55 C8 DC 5B F2 6C 5E 45 CB 63 3B C3 DC 87 C4 1E 84 2E F7 07 32 5C 2E 16 22 7D 35 D8 85 4C 6D 76 97 CF 54 22 38 CC B9 E3 3B 76 4C 57 20 B6 97 8C FE A3 6C CE 51 D8 0A CF 0E A9 A1 E8 74 BF 7C 58 25 83 6B CF AF BB CB F1 A3 7F 97 DE 7A 6D 2A B6 01 73 B7 8E A0 98 CA 90 B6 72 87 29 CA FC 7D E5 46 E0 A9 CD 72 56 45 63 A4 BC 35 BC 07 2E 8E CA 4D 3F E8 F0 78 E8 1C 7D 31 29 A1 8B 17 C1 45 7D 6C F9 20 30 28 D0 80 96 78 F7 AE 90 CC 96 96 A2 25 62 C4 34 24 4C BD BB 64 39 10 1A 66 D0 B0 A8 CE 7B 42 89 C4 92 E2 D8 2B 47 BE BF 9B CC AC 74
Cipher Text	40 5E 7B 8A 8F 86 EB 4A 11 14 0B 03 0D 29 37 46 BA 9B 5E AA C7 4E 29 29 27 4A 66 9E A8 E4 33 62 B8 59 F9 C9 0F 29 08 08 84 C9 3F 97 A6 23 71 C0 59 38 07 26 09 28 54 29 44 A9 3F B7 22 9E 12 83 76 1F 1A 0F 6A EB B5 4A 46 CB 81 3A A4 61 EF A1 15 DB 77 F5 4D C6 A5 84 A5 4E 26 DE 89 66 35 E4 C4 3D 5C A9 8D F6 D5 D7 4C 50 28 C5 D0 AD 9C 8C D7 DC FF 8C 87 66 42 22 55 5B 51 4A 75 57 66 76 BE B0 C7 60 EE 72 F1 0E 60 84 BC F5 21 5E 8D A2 65 90 C0 61 10 11 30 30 CD 52 8A C3 2F 8C DB 4C 41 60 1C 20 1F 32 31 51 D8 5F 9A 34 9F 1D 8C 1F 06 F3 02 63 CD FC BE A2 CA 91 28 A1 52 CF 7F 30 0D 2C 43 5E C7 42 E7 BC 5F 04 BC 96 42 20 B4 87 2C EA 8B 9E 97 7A 7B 57 16 DA B3 8D 99 77 67 58 65 71 6F 9A 9B E6 46 7A 0A 14 0D 07 33 31 41 52 8E 68 E9 27 7B 38 19 19 45 6A A3 A2 CE 2D 5D 8E
Block size	128 bits
Input Size	2048 bits = 256 bytes = 16 blocks

The result proves that even in ECB mode, small changes to the key will cause lots of changes to the cipher text. However, because there is no dependency between each E and D functions, if we made a small change to the cipher text, the only data that is affected is the one that we changed. Thus, this allows attackers to attack by changing some vital data. The experiment that explains this is as following:

Key	0D 0F 0F 13 11 13 13 17 15 17 17 1B 19 1B 1B **1F**
Plain Text	01 05 0B 13 1D 29 37 47 **59** 52 68 80 9A B6 D4 15 37 5B 66 8E B8 E4 33 63 95 C9 20 3D 77 B3 12 52 94 D8 3F 87 B6 23 71 C1 34 88 DE 57 B1 13 71 D1 54 B8 3F A7 32 9E 12 82 15 89 20 98 33 AF 4E B3 56 DA 81 2A B4 61 EF A0 38 CC 83 3C D6 93 52 F2 B5 5F 26 CE 99 66 35 E5 B8 8D 49 22 DC B9 98 79 5C 41 28 D5 C0 AD 9C 8D 80 75 6C 65 45 42 41 42 45 4A 51 5A 65 57 66 77 8A 9F B6 CF EA 28 2C 4D 70 95 BC E5 31 5E 8D A3 D6 2C 63 9C D7 35 74 B5 DD 43 8A D3 3F 8C DB 4D A0 DA 52 AB 27 84 E3 65 C8 4E 9A 24 8F 1D 8C 1E 91 27 9E 1D 98 36 B5 57 DA 80 28 B1 42 CF 7F 31 C4 7A 32 CB 87 2A C9 8B 4F 15 BC 86 52 20 B4 86 5A 30 E7 C1 9D 7B 5B 22 06 CB B3 9D 89 77 67 59 32 28 20 1A 16 14 14 16 1A 05 0D 17 23 31 41 53 67 7D 7A 94 B0 CE 0F 31 55 7B A3 B2 DE 2D 5D 8F C3 1A 52 8C AD 0C 4C 8E
Cipher Text	00 4D D0 24 30 2C 10 8B 7F 39 16 1E 58 C8 C6 BC AA D8 05 F4 88 B4 82 B8 58 74 48 B0 CC 14 D3 89 FA 48 90 A1 7E ED 9D CF FF F3 15 B9 C2 D7 95 2F 99 AB EC CC FA 6B 46 69 BE 12 94 18 C5 E8 74 EC 77 4F 32 A6 D8 E9 E6 4B 6C A0 DA E5 33 47 D9 9E 71 EE 7A 7D D8 E3 D1 E4 EB 41 59 25 32 6C 2F BF 84 6C 35 45 34 FF 8D 93 8B D7 DF B6 F3 3E 1F 5E 83 99 AA 5C 02 7B 0E 72 B6 B8 CA 55 7A E0 C1 80 4A 55 32 10 8B 8E 5C FF A5 41 01 C8 4C 8B 48 76 F7 13 13 33 57 CF BF A3 09 96 37 FF 43 59 1E 98 52 62 4D F0 DA 6D 3F 43 9D 1A A4 8B 70 48 C9 4B 75 51 F3 13 68 C3 D0 D0 CE 15 D5 DD FE DB 7B 25 16 A6 9A 16 5A 45 E1 A6 FB A0 21 8A 8E 54 50 32 5E 08 7A FE 32 45 45 25 B2 7E AE 91 55 03 83 6D 57 D3 5E 3A FE 18 B9 49 A6 B9 19 12 E6 5E BE 6C 6C 1A 28 F7 6E 96 B6 7A 99 B7 C7 E7 4B 92 72 60
Modified Cipher Text	00 4D D0 24 30 2C 10 8B 7F 39 16 1E 58 C8 C6 BC AA D8 05 F4 88 B4 82 B8 58 74 48 B0 CC 14 D3 89 FA 48 90 A1 7E ED 9D CF FF F3 15 B9 C2 D7 95 2F 99 AB EC CC FA 6B 46 69 BE 12 94 18 C5 E8 74 EC 77 4F 32 A6 D8 E9 E6 4B 6C A0 DA E5 33 47 D9 9E 71 EE 7A 7D D8 E3 D1 E4 EB 41 59 25 32 6C 2F BF 84 6C 35 45 34 FF 8D 93 8B D7 DF B6 F3 3E 1F 5E 83 99 AA 5C 02 7B 0E 72 B6 B8 CA 55 7A E0 C1 80 4A 55 32 10 8B 8E 5C FF A5 41 01 C8 4C 8B 48 76 F7 13 13 33 57 CF BF A3 09 96 37 FF 43 59 1E 98 52 62 4D F0 DA 6D 3F 43 9D 1A A4 8B 70 48 C9 4B 75 51 F3 13 68 C3 D0 D0 CE 15 D5 DD FE DB 7B 25 16 A6 9A 16 5A 45 E1 A6 FB A0 21 8A 8E 54 50 32 5E 08 7A FE 32 45 45 25 B2 7E AE 91 55 03 83 6D 57 D3 5E 3A FE 18 B9 49 A6 B9 19 12 E6 5E BE 6C 6C 1A 28 F7 6E 96 B6 7A 99 B7 C7 E7 4B 92 72 60
Resulting decryption	01 05 0B 13 1D 29 37 47 **58** 52 68 80 9A B6 D4 15 37 5B 66 8E B8 E4 33 63 95 C9 20 3D 77 B3 12 52 94 D8 3F 87 B6 23 71 C1 34 88 DE 57 B1 13 71 D1 54 B8 3F A7 32 9E 12 82 15 89 20 98 33 AF 4E B3 56 DA 81 2A B4 61 EF A0 38 CC 83 3C D6 93 52 F2 B5 5F 26 CE 99 66 35 E5 B8 8D 49 22 DC B9 98 79 5C 41 28 D5 C0 AD 9C 8D 80 75 6C 65 45 42 41 42 45 4A 51 5A 65 57 66 77 8A 9F B6 CF EA 28 2C 4D 70 95 BC E5 31 5E 8D A3 D6 2C 63 9C D7 35 74 B5 DD 43 8A D3 3F 8C DB 4D A0 DA 52 AB 27 84 E3 65 C8 4E 9A 24 8F 1D 8C 1E 91 27 9E 1D 98 36 B5 57 DA 80 28 B1 42 CF 7F 31 C4 7A 32 CB 87 2A C9 8B 4F 15 BC 86 52 20 B4 86 5A 30 E7 C1 9D 7B 5B 22 06 CB B3 9D 89 77 67 59 32 28 20 1A 16 14 14 16 1A 05 0D 17 23 31 41 53 67 7D 7A 94 B0 CE 0F 31 55 7B A3 B2 DE 2D 5D 8F C3 1A 52 8C AD 0C 4C 8E
Block size	128 bits
Input Size	2048 bits = 256 bytes = 16 blocks

The result shows us that only 1 byte that is affected by writer's small changes to the cipher text. The main reason is the usage of EBC mode which has no linking dependencies between each E and D function.

C. CBC Mode

In this experiment we want to prove that our SATE algorithm has a great diffusion impact over small changes of keys and/or data. Let us take a look into small key changes (only 1 bit):

Key 1	0D 0F 0F 13 11 13 13 17 15 17 17 1B 19 1B 1B **1F**
Plain Text	01 05 0B 13 1D 29 37 47 59 52 68 80 9A B6 D4 15 37 5B 66 8E B8 E4 33 63 95 C9 20 3D 77 B3 12 52 94 D8 3F 87 B6 23 71 C1 34 88 DE 57 B1 13 71 D1 54 B8 3F A7 32 9E 12 82 15 89 20 98 33 AF 4E B3 56 DA 81 2A B4 61 EF A0 38 CC 83 3C D6 93 52 F2 B5 5F 26 CE 99 66 35 E5 B8 8D 49 22 DC B9 98 79 5C 41 28 D5 C0 AD 9C 8D 80 75 6C 65 45 42 41 42 45 4A 51 5A 65 57 66 77 8A 9F B6 CF EA 28 2C 4D 70 95 BC E5 31 5E 8D A3 D6 2C 63 9C D7 35 74 B5 DD 43 8A D3 3F 8C DB 4D A0 DA 52 AB 27 84 E3 65 C8 4E 9A 24 8F 1D 8C 1E 91 27 9E 1D 98 36 B5 57 DA 80 28 B1 42 CF 7F 31 C4 7A 32 CB 87 2A C9 8B 4F 15 BC 86 52 20 B4 86 5A 30 E7 C1 9D 7B 5B 22 06 CB B3 9D 89 77 67 59 32 28 20 1A 16 14 14 16 1A 05 0D 17 23 31 41 53 67 7D 7A 94 B0 CE 0F 31 55 7B A3 B2 DE 2D 5D 8F C3 1A 52 8C AD 0C 4C 8E
Cipher Text	00 4D D0 24 30 2C 10 8B 7F 39 16 1E 58 C8 C6 BC AA D8 05 F4 88 B4 82 B8 58 74 48 B0 CC 14 D3 89 FA 48 90 A1 7E ED 9D CF FF F3 15 B9 C2 D7 95 2F 99 AB EC CC FA 6B 46 69 BE 12 94 18 C5 E8 74 EC 77 4F 32 A6 D8 E9 E6 4B 6C A0 DA E5 33 47 D9 9E 71 EE 7A 7D D8 E3 D1 E4 EB 41 59 25 32 6C 2F BF 84 6C 35 45 34 FF 8D 93 8B D7 DF B6 F3 3E 1F 5E 83 99 AA 5C 02 7B 0E 72 B6 B8 CA 55 7A E0 C1 80 4A 55 32 10 8B 8E 5C FF A5 41 01 C8 4C 8B 48 76 F7 13 13 33 57 CF BF A3 09 96 37 FF 43 59 1E 98 52 62 4D F0 DA 6D 3F 43 9D 1A A4 8B 70 48 C9 4B 75 51 F3 13 68 C3 D0 D0 CE 15 D5 DD FE DB 7B 25 16 A6 9A 16 5A 45 E1 A6 FB A0 21 8A 8E 54 50 32 5E 08 7A FE 32 45 45 25 B2 7E AE 91 55 03 83 6D 57 D3 5E 3A FE 18 B9 49 A6 B9 19 12 E6 5E BE 6C 6C 1A 28 F7 6E 96 B6 7A 99 B7 C7 E7 4B 92 72 60
Block size	128 bits
Input Size	2048 bits = 256 bytes = 16 blocks

Key 2	0D 0F 0F 13 11 13 13 17 15 17 17 1B 19 1B 1B **1E**
Plain Text	01 05 0B 13 1D 29 37 47 59 52 68 80 9A B6 D4 15 37 5B 66 8E B8 E4 33 63 95 C9 20 3D 77 B3 12 52 94 D8 3F 87 B6 23 71 C1 34 88 DE 57 B1 13 71 D1 54 B8 3F A7 32 9E 12 82 15 89 20 98 33 AF 4E B3 56 DA 81 2A B4 61 EF A0 38 CC 83 3C D6 93 52 F2 B5 5F 26 CE 99 66 35 E5 B8 8D 49 22 DC B9 98 79 5C 41 28 D5 C0 AD 9C 8D 80 75 6C 65 45 42 41 42 45 4A 51 5A 65 57 66 77 8A 9F B6 CF EA 28 2C 4D 70 95 BC E5 31 5E 8D A3 D6 2C 63 9C D7 35 74 B5 DD 43 8A D3 3F 8C DB 4D A0 DA 52 AB 27 84 E3 65 C8 4E 9A 24 8F 1D 8C 1E 91 27 9E 1D 98 36 B5 57 DA 80 28 B1 42 CF 7F 31 C4 7A 32 CB 87 2A C9 8B 4F 15 BC 86 52 20 B4 86 5A 30 E7 C1 9D 7B 5B 22 06 CB B3 9D 89 77 67 59 32 28 20 1A 16 14 14 16 1A 05 0D 17 23 31 41 53 67 7D 7A 94 B0 CE 0F 31 55 7B A3 B2 DE 2D 5D 8F C3 1A 52 8C AD 0C 4C 8E
Cipher Text	40 5E 7B 8A 8F 86 EB 4A 11 14 0B 03 0D 29 37 46 A4 91 54 A0 8D 45 37 37 29 40 6C 95 F2 FE 2D 6C B6 53 F3 C2 55 33 16 06 90 D9 6F 96 F2 37 61 C5 57 32 0D 2D 53 32 4A 27 78 81 57 AE 6E 93 0A 8C 66 1F 1A 0E 7A FA B5 5A 64 E9 E3 29 A2 67 E9 B0 2D C2 7F FD 45 DC 9D BC 97 7C 54 CC 9F 70 23 F4 EC 14 64 A0 95 FE DD CC 46 7B 52 DF CE A1 B2 A4 E7 CE CF AD B7 74 72 03 67 48 13 49 63 53 40 44 B6 9A CF 58 D6 68 C9 35 72 97 FE F4 17 78 AB B0 3D 8A F8 69 18 3D 58 5B A7 79 F0 EA 31 86 95 37 21 43 0C 20 1F 06 41 20 A4 64 D2 2F F3 05 D4 72 E1 52 90 E1 1F 4A 49 54 B6 AA 60 BA 3E D7 27 5D F2 95 C9 C7 3D DC 38 52 03 1F D4 8F 0E 18 CC E8 EB 6B 39 1E 65 EC AC A3 22 A8 A3 AC ED 67 37 1B DA 88 A5 32 41 58 B9 A3 06 5E 25 3F 4F 29 19 0B 29 8A 0B A7 89 9E FE D8 3B 12 BD 8C A4 27 17 C7

Block size	128 bits
Input Size	2048 bits = 256 bytes = 16 blocks

From the experiment result, we know that in CBC mode, even the slightest change (1 bit), change all the cipher text. So we can conclude that SATE has a very substantial diffusion impact over small changes. The same thing also applied if we modify even the smallest change of cipher text. For example:

Key	0D 0F 0F 13 11 13 13 17 15 17 17 1B 19 1B 1B 1F
Plain Text (Original)	01 05 0B 13 1D 29 37 47 59 52 68 80 9A B6 D4 15 37 5B 66 8E B8 E4 33 63 95 C9 20 3D 77 B3 12 52 94 D8 3F 87 B6 23 71 C1 34 88 DE 57 B1 13 71 D1 54 B8 3F A7 32 9E 12 82 15 89 20 98 33 AF 4E B3 56 DA 81 2A B4 61 EF A0 38 CC 83 3C D6 93 52 F2 B5 5F 26 CE 99 66 35 E5 B8 8D 49 22 DC B9 98 79 5C 41 28 D5 C0 AD 9C 8D 80 75 6C 65 45 42 41 42 45 4A 51 5A 65 57 66 77 8A 9F B6 CF EA 28 2C 4D 70 95 BC E5 31 5E 8D A3 D6 2C 63 9C D7 35 74 B5 DD 43 8A D3 3F 8C DB 4D A0 DA 52 AB 27 84 E3 65 C8 4E 9A 24 8F 1D 8C 1E 91 27 9E 1D 98 36 B5 57 DA 80 28 B1 42 CF 7F 31 C4 7A 32 CB 87 2A C9 8B 4F 15 BC 86 52 20 B4 86 5A 30 E7 C1 9D 7B 5B 22 06 CB B3 9D 89 77 67 59 32 28 20 1A 16 14 14 16 1A 05 0D 17 23 31 41 53 67 7D 7A 94 B0 CE 0F 31 55 7B A3 B2 DE 2D 5D 8F C3 1A 52 8C AD 0C 4C 8E
Cipher Text (Original)	**00** 4D D0 24 30 2C 10 8B 7F 39 16 1E 58 C8 C6 BC AA D8 05 F4 88 B4 82 B8 58 74 48 B0 CC 14 D3 89 FA 48 90 A1 7E ED 9D CF FF F3 15 B9 C2 D7 95 2F 99 AB EC CC FA 6B 46 69 BE 12 94 18 C5 E8 74 EC 77 4F 32 A6 D8 E9 E6 4B 6C A0 DA E5 33 47 D9 9E 71 EE 7A 7D D8 E3 D1 E4 EB 41 59 25 32 6C 2F BF 84 6C 35 45 34 FF 8D 93 8B D7 DF B6 F3 3E 1F 5E 83 99 AA 5C 02 7B 0E 72 B6 B8 CA 55 7A E0 C1 80 4A 55 32 10 8B 8E 5C FF A5 41 01 C8 4C 8B 48 76 F7 13 13 33 57 CF BF A3 09 96 37 FF 43 59 1E 98 52 62 4D F0 DA 6D 3F 43 9D 1A A4 8B 70 48 C9 4B 75 51 F3 13 68 C3 D0 D0 CE 15 D5 DD FE DB 7B 25 16 A6 9A 16 5A 45 E1 A6 FB A0 21 8A 8E 54 50 32 5E 08 7A FE 32 45 45 25 B2 7E AE 91 55 03 83 6D 57 D3 5E 3A FE 18 B9 49 A6 B9 19 12 E6 5E BE 6C 6C 1A 28 F7 6E 96 B6 7A 99 B7 C7 E7 4B 92 72 60
Modified Cipher Text	**01** 4D D0 24 30 2C 10 8B 7F 39 16 1E 58 C8 C6 BC AA D8 05 F4 88 B4 82 B8 58 74 48 B0 CC 14 D3 89 FA 48 90 A1 7E ED 9D CF FF F3 15 B9 C2 D7 95 2F 99 AB EC CC FA 6B 46 69 BE 12 94 18 C5 E8 74 EC 77 4F 32 A6 D8 E9 E6 4B 6C A0 DA E5 33 47 D9 9E 71 EE 7A 7D D8 E3 D1 E4 EB 41 59 25 32 6C 2F BF 84 6C 35 45 34 FF 8D 93 8B D7 DF B6 F3 3E 1F 5E 83 99 AA 5C 02 7B 0E 72 B6 B8 CA 55 7A E0 C1 80 4A 55 32 10 8B 8E 5C FF A5 41 01 C8 4C 8B 48 76 F7 13 13 33 57 CF BF A3 09 96 37 FF 43 59 1E 98 52 62 4D F0 DA 6D 3F 43 9D 1A A4 8B 70 48 C9 4B 75 51 F3 13 68 C3 D0 D0 CE 15 D5 DD FE DB 7B 25 16 A6 9A 16 5A 45 E1 A6 FB A0 21 8A 8E 54 50 32 5E 08 7A FE 32 45 45 25 B2 7E AE 91 55 03 83 6D 57 D3 5E 3A FE 18 B9 49 A6 B9 19 12 E6 5E BE 6C 6C 1A 28 F7 6E 96 B6 7A 99 B7 C7 E7 4B 92 72 60
Resulting decryption	01 05 0B 13 1D 29 37 47 58 52 68 80 9A B6 D4 15 A6 CB D6 3F 09 75 A3 F3 34 29 C0 DC D6 12 B2 F2 05 48 8F 36 07 B2 E1 51 95 68 3E B6 10 B2 D1 71 E4 49 EE 77 A0 2C A1 33 95 08 A1 18 B1 2D CD 32 F7 3A 61 CB 15 C0 4F 00 A9 5C 33 8D 67 02 C2 62 14 BF C6 2F 38 C7 95 45 4A BE 5A 30 08 4D 6D 8A 74 29 40 BC F9 95 A5 A4 DA AF 96 9E 2A 0C 0E 19 49 46 55 5F 70 4B 7B 7A A1 74 5D A5 56 95 10 67 1A 7F 5E 02 86 E1 B3 CD DA 21 6E 15 88 6B 6A BD B6 A8 68 35 89 33 E5 23 C4 BF 7C 04 5E FD DA 00 04 02 DD 63 D8

	40 50 D2 33 84 37 31 22 8D 4E F0 77 AD 4C D5 36 B3 82 9C 3D C2 E2 92 08 86 25 7B 02 58 78 42 86 FD 68 CA 8B 60 DF 60 EA 2E 9E FB 57 9A CB 6D 69 BE AF 81 D9 C3 43 60 2C 2C BC 74 A3 BD 5C CE FA 83 F2 60 14 0F A0 57 23 7C 2D C9 CC 63 52 CB 27 3F 6E 3C 10 48 88 6F 9E BE EE F6
Block size	128 bits
Input Size	2048 bits = 256 bytes = 16 blocks

Although the impact is not as great as modifying the key, modifying small amount of cipher text still cause lots of errors, and of course, it will broke the original messages and let users be warned about possibility of attack.

D. CFB Mode

SATE can be operated on CFB mode so that it can encrypt smaller data unit than its block size. The smallest size that SATE allows is 8 bits or 1 byte. This experiment's goal is proving that CFB Mode operation do not cause SATE to lose its strong characteristics.

First, plain text was encrypted per each byte. Then, the writer made a small change to the key (1 bit). The result is as following:

Key 1	0D 0F 0F 13 11 13 13 17 15 17 17 1B 19 1B 1B **1F**
Plain Text	01 05 0B 13 1D 29 37 47 59 52 68 80 9A B6 D4 15 37 5B 66 8E B8 E4 33 63 95 C9 20 3D 77 B3 12 52 94 D8 3F 87 B6 23 71 C1 34 88 DE 57 B1 13 71 D1 54 B8 3F A7 32 9E 12 82 15 89 20 98 33 AF 4E B3 56 DA 81 2A B4 61 EF A0 38 CC 83 3C D6 93 52 F2 B5 5F 26 CE 99 66 35 E5 B8 8D 49 22 DC B9 98 79 5C 41 28 D5 C0 AD 9C 8D 80 75 6C 65 45 42 41 42 45 4A 51 5A 65 57 66 77 8A 9F B6 CF EA 28 2C 4D 70 95 BC E5 31 5E 8D A3 D6 2C 63 9C D7 35 74 B5 DD 43 8A D3 3F 8C DB 4D A0 DA 52 AB 27 84 E3 65 C8 4E 9A 24 8F 1D 8C 1E 91 27 9E 1D 98 36 B5 57 DA 80 28 B1 42 CF 7F 31 C4 7A 32 CB 87 2A C9 8B 4F 15 BC 86 52 20 B4 86 5A 30 E7 C1 9D 7B 5B 22 06 CB B3 9D 89 77 67 59 32 28 20 1A 16 14 14 16 1A 05 0D 17 23 31 41 53 67 7D 7A 94 B0 CE 0F 31 55 7B A3 B2 DE 2D 5D 8F C3 1A 52 8C AD 0C 4C 8E
Cipher Text	58 A5 C6 C5 4B D1 94 01 6B 7E 8C 97 17 AC C7 1C 16 93 9F F9 B8 36 C4 BD F7 AA 5B FA 94 BE B0 20 6D 46 36 0F E4 95 5F 6A 7A 50 A7 2B 00 E1 E3 86 B1 7E 2D 56 64 70 D9 43 2A B8 57 F2 A3 1F 5F 63 91 63 4C 77 96 9F 57 AF D3 95 3E 2D EE 8C 86 66 36 88 48 B3 AC 96 DE 58 77 27 19 9C DE 86 83 25 A7 15 94 26 10 8E 41 D3 AB 45 05 CA D6 C0 5B 30 3C D7 68 8A 4A 05 59 8B EA EA 3C 08 F4 AB BF BE 7D 7B 5A 76 BF A6 10 7D 34 4F 5E D5 05 76 DC B8 39 63 A3 E8 1C 2A 10 16 8B 44 52 05 F5 F3 8C 83 05 D9 EE FD 06 0E A1 95 C0 BA A3 13 29 85 D0 CD 6C EA 99 4E F2 B8 E3 DB 02 BD 44 07 9D 2A 08 FE FF 55 E3 88 F8 75 59 2C 23 D4 50 C7 FD D6 78 8A C8 76 4C CA DB 25 E0 15 1E 4F 81 E3 1F 75 EA 65 1B D1 4F 7D B9 69 E3 63 1C 31 71 44 31 A0 1E 06 F3 F1 47 89 A7 11 F9 76 99 0A 71 6D 64 A3 B0 0E
Block Size	128 bits block size (queue) 8 bits processed at a time
Input Size	2048 bits = 256 bytes = 16 blocks

Key 1	0D 0F 0F 13 11 13 13 17 15 17 17 1B 19 1B 1B **1E**
Plain Text	01 05 0B 13 1D 29 37 47 59 52 68 80 9A B6 D4 15 37 5B 66 8E B8 E4 33 63 95 C9 20 3D 77 B3 12 52 94 D8 3F 87 B6 23 71 C1 34 88 DE 57 B1 13 71 D1 54 B8 3F A7 32 9E 12 82 15 89 20 98 33 AF 4E B3 56 DA 81 2A B4 61 EF A0 38 CC 83 3C D6 93 52 F2 B5 5F 26 CE 99 66 35 E5 B8 8D 49 22 DC B9 98 79 5C 41 28 D5 C0 AD 9C 8D 80 75 6C 65 45 42 41 42 45 4A 51 5A 65 57 66 77 8A 9F B6 CF EA 28 2C 4D 70 95 BC E5 31 5E 8D A3 D6 2C 63 9C D7 35 74 B5 DD 43 8A D3 3F 8C DB 4D A0 DA 52 AB 27 84 E3 65 C8 4E 9A 24 8F 1D 8C 1E 91 27 9E 1D 98 36 B5 57 DA 80 28 B1 42 CF 7F 31 C4 7A 32 CB 87 2A C9 8B 4F 15 BC 86 52 20 B4 86 5A 30 E7 C1 9D 7B 5B 22 06 CB B3 9D 89 77 67 59 32 28 20 1A 16 14 14 16 1A 05 0D 17 23 31 41 53 67 7D 7A 94 B0 CE 0F 31 55 7B A3 B2 DE 2D 5D 8F C3 1A 52 8C AD 0C 4C 8E
Cipher Text	18 5C 9C 54 CE 4A 7A 70 3A C7 4E A9 9C 15 30 FA 12 D9 51 E2 8F 08 E1 55 DF 6F F3 39 3C DF 1F 75 7B AD 4A 2D F8 C5 8E F6 96 05 C4 E2 5B 16 0B F2 F2 17 07 DC BF 8D 7B 05 1C 9C 0A 5F 77 E1 85 C7 DC 61 85 57 26 C3 BC AB 7B 85 D3 9D 69 E8 CD EE 9B E5 0E 9A 03 D1 52 91 29 44 13 C0 B8 6C 43 8A 04 47 A0 F4 48 A0 A5 FE 56 8D 0A 0C 52 DE F8 6C AF 88 6A 45 90 2D EE 73 27 2E 23 FF 68 E3 75 08 80 49 88 1A 87 23 D3 CB F3 57 20 E8 3E 9C 9E FD 6F D9 C3 F7 DB 86 AA 96 2E 71 97 53 14 D1 2E 9A B8 A2 D2 BB D6 FF 86 CF D4 58 24 AD E5 ED CA 7E 3F 47 FB A1 8F 09 21 71 D0 93 E9 86 5E 14 63 1A B1 E4 D9 C4 29 F9 FF B6 CD FA 43 C3 12 D4 E5 4A 97 97 4A 94 DD 7A 17 33 76 D0 2E 29 74 0B E5 27 5D 69 FE E5 88 E0 B7 73 4E 1F 4B 61 F0 C9 01 B6 00 32 82 B4 76 5A 3A 20 C8 4B B5 A6 CE 80 20 36
Block Size	128 bits block size (queue) 8 bits processed at a time
Input Size	2048 bits = 256 bytes = 16 blocks

The result shows us that SATE still has its strong diffusion when it is operated in CFB mode. The next experiment also proves that even smallest change in cipher text leads to huge difference of the resulting deciphered text. The experiment is as following:

Key	0D 0F 0F 13 11 13 13 17 15 17 17 1B 19 1B 1B 1F
Plain Text (Original)	01 05 0B 13 1D 29 37 47 59 52 68 80 9A B6 D4 15 37 5B 66 8E B8 E4 33 63 95 C9 20 3D 77 B3 12 52 94 D8 3F 87 B6 23 71 C1 34 88 DE 57 B1 13 71 D1 54 B8 3F A7 32 9E 12 82 15 89 20 98 33 AF 4E B3 56 DA 81 2A B4 61 EF A0 38 CC 83 3C D6 93 52 F2 B5 5F 26 CE 99 66 35 E5 B8 8D 49 22 DC B9 98 79 5C 41 28 D5 C0 AD 9C 8D 80 75 6C 65 45 42 41 42 45 4A 51 5A 65 57 66 77 8A 9F B6 CF EA 28 2C 4D 70 95 BC E5 31 5E 8D A3 D6 2C 63 9C D7 35 74 B5 DD 43 8A D3 3F 8C DB 4D A0 DA 52 AB 27 84 E3 65 C8 4E 9A 24 8F 1D 8C 1E 91 27 9E 1D 98 36 B5 57 DA 80 28 B1 42 CF 7F 31 C4 7A 32 CB 87 2A C9 8B 4F 15 BC 86 52 20 B4 86 5A 30 E7 C1 9D 7B 5B 22 06 CB B3 9D 89 77 67 59 32 28 20 1A 16 14 14 16 1A 05 0D 17 23 31 41 53 67 7D 7A 94 B0 CE 0F 31 55 7B A3 B2 DE 2D 5D 8F C3 1A 52 8C AD 0C 4C 8E
Cipher Text (Original)	**58** A5 C6 C5 4B D1 94 01 6B 7E 8C 97 17 AC C7 1C 16 93 9F F9 B8 36 C4 BD F7 AA 5B FA 94 BE B0 20 6D 46 36 0F E4 95 5F 6A 7A 50 A7 2B 00 E1 E3 86 B1 7E 2D 56 64 70 D9 43 2A B8 57 F2 A3 1F 5F 63 91 63 4C 77 96 9F 57 AF D3 95 3E 2D EE 8C 86 66 36 88 48 B3 AC 96 DE 58 77 27 19 9C DE 86 83 25 A7 15 94 26 10 8E 41 D3 AB 45 05 CA D6 C0 5B 30 3C D7 68 8A 4A 05 59 8B EA EA 3C 08 F4 AB BF BE 7D 7B 5A 76 BF A6 10

	7D 34 4F 5E D5 05 76 DC B8 39 63 A3 E8 1C 2A 10 16 8B 44 52 05 F5 F3 8C 83 05 D9 EE FD 06 0E A1 95 C0 BA A3 13 29 85 D0 CD 6C EA 99 4E F2 B8 E3 DB 02 BD 44 07 9D 2A 08 FE FF 55 E3 88 F8 75 59 2C 23 D4 50 C7 FD D6 78 8A C8 76 4C CA DB 25 E0 15 1E 4F 81 E3 1F 75 EA 65 1B D1 4F 7D B9 69 E3 63 1C 31 71 44 31 A0 1E 06 F3 F1 47 89 A7 11 F9 76 99 0A 71 6D 64 A3 B0 0E
Modified Cipher Text	**57** A5 C6 C5 4B D1 94 01 6B 7E 8C 97 17 AC C7 1C 16 93 9F F9 B8 36 C4 BD F7 AA 5B FA 94 BE B0 20 6D 46 36 0F E4 95 5F 6A 7A 50 A7 2B 00 E1 E3 86 B1 7E 2D 56 64 70 D9 43 2A B8 57 F2 A3 1F 5F 63 91 63 4C 77 96 9F 57 AF D3 95 3E 2D EE 8C 86 66 36 88 48 B3 AC 96 DE 58 77 27 19 9C DE 86 83 25 A7 15 94 26 10 8E 41 D3 AB 45 05 CA D6 C0 5B 30 3C D7 68 8A 4A 05 59 8B EA EA 3C 08 F4 AB BF BE 7D 7B 5A 76 BF A6 10 7D 34 4F 5E D5 05 76 DC B8 39 63 A3 E8 1C 2A 10 16 8B 44 52 05 F5 F3 8C 83 05 D9 EE FD 06 0E A1 95 C0 BA A3 13 29 85 D0 CD 6C EA 99 4E F2 B8 E3 DB 02 BD 44 07 9D 2A 08 FE FF 55 E3 88 F8 75 59 2C 23 D4 50 C7 FD D6 78 8A C8 76 4C CA DB 25 E0 15 1E 4F 81 E3 1F 75 EA 65 1B D1 4F 7D B9 69 E3 63 1C 31 71 44 31 A0 1E 06 F3 F1 47 89 A7 11 F9 76 99 0A 71 6D 64 A3 B0 0E
Resulting decryption	0E 0A 0B 13 1D 29 37 47 59 5D 18 10 0B EE 8C 7C 26 DB 4D 25 31 BF 6C 6C 80 DC ED A9 0F 8E 17 5D F8 AC A0 12 4E BA 69 9A 20 7E 8F 8C 09 7A F0 49 2E CE 75 A4 07 02 48 B8 30 F8 CA 86 48 06 C6 71 20 99 94 4A 17 A2 F6 D6 9E 3A 3E A5 0F C4 23 31 72 AE 1A DA 0B FB 8B C6 81 F8 3D 3D 81 45 3F C4 7A 62 C0 D9 49 A7 86 DF 16 B9 A8 46 A2 B6 91 A5 79 B9 2B 33 DD 40 84 A2 7E 52 65 3A B1 AB 6C 22 58 3F 68 A5 24 30 90 DD E6 61 88 73 C0 28 EF 13 F2 E5 34 2F 1C 05 FD 73 A9 74 36 37 D5 D3 90 D5 58 85 36 0D 70 6C D9 78 05 EE 13 9E C5 6B 9E 67 4F 6D E6 17 28 E0 57 FE 93 5B F7 37 00 30 69 EF 61 13 71 55 AE 99 2F 53 9F 4E 90 05 BB B0 78 10 FD C7 A1 97 90 A5 BF 35 BA A7 82 18 43 51 D3 A7 E0 EA EC 4F FE 1D 2E C3 6D C3 E7 FC 4C 2D EF 86 FE 4E C5 F7 F2 4F 02 F6 3F F7 2D FA 11 51 B1 A4
Block size	128 bits
Input Size	2048 bits = 256 bytes = 16 blocks

The result shows us that the smallest change occurred in cipher text lead to totally different decryption result. Thus, this allows users to detect even the smallest change in their data.

V. SECURITY ANALYSIS

The securities of cryptographic algorithms are the most difficult characteristic to compare. As there is no theoretical way to measure them, the only things that we can rely are estimates and guesses. Cryptographic algorithms are considered strong if no one able to break it.

Hence, the writer's security analysis is based from estimated measurement and guesses. The writer uses two main properties to measure SATE's security. They are Shannon's confusion and diffusion properties.

A. *Confusion*

The main principle of confusion property is preventing any connections between cipher text, plain text, and key to make statistical analysis very frustrating. SATE implements this property by:

1. Using entirely internal keys. The original key is only used to generate the internal keys. Therefore, this cuts the direct connection between the key and cipher text.
2. Using an S-Box substitution. This cuts the direct connection between plain text and cipher text.
3. Using tree structure. This contributes a large random factor to the cipher text.

By implementing those, SATE has a very strong confusion property. This cause statistical cryptanalysis to SATE is extremely hard.

B. *Diffusion*

The main principle of diffusion property is making even a small change become huge. SATE implements this property by:

1. Using iterative encryption. Thus, this makes small changes iterated into huge changes.
2. Using tree structure. This makes different path selection in the tree causes a very different cipher text.
3. When operated in CBC or CFB mode, each D and E function is linked, so small changes could lead to major changes.

The implementation is proven in the previous chapter, which is experiment. SATE shows incredible diffusion performances over the smallest change possible (1 bit).

VI. CONCLUTIONS

Simplicity in cryptography is one of the most important factors that is considered when we assessing cryptographic algorithm. Making it simple means cheaper cost. This also allows algorithms' applications to be wide. NIST's choice of Rijndael algorithm as the AES proves that simplicity is a very important criteria every algorithms must have.

SATE, with its Fistel-Tree structure, is a very simple algorithm. It uses cheap computation method which can be implemented in hardware forms easily. Experiments also prove that SATE algorithm is a very strong encryption algorithm. It implements confusion-diffusion properties in a very eloquent way.

In the future, deeper and more comprehensive research can be conducted to make this algorithm even simpler and faster than before. The writer encourages the cryptographic community to use this work as a baseline for other works. The writer hopes that this work can contribute as much as possible to cryptographic community.

ACKNOWLEDGMENT

Andre Susanto, as the writer of this paper, want to express his deepest gratitude to Dr. Ir. Rinaldi Munir, M.T. as the lecturers of IF 4020– “Kriptografi”. Special thanks to all of my family, my friends in Informatics 2012, and other people that give any form of support to me to finish this paper.

REFERENCES

[1] J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, E. Roback, “Report on the Development of the Advanced Encryption Standard (AES),” National Institute of Standards and Technology, U.S. Department of Commerce, October, 2000.

[2] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson, T. Kohno, M. Stay, “The Twofish Team’s Final Comments on AES Selection”, May, 2000.

[3] R. Munir. Security Analysis of Selective Image Encryption Algorithm Based on Chaos and CBC-like Mode. 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA), January 2012.

[4] S. Trenholme, S-box Used by the AES Cryptographic Algorithm. http://www.samiam.org/s-box.html, Accessed on 10 March 2016, 02.30 pm.