SCAC-MAT

Stream Cipher Algorithm with Ikeda Map Trajectories

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

***Abstract*—Stream ciphers are used widely to encrypt continuous data streams such as WiFi or cellphone signals. In order to "mask" data, they need special bit stream generators that generate random sequences. The resulting sequences would determine the algorithms' security, if they were random and hard to be predicted, then the algorithm would be secure, on the contrary, if they contained repeated pattern or were easily predicted, then the algorithm would not be secure. Chaotic Maps functions are one way of achieving good random numbers as their chaotic behavior are extremely sensitive to initial parameter modifications, making it impossible to predict future sequence without knowing the correct initial parameters. In this paper, we presented our stream cipher algorithm, which was called "SCAC-MAT", which used Ikeda Map trajectories to generate random number sequences. Experiments showed us characteristics of strong cipher that SCAC-MAT had, although further studies might be needed to study one potential problem.**

Keywords—Stream Cipher; Ikeda Map; Chaos; Chaotic Random Number Generator; Chaotic Stream Cipher.

I. Introduction

Since Caesar's era, cryptography has been used to protect valuable information from enemy. It was also used in the World Wars by all parties involved to make sure that precious information did not reach enemies' ears. Nowadays, modern cryptography is used in wide area of IT security [1].

In modern cryptography, Stream Cipher is used to encrypt continuous data stream such as WiFi, cellphone communication, handy talkie communication, or any kind of continuous communication that require transmitter to send data as streams. RC4, A5, Chameleon are examples of widely used Stream Ciphers [2].

Stream ciphers require bit stream generators to "mask" the plain text. In fact, they are the key factor to every stream cipher algorithms' security. Hence, to improve security, stream ciphers need to use better bit stream generators. The main job of bit stream generators is generating random sequence. Therefore, the more random they generate, the better we get secured.

Chaotic Map functions are one of our golden tickets to get good random sequences. They have a very valuable property that can be used to enhance stream ciphers' security. It is their sensitivity to their initial values [3]. That means even the slightest change would cause very big impacts. By having this property, Chaotic Map trajectories are very hard to be predicted. Nowadays, there are so many Chaotic Maps available. They have two kinds of time dimension, which are continuous and discrete, three space domains, which are real, relational, and complex, and ranging space dimensions (from 1 to 4).

In this paper, the writer explained a new stream cipher algorithm that exploits the valuable property of Chaotic Maps. The cipher is called "SCAC-MAT', which stands for Stream Cipher Algorithm with Ikeda Map Trajectories. As being mentioned in the name, the cipher uses Ikeda Map as its number generator.

II. Theories

A. Stream Cipher

Stream cipher is one of modern cryptography methods. It is one of symmetric key ciphers and commonly used in telecommunication applications, where encrypting data in unit of blocks is not feasible. Hence, stream ciphers encrypt data by masking them bit by bit or byte by bytes rather than block of bytes.

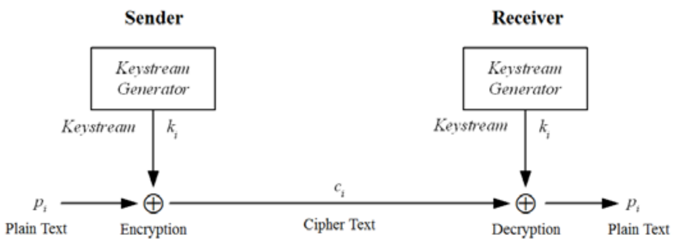

Figure 1 - How stream ciphers work [4]

Figure 1 shows us how stream ciphers work. As shown in the picture, stream ciphers used key stream generator in both sender (where the encryption process taking place) and receiver (where the decryption process taking place). As stream ciphers rely only to key stream generator to mask plain texts, the security of every stream ciphers is determined by the security of key stream generator that it uses.

If a key stream generator generated truly random sequences, then the security of the cipher would be comparable with One Time Pad (OTP). Hence, it would be unbreakable. On the other hand, if a key stream generator generated recurring sequences (especially short and predictable one), then the security of cipher that uses it would be comparable with simple XOR ciphers. Accordingly, it would be easy to break. Typically, stream ciphers are designed by cryptographers in between simple XOR and OTP [4].

B. Chaos Theory

Chaos theory describes behaviors and conditions of dynamical systems that are extremely sensitive to initial parameters. Edward Lorenz described chaos as “When the present determines the future, but the approximate present does not approximately determine the future” [5]. As Lorenz described, long-term prediction is generally impossible.

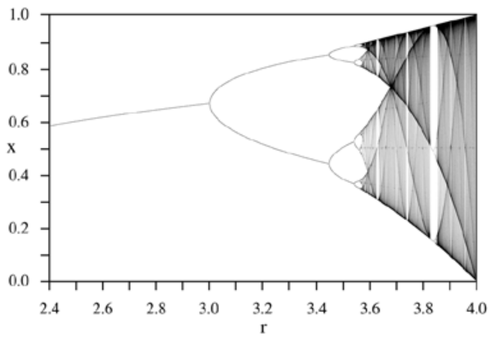

Figure 2 - Logistic Map Bifurcation Diagram

Logistic Map is one of chaotic maps that models discrete-time population size. Mathematically, the map is written:

$$x_{n+1} = rx_n(1 - x_n)$$

where:

x_n is a number (0…1) that represent ratio of existing population to maximum possible population.

r is a number (0…4] that represent values of interest.

Figure 2 shows us period-doubling as r increases in Logistic Map that will eventually produce chaos. From the diagram, chaotic behaviors are seen for $r > 3.57$.

As it is very sensitive to initial parameters and normally impossible to predict future by present data, chaotic systems are excellent to be used as pseudo-random number generator (PRNG) in cryptography world. Stream cipher is one of cryptography systems that can utilize this chaotic PRNG.

C. Ikeda Map

Ikeda Map is one of Chaotic Maps that models light movement around nonlinear optical resonators [5]. The original map is usually modified so that saturation effect of nonlinear dielectric medium is taken into account [6]. The modified form is mathematically written:

$$z_{n+1} = A + Bz_n e^{iK/(|z_n|^2+1)+C}$$

where:

z_n is the electric field inside the resonator

A and C are applied laser light from outside

B is the loss of resonator

The above form can be written in 2D as:

$$x_{n+1} = 1 + u(x_n \cos t_n - y_n \sin t_n)$$

$$y_{n+1} = u(x_n \sin t_n - y_n \cos t_n)$$

where u is a parameter, and:

$$t_n = 0.4 - \frac{6}{1 + x_n^2 + y_n^2}$$

The system has a chaotic attractor for $u \geq 6$.

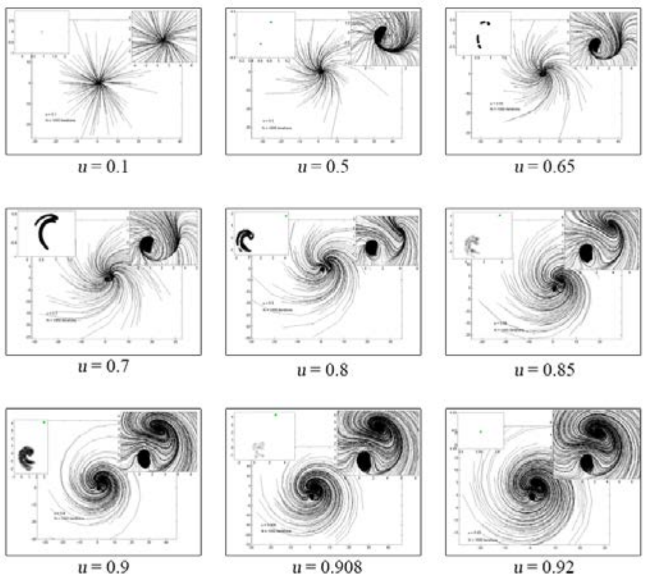

Figure 3 - Point Trajectories on Different *u*

Figure 3 shows us different point trajectories on different u parameters. Smaller diagrams on the left upper side of each diagram are the attractors while the one in the right upper side is the zoomed view of trajectories’ centers. We can’t see chaotic attractor for $u = 0.1$ and 0.5, but it can be seen clearly for $u \geq 0.65$. Consequently, in order to get chaotic behavior, we need to set u value in between 0.6 to 1.0.

III. THE PROPOSED ALGORITHM

A. Number Compression Algorithm

SCAC-MAT operates in unit of byte, which is 8-bits, while the Ikeda Map operates in *real* numbers, which are implemented in single floating point (32-bits), and double floating point (64-bits). Therefore, a number conversion (or simply compression because we actually compress larger bits to smaller ones) method is required by the algorithm in order to convert those real numbers to byte values.

The algorithm that is used by SCAC-MAT to convert *real* numbers to bytes is as following:

1. Divide the bits so that there are *n* pieces of 8-bits block. As SCAC-MAT implementation uses single floating point (32-bits), there would be 4 blocks of 8-bits.
2. XOR first block and second block. Circular shift the resulting bits to the right twice.
3. XOR the resulting XOR from previous step to next blocks and circular shift the bits likewise in second step.
4. Repeat the process until all blocks of bits involved in XOR process. The result of this algorithm is 8-bits (or 1 byte) number representation of the input number.

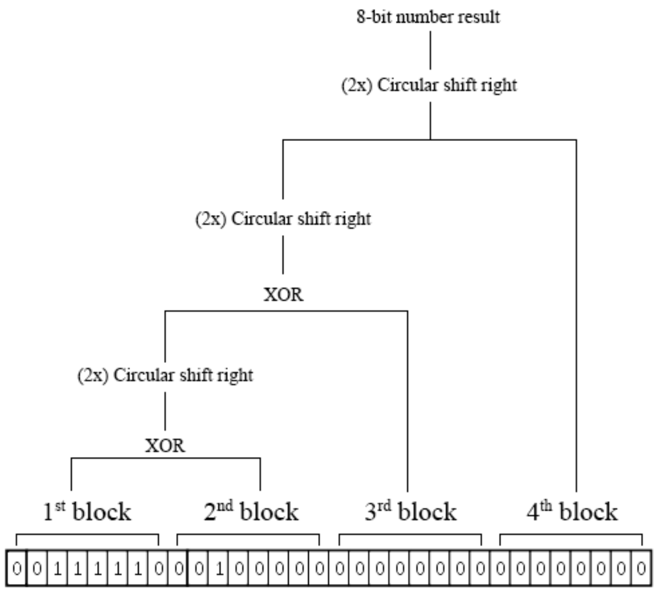

Figure 4 - SCAC-MAT's Number Compression Algorithm

Figure 4 shows us steps that SCAC-MAT's number compression algorithm has to take in order to compress a 32-bit single floating point number. At the bottom of the picture, there was a bits representation of IEEE 754 single-precision binary floating-point format. The bits were divided into 4 blocks and labeled 1st block to 4th block.

At first, the first block was XORed with second block. Then, the resulting block was circular shifted to right twice. After that, the resulting block was XORed with the third block. Once again, the resulting block was circular shifted to right twice. After that, the resulting block was XORed for the last time with the forth block. The resulting block then was circular shifted to right twice. The process was finished and an 8-bit number that represents 32-bit number was generated.

B. *Key and Initial Parameters*

To generate chaotic sequences, Ikeda Map needs initial parameters as in order to get either x_n or y_n, we need both x_{n-1} and y_{n-1}. A slight change on these two initial parameters would result a significant difference on the resulting sequences. Therefore, these two initial parameters are excellent key candidates for the cipher.

To make the cipher even more secure, the *u* parameter is also used as a part of this cipher's key. Hence, there are three real values that are used in forming key for this cipher: 1) x_0 value, 2) y_0 value, and 3) *u* parameter (0.6…1). As SCAC-MAT uses 32-bit floating point variable, the key size of this cipher is 96-bit.

C. *Encryption and Decryption*

Both encryption and decryption process require key stream generator to generate random sequences that are used to "mask" and "un-mask" data. SCAC-MAT uses Ikeda Map Pseudo Random Number Generator to generate those random sequences. As 2D Ikeda Map generates both x and y values, the resulting random numbers are the x and y values alternately.

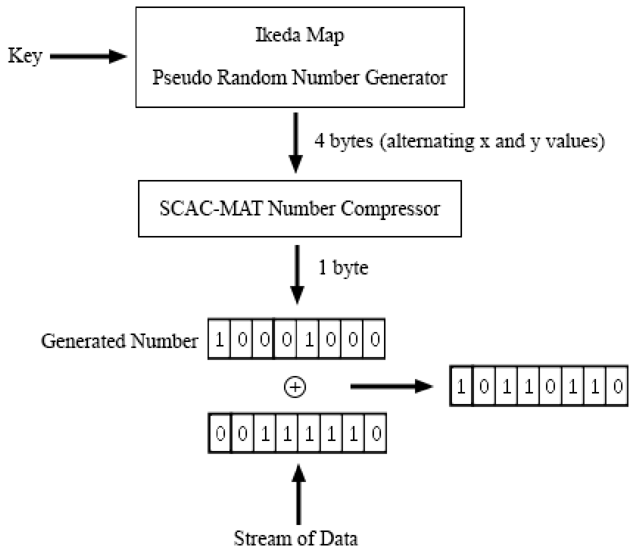

Figure 5 - SCAC-MAT Encryption/Decryption

Figure 5 shows us SCAC-MAT encryption and decryption process, which are similar. The only difference between encryption and decryption process is at the stream of data. For the encryption process, stream of data would be plain texts. On the other hand, the stream of data would be cipher texts. The detailed encryption/decryption algorithm is as following:

1. Key was provided to Ikeda Map PRNG.

2. Ikeda Map PRNG calculated x_{n+1} and y_{n+1}, then returned them alternately (x was retuned first, then y).
3. SCAC-MAT number compressor got the returned number from Ikeda Map PRNG and compressed it into 1 byte number.
4. The compressed number was used to "mask" or "un-mask" data by XORing them to the data.
5. The process was repeated from number 2 for each arriving data stream.

IV. Experiments and Analysis

A. Random Number Trajectories

In order to enable us to view SCAC-MAT's Ikeda Map PRNG's output trajectories, we conducted experiments with various variations of parameters. One of our experiments was conducted with the following configuration:

1. u parameter = 0.98425f,
2. x_0 = 0.12332455536f,
3. y_0 = 0.678764532435f,
4. Iteration = 10,000,000 times.

The result of this experiment was 10 million pairs of x and y values. Later, we plotted those values in a scatter plot.

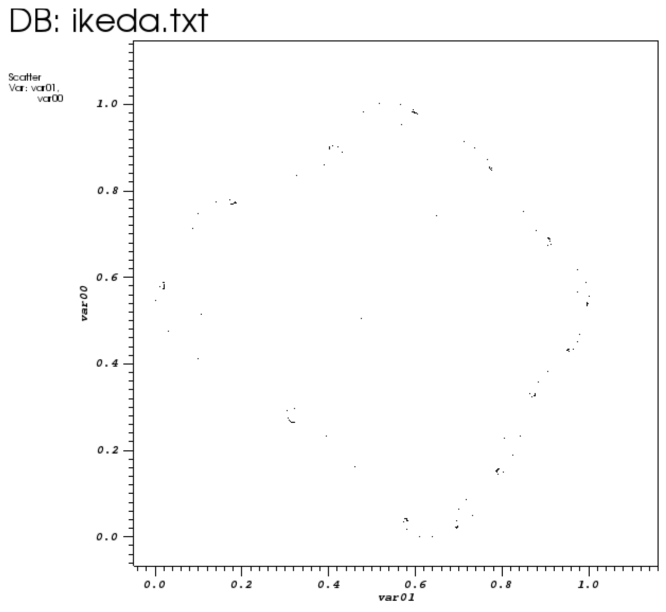

Figure 6 - Ikeda Map PRNG Trajectories

Figure 6 shows us the Ikeda Map PRNG trajectories that we plotted in our experiment. In the picture, the trajectories were forming a diamond shaped formation with only a little amount of deviation. We can also see that there were some bright spots in the picture where some points concentrated. Other experiments that we have conducted also didn't give good results. They would always form circular-ish shaped formation, with some points concentrated at specific areas.

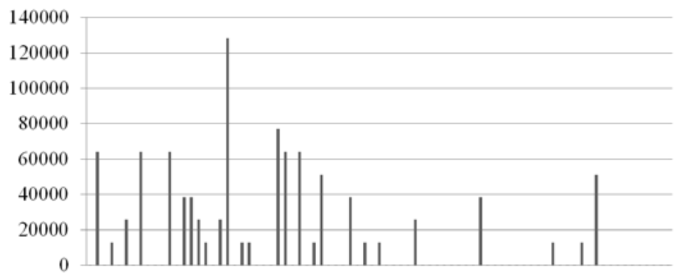

Figure 7 – Compressed Number Histogram

As there were points that concentrated at specific regions, obviously the number distribution was not spread evenly. Figure 7 shows us the distribution of one million of compressed Ikeda Map PRNG's numbers. Clearly, there was a number which was used over 120,000 times while the average usage of each number was only around 40,000 times. Furthermore, some numbers were used less than 100 times.

B. Number Sequence Repetitions

We conducted numbers of experiments to find out whether our Ikeda Map PRNG would result repeated number sequences or not. Our setups were as following:

1. Number of samples = 10 samples
2. Sample size = 100 million bytes
3. Minimum sequence size = 1000 bytes

We didn't found any repeated sequence in the generated number sequences. Therefore, our Ikeda Map PRNG is considered as a secure PRNG.

C. Initial Parameters Sensitivity

Chaotic systems are sensitive to initial parameters changes, so does our Ikeda Map PRNG. We conducted several experiments to test how sensitive our Ikeda Map PRNG was.

This was our reference for this test:

Initial Parameters: u = 0.98425 x_0 = 0.5644 y_0 = 0.7689

First 150 Generated Number Sequence:
`1E EE DC 1C 4E F7 F7 EC 0B 9F F6 F0 FD 88 9E EF`
`0F EF F7 F5 F2 E8 FC EE F5 EF F0 F2 5A F1 F2 09`
`1F EF 5E DE F3 98 F6 FF 89 9F 9E EF F7 D8 FF 1E`
`FF 5F F1 F4 0A EB F3 F1 8E F8 EE 4F 4E FF EB 9F`
`F6 EF 9F FD EF FF ED 1E DC F0 0E CF FF F5 EE ED`
`FD F4 F6 F5 CE 0F FE EF 1F F4 F6 1F EF 58 EE FF`
`FC F9 F2 1E 0E FF F5 FC 0E FC F3 09 DE 1E F5 8F`

```
FE EC 1B EE FF 1F EF EF 9F 8F FC F5 F6 FE F7 FE
EF F5 FE EB EF 8F F7 F1 1E EF 8E F0 F7 F5 F6 5F
EF DE DF EE 1C ED
```

Then we did a slight *u* parameter modification, following was the result:

Initial Parameters:

$u = 0.9842500001$

$x_0 = 0.5644$

$y_0 = 0.7689$

First 150 Generated Number Sequence:

```
5E F6 CC EC F4 EC FA F2 EC 8C F2 19 FD F0 8E 0F
1E F4 9F F4 ED EF FC EE D9 EF F8 FE 8C E9 F2 F4
FC 9E F8 FC FD F5 5E EF F0 DF 1F E8 4B EE F7 EE
EE 4F FF F5 F2 F2 F7 F1 F2 F5 DE 1C F0 FE F7 F0
8A EF CF DF FF FF F7 FD ED FF 0E DE 0E FF 1E F6
4E 4E F6 ED EE F8 DE F5 F7 0F EF 1F EF FE EE 4F
FC F1 F3 F1 FE FF DF FD 0E F3 F3 19 F6 18 F0 DF
4E 99 1D FE FB 1F 5F 88 8F 1F 1C F1 FF F5 9A FF
EF 0E CE FF F7 F0 1B F1 FE EF 8E E8 4C F1 F7 F1
F4 E9 F1 F0 F7 FE
```

The result shows us that a slight *u* parameter change generated a totally different number sequence from the original. We also saw a likewise behavior for a change that we did at x_0. The following was the result:

Initial Parameters:

$u = 0.98425$

$x_0 = 0.56400001$

$y_0 = 0.7689$

First 150 Generated Number Sequence:

```
1F F2 CC EC 1D FC 4B 9E 0B FC F2 FE FD 98 FF EF
1F F4 F6 F4 ED F1 FC EE F1 EE F8 FA EE ED DA 1E
E8 9F FF CE EC FE F2 DF EC 5E 1F E9 0B EF F3 F0
EF 4F F1 F0 EA FD 9B F5 8A 4D DE DF F1 FF F7 5E
8A FF EF CF 0F 4D F3 FF EC FF FE EF FF 89 8D 1B
5A D9 EA E9 DE 4E EE 8F 1F 09 FF 1F EF F0 DF 4F
9C F1 FD 5F 0E FF CF FD 0E DF FB 1D F6 F4 EF F1
CE 5C FF F0 EF 1F EF DF FF EF EB FD F6 CC 9C FF
EF E9 EE F7 FF 8F 0B F1 EE 4E 8E EF EF F5 CC DC
1E F4 EF F9 CA F5
```

The result shows us that a slight change in x_0 also generated a totally different number sequence. Lastly, we applied likewise small change to the y_0, the following was the result that we got:

Initial Parameters:

$u = 0.98425$

$x_0 = 0.5644$

$y_0 = 0.76900001$

First 150 Generated Number Sequence:

```
DE EA CC EC F0 F2 F7 8E F7 FD F2 1E 5D 98 EE 8C
FD 49 F2 F9 8A EE FC FE D9 59 F0 F2 F7 F1 9E 1E
EE 8F EE EC F7 8F FC EF F0 DF 5F F5 F3 F0 F7 F0
F5 4F F5 49 F6 5D F7 FD F2 4E DE 9F EE FF F7 5E
8E EF CF DF EF E9 F6 FE FC F0 0E F5 FF 4D CD 0B
DC 58 ED EF CE 58 EE 4E F7 09 EB 1F DF F4 DF 4D
FA ED CC 5F 0E FF CF F9 0E EC F7 19 DF F1 1E 9F
4E F0 F7 EE EB F1 DF EE 9F EF EC F1 4D F0 9F FF
EF C9 5F F7 F7 E9 EC 99 4E FF 8E F0 EB F1 F7 FF
EE ED EF EF 5C F9
```

From all 3 number sequences that we got from very trivial changes, no one of them produced the same sequence as the original one. They were hardly even close to it.

V. Security Analysis

A. Key Security

SCA-MAT's key size is 96-bits. However, there are some bits that are not usable; as a consequence, brute force complexity to break the key is less than 2^{96}. The reasons for that are:

1. Ikeda Map works in range of [0…1]. According to IEEE Standard 754 Single Precision Floating Point Number, there are only $2^{23} - 1$ subnormal numbers and $126 * 2^{23}$ normal numbers in that range. As Ikeda Map only operates normal numbers, total numbers in the range is $126 * 2^{23}$ or 1,056,964,608.

2. The *u* parameter that Ikeda Map uses need to be larger than 0.6 to produce chaotic behavior. Roughly, only 40% of the number mentioned in previous explanation is required to brute force this parameter, or 422,785,844.

So, rather than 2^{96}, an exhaustive search to find the key only requires $15876 * 2^{46} * (40\% * 126 * 2^{23}) \approx 2^{88.6}$ or 472,325,429,670,581,816,751,292,416. Although its key complexity is only about 1.04% compared to complexity of 2^{96} keys, it is still a lot better than DES's key complexity, which is 2^{56}.

B. Known Plain Text Attack

One big problem that stream ciphers face is their ability to withstand known plain text attacks. Known plain text attack is a type of attack to ciphers that uses both plain text and cipher text to estimate the corresponding key.

In simple Ikeda Map ciphers (without number compression algorithm), if we had a sequence that contained at least three plain texts and their corresponding cipher texts, we could obtain all three parameters that are required to form a key (u, x_n, and y_n). Although the key would not be same with the original one, it still could be used to calculate the rest of number sequence in the same way the original key would do. The detailed explanations about how to do this attack are:

1. The first plain and cipher text pair was used to obtain x_n. To get x_n, we could simply XOR the plain text and its corresponding cipher text.
2. The second plain and cipher text pair was used to obtain y_n. The process was done likewise the process we did in step 1.
3. As now we got x_n and y_n, we could calculate $(x_n \cos t_n - y_n \sin t_n)$. We could get u parameter by substituting the result we got from that calculation into this equation: $x_{n+1} = 1 + u(x_n \cos t_n - y_n \sin t_n)$. The XOR of third plain and cipher text pair would be the value of x_{n+1}.

This type of attack could be avoided by cutting direct connection between key and cipher text. SCAC-MAT's number compression algorithm also does this job perfectly besides compressing larger number format. For example, given an 8-bits number $(11010001)_2$, there are so many possible corresponding 32-bits numbers. Even expanding that number to only 16-bits number might result in $(0100011100000000)_2$, $(0100011000000001)_2$, $(0100010100000010)_2$, and many more possibilities. Hence, it is impossible to recover the original number from our compressed number.

C. *Potential Uneven Number Distribution Problem*

As being mentioned before in IV.A, the distribution of number that was generated by SCAC-MAT was not spread evenly. Some numbers were used frequently rather than the others. Statistically speaking, this was not a good thing and might allow statistical-based attack to SCAC-MAT.

Although currently we have not found any way to attack our cipher by exploiting this potential statistical problem, further study about this potential problem is suggested to avoid future attacks.

D. *Shannon's Confusion and Diffusion*

The main principle of Shannon's confusion property is preventing any connection between cipher text, plain text, and key to make statistical analysis very frustrating. As being explained before in V.B, our number compression algorithm cut the connection between keys and cipher texts so that it was impossible to derive a key, even by using known plain texts attack. Hence, SCAC-MAT has a powerful confusion property.

As for Shannon's diffusion property, its main principle is even for an imperceptible change would cause immense impacts. SCAC-MAT naturally possesses this property thanks to the chaotic behavior that it has. As being discussed before in IV.C, tiny changes to key parameters resulted dramatic changes, which are characteristics of strong diffusion property. However, due to the nature of stream ciphers, tiny changes in cipher text would not cause dramatic changes in corresponding plain text. Therefore, if the cipher was not used properly by a system (for example: no data integrity checking), flip bit attacks might be possible for the system.

VI. CONCLUTIONS

1. Ikeda Map PRNG didn't generate any repeated number sequence. Hence, it was classified as secure PRNG and could be used to make secure stream ciphers.
2. Due to the nature of chaotic behavior that SCAC-MAT possessed, a minuscule change in the key would cause prodigious effects in the resulting cipher or plain texts. Experiments that we had conducted showed us accordingly.
3. Despite the fact that SCAC-MAT's key size is 96-bits, the complexity to break the key by doing exhaustive search is less than 2^{96}. The reason lies with the nature of numbers that it operates. However, in order to break the key, humongous computations are still required. In fact, the complexity to break SCAC-MAT's key is way larger than DES, which are $2^{88.6}$ and 2^{56} respectively.
4. SCAC-MAT was able to withstand known plain text attacks due to the usage of number compression algorithm. The algorithm cut the connection between cipher texts and keys, thus making it impossible to guess the corresponding key from plain and cipher text pairs.
5. According to the result of experiments, the number distributions were not spread evenly. Hence, further study is suggested to explore potential problem that it might cause.

ACKNOWLEDGMENT

Andre Susanto, as the writer of this paper, want to express his deepest gratitude to Dr. Ir. Rinaldi Munir, M.T. as the lecturer 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] Luciano, Dennis; Gordon Prichett (January 1987). "Cryptology: From Caesar Ciphers to Public-Key Cryptosystems". The College Mathematics Journal 18 (1): 2–17.

[2] Matt J. B. Robshaw, Stream Ciphers Technical Report TR-701, version 2.0, RSA Laboratories, 1995.

[3] Munir, Rinaldi; Riyanto, Bambang; Sutikno, Sarwono. "Perancangan Algoritma Kriptografi Stream Cipher dengan Chaos", KNSI, 2006.

[4] Munir, Rinaldi. "Diktat Kuliah IF5054 Kriptografi", Departemen Teknik Informatika ITB, 2005.

[5] K.Ikeda, Multiple-valued Stationary State and its Instability of the Transmitted Light by a Ring Cavity System, Opt. Commun. 30 257-261, 1979.

[6] K. Ikeda, H. Daido and O. Akimoto, Optical Turbulence: Chaotic Behavior of Transmitted Light from a Ring Cavity, Phys. Rev. Lett. 45, 709–712, 1980.

[7] Danforth, Christopher M. (April 2013). "Chaos in an Atmosphere Hanging on a Wall". Mathematics of Planet Earth 2013. Retrieved 4 April 2013.