Towards Generating a New Strong key for AES Encryption Method Depending on 2D Henon Map

ALa'a Talib khudhair* and Abeer Tariq Maolood

Computer science Department – University of technology

*Csalaa03@gmail.com

Received: 2 July 2018 Accepted: 4 December 2018

Abstract

In recent years, chaos-based encryption approaches introduced numerous advantages over the classic approaches like the extensive levels of security, complexity and speed. In this research, a safer algorithm is designed to generate keys using chaos theory and a new primitive table and primitive initial key and less time processing. This resulting key was used in AES (Advanced Encryption Standard) algorithm. First, perform many operations based on digits produced from chaos equations to produce 2048 bits. Second, this primitive initial key takes results from chaos theory to produce 64 symbols (512 bits) that obtains suitable key for randomness and complexity and apply primitive table on the result come from initial key to produce another 512 bits and worked XOR operation between the result of initial key and primitive table to produce another 512 bits and merge the results of this steps to get key of 3584 bits. The results of the experimentations indicated that the suggested method for key generation has the advantage of large key space with a safety protection against the brute force attacks. Thus, the results showed a high level of security for encryption on the basis of strong secret key features.

Keywords: 2D Henon map, primitive table, primitive initial key, key generation, chaos theory.

DIYALA JOURNAL FOR PURE SCIENCES

Towards Generating a New Strong key for AES Encryption Method Depending on 2D Henon Map

ALa'a Talib khudhair and Abeer Tariq Maolood

في الاتجاه نحو إنشاء مفتاح قوي جديد اعتمادًا على خريطة Henon ثنائية الأبعاد

الاء طالب خضير و عبير طارق مولود

قسم علوم الحاسبات – الجامعة التكنولوجية

الخلاصة

في السنوات الأخيرة، قدمت أساليب التشفير القائمة على الفوضى العديد من المزايا على الطرق التقليدية مثل المستويات الواسعة من الامن والتعقيد والسرعة. في هذا البحث، تم تصميم خوارزمية أكثر أمان لتوليد المفاتيح باستخدام نظرية الفوضى والجدول البدائي الجديد والمفتاح الأولي البدائي وفي أقصر وقت. تم استخدام المفتاح الناتج في خوارزمية AES.
أولا، تنفيذ العديد من العمليات على الأرقام الناتجة من معادلات الفوضى لإنتاج 2048 بت. ثانياً: ادخال الmaster key وتوسيعه اعتمادا على نتائج من نظرية الفوضى لإنتاج 64 رمزاً (512 بت) وبذلك يتم الحصول على مفتاح مناسب للعشوائية والتعقيد ومن ثم تطبيق جدول بدائي على النتيجة الناتجة من توسع الmaster key لإنتاج 512 بت أخرى ومن ثم تشغيل عملية XOR بين النتيجة من توسيع الmaster key والجدول البدائي لإنتاج 512 بت آخرى ودمج نتائج هذه الخطوات للحصول على مفتاح طول 3584 بت. أشارت نتائج التجارب إلى أن الطريقة المقترحة للتوليد الرئيسي لها ميزة الفضاء الرئيسي الكبير مع سلامة المفتاح ضد هجمات القوة الغاشمة. وبالتالي أظهرت النتائج مستوى عالٍ من الأمان للتشفير على أساس ميزات مفاتيح سرية قوية.

كلمات مفتاحية : خريطة Henon ثنائية الابعاد، جدول سري، مفتاح ابتدائي سري، توليد مفتاح، نظري الفوضى.

Introduction

Generating random numbers are mainly used to generate secret keys or random sequences and it can be carried out by many different techniques such as chaotic maps [1]. In recent years many pseudo-random numbers or sequences generators are proposed based on chaotic maps [2]. Chaos is considered the science of surprises, of the non-linear and the unexpected. It teaches the users to expect the unpredictable [3,4]. As the majority of the conventional sciences deal with rather predictable phenomena, Chaos deals with non-linear things which are effectively not possible to be predicted or controlled [5,6]. Chaos theory is a field of mathematics that is focused on dynamical systems behavior which are very sensitive to initial conditions [7,8]. “Chaos” is an interdisciplinary theory which states that within the seemingly random chaotic

complicated systems, underlying patterns can be observed, constant feed-back loops, fractals, repetitions, self-similarity, self-organization, and dependence on programming at the initial point referred to as sensitive reliance on initial states [9].

Related work

This section discusses important previous works of 2D Henon Map Chaos Theory

Francois [10], There logistic maps are cumulated in the algorithmic to produce a novel pseudo-random bit generator and 32 arbitrary bits block is created at each iteration [11]. New Dynamic private Key Generation for an effective Image Encryption Approach, a new approach for the generation of dynamic nonlinear secret keys for a symmetrical block cipher utilizing exclusive-OR operation is suggested. The dynamical nonlinear secret key generating depends on a mix of logistical and piecewise chaos map approaches with a new automatic generation of initial values [11] design chaotic linear congruently generator called CLCG is a pseudo random number generator and shows that what chaotic features of the discrete logistic map are utilizable for creating pseudo-random numbers in cryptographic perspective [12]. Suggested New Key stream Generator Based on 3D Henon map and 3D Cat map, they proposed a new way to generate key stream based on a combination between 3D Henon map and 3D Cat map. The principle of the method consists of generating random numbers by using 3D Henon map and these numbers will transform to a binary sequence. These sequence positions are permuted and Xored using 3D Cat map.

Theoretical background

2D Henon map

The Henon map introduced by Michel Henon [13]. The Henon chaotic is the most known and commonly used dynamical systems. It's a discrete-time dynamical system that exhibits chaotic behavior. Henon map introduces uniform distribution of bits and is a discrete time transform takes initial condition (x_0, y_0) as a secret value in the 2D real plane and map it to next point by using the following equation [14].

$$\begin{cases} x_{n+1} = 1 - a * x_n^2 + y_n \\ y_{n+1} = b * x_n \end{cases} \quad (1)$$

Where: (a=1.4, b=0.3) represented the control parameters and (x_0, y_0) represented the initial secret keys.

AES introduction

The Advanced Encryption Standard (AES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST) in December 2001 [15].

AES Overview

AES algorithm is a substitution-permutation network *(SPN)* structure block cipher, which processes 128-bit length block with variable length keys (128, 192, or 256) bits. Figure 1 shows the AES input/output parameters [16].

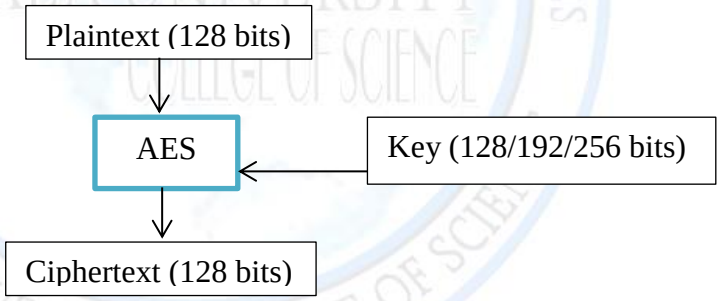


Figure 1: AES Input /Output Parameters [16]

AES is an iterated block cipher; the iterations are called rounds. The number of rounds depends on the block length and the key length. Three key lengths are supported by AES algorithm. The number of internal rounds of the cipher is a function of the key length according to table 1 [16].

Table 1: Key Lengths and Number of Rounds for AES [16]

Key lengths	Rounds
128 bit	10
192 bit	12
256 bit	14

A brief description of the AES three layers [16]

AES has only three different types of layers. Each round, with the exception of the first, consists of all three layers. Moreover, the last round does not make use of the Mix Column transformation, which makes the encryption and decryption scheme symmetric.

1. Key Addition layer (Add Round Key): A 128-bit round key, or subkey, which has been derived from the main key in the key expansion, is XORed to the state.

2. Byte Substitution layer (Sub Bytes): Each element of the state is nonlinearly transformed using a lookup table with special mathematical properties called substitution box (S-box). This introduces *confusion* to the data.

3. Diffusion layer: It provides *diffusion* overall state bits. It consists of two sublayers, both of which perform linear operations:

A. The **Shift Rows sublayer:** Permutes the data on a byte level.

B. The **Mix Column sublayer:** It is a matrix operation which combines (mixes) blocks of four bytes.

Proposed system design

The key is the important part of any security system because it determines whether the system is strength or weak. This paper designs a safer algorithm to generate key of length 3584 bits for AES algorithm and this key for 14 rounds using chaos theory, new primitive table, initial key (must be primitive between sender and receiver) as shown in figure 2, and many other equations based on the result of chaos in shortest time.

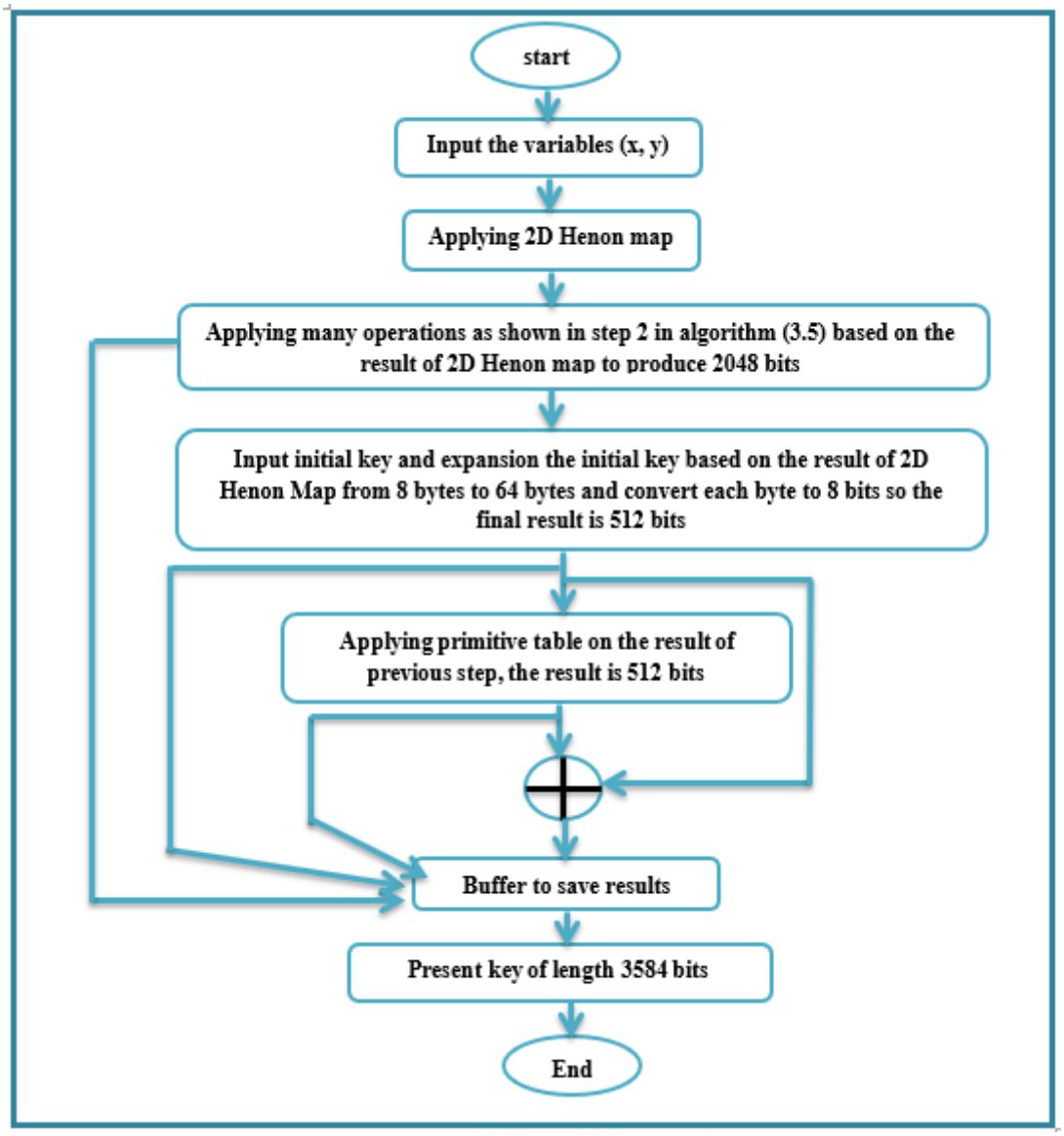


Figure 2: Flowchart Appears Steps of Generating a New Strong Key for AES Encryption Method Depending on 2D Henon map

Below the algorithm shown the steps of the key generation method

Algorithm 1: Generating Key for AES Encryption Method Depending on 2D Henon Map
Input: • a=1.4 • b=0.3

- x=0.6
- y=0.1
- initial key (8 bytes)

Output:

- key of 3584 bits (10 keys of (4*8) in HEX)

Process:

Begin

Step 1: Apply 2D Henon map

1.1 For i=1 to 53

$$\begin{cases} x_{n+1} = 1 - \mathrm{a} * x_n^2 + y_n \\ y_{n+1} = \mathrm{b} * x_n \end{cases}$$

1.2 Remove the negative signal

1.3 Cut 15 number after the comma

Next

Step 2: Do the following operation based on the result of step 1

Begin

2.1: Aggregation the numbers produced by x in string 1

2.2: Aggregation the numbers produced by y in string 2

2.3: Cuts string 1 to substrings each of length 3, each substring take module 256; convert the result to binary of length 8 and aggregation the results in string 3

2.4: Cuts string 2 to substrings each of length 3, each substring take module 256; convert the result to the binary of length 8 and aggregation the results in string 4

2.5: XOR operation between string 3 and string 4, the result is 2048 bits

End

Step 3: Input initial key (8 bytes) and expand the initial key by performing the following steps

3.1 Aggregation the numbers produced by x, take the numbers in odd locations and cut 112 number

3.2 Input initial key of 8 bytes and expand from 8 bytes to 64 bytes based on the numbers of step 3.1 by applying the following stages:

3.2.1- The first 8 bytes are the initial key

3.2.2- Cuts the first 16 numbers produced from x

3.2.3- Cuts 16 numbers to substrings each of 2 bytes, convert each substring to the binary of 8 bits.

3.2.4- Cuts initial key to substrings each of 1 byte, take the ASCI code of each byte , convert the ASCI code of each byte to the binary of 8 bits.

3.2.5- Working XOR operation between the binary of each substring produced from step 3.2.3 and the binary of each substring produced from 3.2.4 to produced new word of 8 byte

3.3- Now produce new word of length 8 bytes, this new word be the initial key instead of an old initial key and return performing the same operations with the next 16 numbers from Chaos and so on until the length be 64 bytes

3.4- Convert the result (64 bytes) to binary of length 512 bits by converting each byte to 8 bits

Step 4: applying a primitive table to the output of step 3**:**

{44, 10, 42, 34, 26, 18, 50, 2, 60, 52, 58, 36, 28, 20, 12, 4, 62, 54, 46, 38, 30, 22, 14, 6, 64, 56, 48, 40, 32, 24, 16, 8, 57, 49, 41, 33, 25, 17, 9, 1, 59, 51, 43, 35, 27, 19, 11, 3, 61, 53, 45, 37, 29, 21, 13, 5, 63, 55, 47, 39, 31, 23, 15, 7, 122, 114, 106, 98, 90, 82, 74, 66, 124, 116, 108, 100, 92, 84, 76, 68, 126, 118, 110, 102, 94, 86, 78, 70, 128, 120, 112, 104, 96, 88, 80, 72, 121, 113, 105, 97, 89, 81, 73, 65, 123, 115, 107, 99, 91, 83, 75, 67, 125, 117, 109, 101, 93, 85, 77, 69, 127, 119, 111, 103, 95, 87, 79, 71}

This table determines the input permutation on a 128-bit block. The meaning is as follows: the first output bit is taken from the 44-th input bit; the second one from the 10th bit, and so on, with the final bit of the output taken from the 71st bit of the input.

-The length of the result is 512 bits

Step 5: working XOR operation between the result of step 3 and step 4, the length of the result is 512 bits

Step 6: aggregation the result produced from steps 2, 3, 4 and 5.

Step 7: present final key of 3584 bits

Example

Step 1-Input a = 1.4, b = 0.3, x = 0.6, y = 0.1

Step 2- Applying 2D henon map

For i=1 to 52

When i=1

X=1- (1.4 * (0.6 ^ 2)) + 0.1= 0.596

Y= 0.596 * 0.3 = 0.1788

When i=2

X=0.6814976

Y=0.20444928

And continue until i =53

-Remove the negative signal and Cut 15 number after the comma

Next

- The numbers output from x are:

{5966814976554234709671936736223854134392462031367485221839747231889198264678590714961981326918278488053805453269588979805326496959050084271015449981462910810956054138350173234979655149604758049717352124541981624532194520545333683462699794765459276530492810618530169818156091637905500196380759297926410842595404876223914400498204246990392705555763680182897901139754964604814522388006198231128522053605621625993410897871074698639604741139267741998262960597891705699643777111897835303718262804653862035547498300623343493057919886930221977927435298120455006656028269834513655207320590118140262089774211039196080442664889665129839767224464550603012497091189768188880841904279952973999098459562143365306182126641475397610300350204105700177327698532295111556206360895985844 3}

-The numbers output from y are:

{1788204449281662704129015812208671562403171386094102455662519241695667590794035772144882943980754835460161416359808762939415979490880150252813046352944388732432870162415050519702938965448814270149152056373622944872759658360163600105038802938429637782960147843185559052945446827491370165005891422772937792325277860146286717432012946127409711780166672910405482937034192648940 1}

4443567164018294693385566161016864877980232293613224095919014223417803225294788881793675017098931331335293505911154789013961586106642294901870030479017375966079066293378230589436013650199680848295035409656219017703544207862293226331175882013279946689953295193016733937018090374912735293045666425257012839858921997295378686430096018546379924426292830901050601231710053198229559688533466901908268795753 3}

Step 3: This step produced 2048 bits by using the number produced from x and y

596 mod 256= 84 =01010100 → XOR 11100110 → the first 8 bits

178 mod 256= 178=10110010

681 mod 256= 169= 10101001 → XOR 10011101 → the second 8 bits

820 mod 256= 52= 00110100

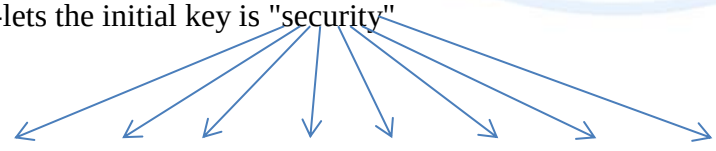
And continue until the output be 2048 bits

Step 4: Now expand the initial key by using the numbers produced from x, take the numbers in odd locations, cut 112 number, the output numbers are

{5684754379796323514942337821377389924750191836128803043658785246500821149842180504330724765464509132251864529504}

-**Note:** when the char is "" then char="a"

-lets the initial key is "security"



115 101 99 117 114 105 116 121

-Cuts the first 16 numbers {56 84 75 43 79 79 63 23}

01110011 xor 00111000 = 01001011 = 75 ="K"

01100101 xor 01010100 = 00110001 = 49 ="1"

01100011 xor 01001011 = 00101000 = 40 = "("

01110101 xor 00101011 = 01011110 = 94 ="^"

01110010 xor 01001111 = 00111101 = 61 ="="

01101001 xor 01001111 = 00100110 = 38 ="&"

01110100 xor 00111111 = 01001011 = 75 ="K"

01111001 xor 00010111 = 01101110 = 110 ="n"

- Now this word "**K1(^=&Kn**" be the initial key instead of the old initial key with the second 16 numbers and continue until the length be 64 bytes

-The final words are

"**securityK1(^=&Knxaaas3n'!=NS`!J+y>JwZo~aK6_y8EldOaXats,VaAAs4GsR**"

And covert each symbol to the binary of length 8 bits so the final result is 512 bits

The result is

{01110011011001010110001101110101011100100110100101110100011110010100101100
110001001010000101111000111101001001100100101101101110011110000110000101100
0010110000010111001100110011011011100010011100100001001111010100111001010011
011000000010000101001010001010110111100100111110010010100111011101011010011
011110111111001100001010010110011011001011111011110010011100001000101011011
000110010001001111011000010101100001100001011101000111001100101100010101100
1100001010000001010000010111001100110100010001110111001101010010}

Step 5: Applying a primitive table to the output of step 4**:**

The result is

{01111111111110010100101010101111000000001111111110100000000101011100100100
011010101110000101001100000000101101101101110111101001110111110001000111000
0001011111000000000111111110100000111110000010111000000010100000011010101011
00000000101100111100011011001100001111110111101101101010101001000000000111
010110111011101111110111011010001111011100110001011010000000011011010010111

01000001111111110110110100110100010010101100000000011110100100010110100001111011111011000001100000110111100000000010110010000000011101000}

Step 6: Applying XOR operation between the result of step 4 and step 5, the result is {00001100100111000010100111011010011100101001011011010100011011001000001000101011100100000000110100111011001000010010110100001111010011101110000101000011101111011100111100110000101111110101110111110100110111010010001111100001100000100100101000110011100111010001100100010100100000110111100101101010000100100000100100011111101001100010100010111001010101000011100010011111001100010110001110110010110101011000100101001010011101000000100101101001111110111100011010011001011100010001110000110100000111100111001110111010}

Step 7: Aggregation the output from steps 3, 4 , 5 and 6 to present the final key of 3584 bits.

Results

Implementation and evaluation result

1. The 5 Statistical Tests

Based on the results of the main 5 statistic tests which have been performed on (towards generating a new strong key for AES encryption method depending on 2D Henon map**)** as shown in the Table 2. The suggested modified methods showed better results, particularly on the test of frequency, considered as the most important one of the five tests.

Table 2: Results of the 5 Tests that have been applied to the Proposed key

Statistical Test					
Tests		Freedom Degree	First sample	Second sample	Third sample
Frequency Test		Must be <= 3.84	Pass =3.500	Pass =1.969	Pass =2.571
Run Test	T0	Must be <=19.391	Pass =13.400	Pass =14.150	Pass =14.275
	T1		Pass =10.246	Pass =7.879	Pass =12.875
Poker Test		Must be <=11.1	Pass =5.921	Pass =3.943	Pass =5.367
Serial Test		Must be <=7.81	Pass =4.464	Pass =4.339	Pass =4.929
	Shift No. 1		Pass =0.516	Pass =0.268	Pass =2.016
	Shift No. 2		Pass =0.755	Pass =1.613	Pass =1.698

Auto Correlation Test	Shift No. 3	Must be <= 3.84	Pass =0.023	Pass =0.175	Pass =0.047
	Shift No. 4		Pass =0.362	Pass =0.018	Pass =0.072
	Shift No. 5		Pass =2.850	Pass =0.342	Pass =0.785
	Shift No. 6		Pass =2.908	Pass =0.362	Pass =0.815
	Shift No. 7		Pass =0.846	Pass =1.181	Pass =0.618
	Shift No. 8		Pass =1.450	Pass =2.068	Pass =1.007
	Shift No. 9		Pass =0.148	Pass =0.123	Pass =1.110
	Shift No. 10		Pass =2.798	Pass =2.579	Pass =2.266

2. National Institute of Standards and Technology (NIST) Test Suite

The NIST Test Suite is a statistical package consisting of 16 tests and running on a binary sequence. The NIST test has been developed for the randomization test, testing the output of the key. These tests are based on a variety of different randomizations that can be found in sequence such as Randomness Excursions, Linear Complexity, Discrete Fourier Transform . . . etc. The following Table 3 is showing the resulted states of the proposals.

Table 3: NIST Test Suite of the Key from 2D Henon Map

NO.	Test name	Key from 2D Henon map
1	Frequency	Success
2	Block Frequency	Success
3	Cumulative Sums	Success
4	Runs	Success
5	Longest Run	Success
6	Rank	Success
7	Discrete Fourier Transform	Success
8	Non-periodic Templates	Success
9	Overlapping	Success
10	Universal	Discard
11	Approximate Entropy	Success
12	Random Excursions	Test not applicable
13	Random Excursions Variant	Test not applicable
14	Seri al	Success
15	Lempel-Ziv Compression	Success
16	Linear Complexity	success

3. Complexity

The proposed modified AES algorithm has been put in a comparison with standard AES algorithm as shown in Table 4 in order to measure the level of complexity. It is measured the complexity based on several measures such as the number of rounds, input and key sizes, brute force attack, key scheduling (key generation) in addition to the internal algorithms of the proposed algorithm.

Table 4: The Complexity Comparison of the Proposed Modified AES Algorithm against the AES Algorithm

Caption	AES Algorithm	Proposed Modified AES Algorithm
Rounds	10,12, 14 Rounds	14 Rounds
Input Block size	128 bits	256 bits
Key Block	128,192, 256 bits	256 bits
Brute Force Attack	2^{128} ,2^{192},2^{256} possible keys	2^{256} possible keys
Key Scheduling	Standard key generation	Generating New Strong Key for AES Algorithm Depending on 2D Henon Map, the generating of the key depending on: -Initial values of 2D Henon Map, this mean When the initial values of the chaos theory change, all the results change. -It is impossible to think of the initial values of the chaos theory because its number of probability is too large. -Initial key (8 bytes), and expansion the initial key from 8 bytes to 32 bytes,The expansion of the initial key is dependent on the results of 2D Henon map so that any manipulations in the coefficients of chaos theory will affect the subsequent results. -Which increases the difficulty of breaking the key is the primitive table. -Each step of the key generation depends on the results of the previous step and this has a big role in the difficulty of breaking the key
Internal Functions	Four layers -Sub byte layer -Shift row transformation layer -Mix column layer -AddRoundKey layer	Four layers • Sub byte layer • Shift Row Transformation layer • Mix column layer • Proposed Modified AddRoundKey layer
		AddRoundKey layer depending on the secret map, the secret map depend on 2D Henon Map, this mean this layer is sensitive to initial values of 2D Henon map

This below figure shows the result of 2D Henon map(equation x, equation y), the result of equations (x,y) after removing the negative signal and mid 16 numbers after the comma,show how to aggregate numbers produced from equation x in the buffer and how to aggregate numbers produced from equation x in the buffer

Towards Generating a New Strong key for AES Encryption Method Depending on 2D Henon Map

ALa'a Talib khudhair and Abeer Tariq Maolood

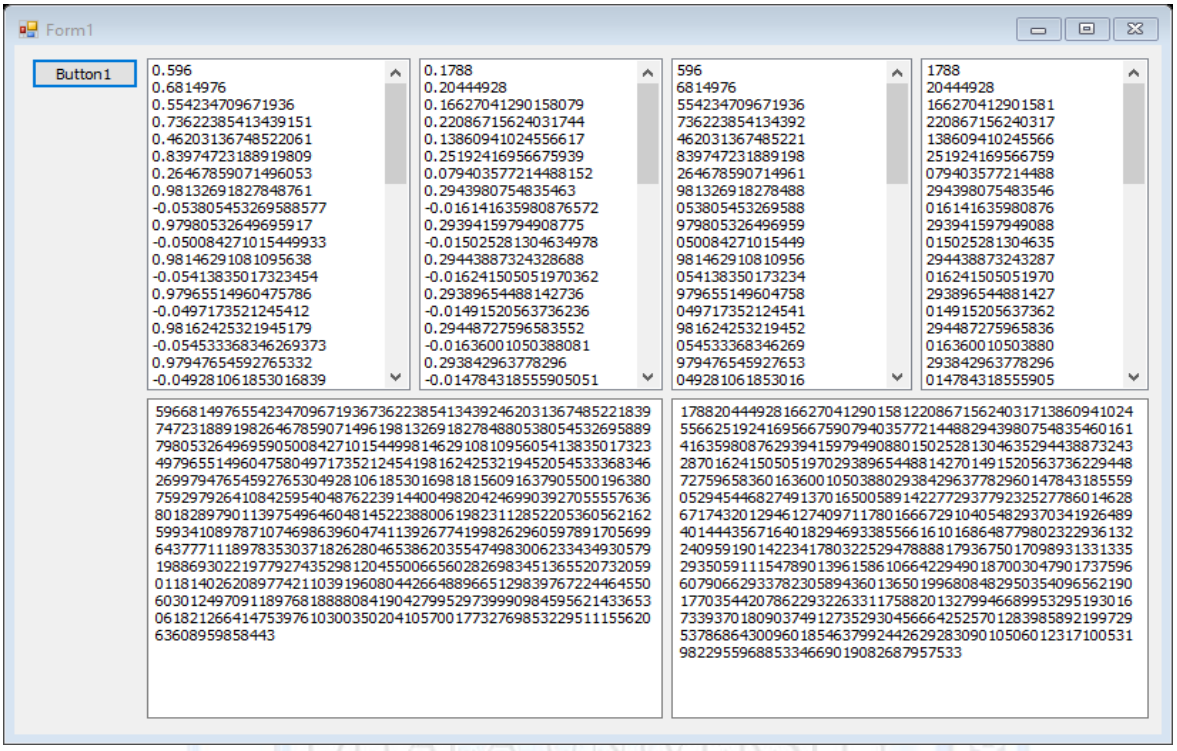


Figure 3: The Result of 2D Henon Map

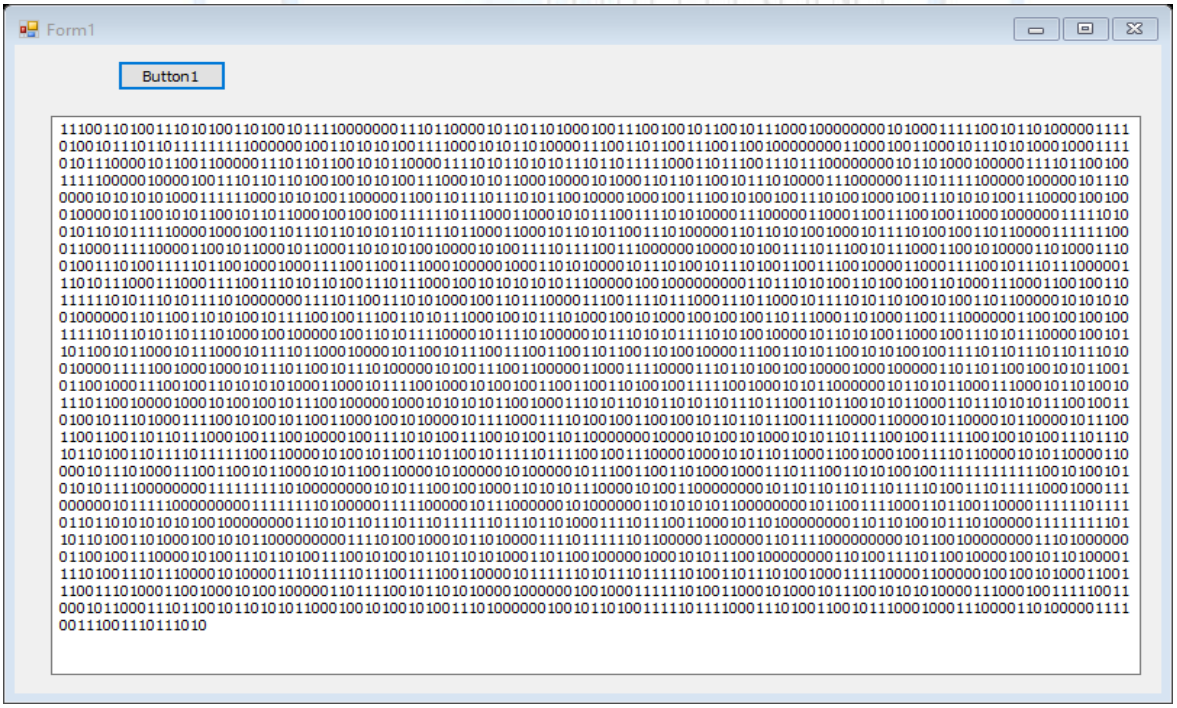


Figure 4: Final Key of Length 3584 Bits

Conclusions

When the initial values of the chaos theory change, all the results change. It is impossible to think of the initial values of the chaos theory because its number of probability is too large. The time taken is very small (parts of a second (0.0004 milliseconds) because the number of the equation x and equation y stored in a buffer to be available for any subsequent operations. The expansion of the primary key dependent on the results of the 2D Henon map so that any manipulations in the coefficients will change the results. Each step of the key generation depends on the results of the previous step and this has a big role in the difficulty of breaking the generated key. The results showed a high level of security for encryption on the basis of strong secret key features.

References

1. Vajargah, B. F.; Asghari, R. *International Journal of Mechatronics, Electrical and Computer Technology (IJMEC)* 2015, *5*(15), 2120-2129.
2. François, M.; Defour, D.; Negre, C. *Informatica* 2014, *38*(2), 115-124.
3. Jalil, A. P. D. L. F.; Saleh, H. H.; Albhrany, E. A. *IRAQI JOURNAL OF COMPUTERS, COMMUNICATION AND CONTROL & SYSTEMS ENGINEERING* 2015, *15*(3), 77-89.
4. Sava, D.; Vlad, A.; Tataru, R. In *2014* IEEE *10th International Conference on Communications (COMM)* 2014 May, 29-31.
5. Al-Shameri, W. F. H. Int. Journal of Math. Analysis 2012, 6(49), 2419-2430.
6. Vajargah, B. F.; Asghari, R. Sci. Int. (Lahore) 2015, 27(3),1797-1801.
7. Hamdi, M.; Rhouma, R.; Belghith, S. International World Academy of Science, Engineering and Technology, Journal of Computer, Electrical, Automation, Control and Information Engineering 2015, 9(2), 481-485.
8. Raj, M.; Shelly, G. American International Journal of Research in Science, Technology, Engineering & Mathematics 2014, 7(2), 111-116.
9. Sheela, S.; Sathyanarayana, S. V. *ACCENTS Transactions on Information Security* 2017, *2*(5), 1-15.

10. François, M.; Defour, D.; Negre, C. an international journal of computing and informatics 2014, 38(2), 115-124.

11. Jawad, L. M.; Sulong, G. Modern Applied Science 2015, 9(13), 85.

12. Boriga, R. E.; Dăscălescu, A. C.; Diaconu, A.V. *IAENG International Journal of Computer Science* 2014, *41*(4), 249-258.

13. Albhrany, E. A.; Jalil, L. F.; Saleh, H. H. *IJSRSET* 2016, 2(2), 67-73.

14. Albhrany, D. E.; Alshekly, T. *International Journal of Scientific & Engineering Research* 2017, *8*(1). 2114-2120.

15. Murillo-Escobar, M. A.; Cruz-Hernández, C.; Abundiz-Pérez, F.; López-Gutiérrez, R. M. *Microprocessors and Microsystems* 2016, 45, 297-309.

16. Kak, A. "AES: The Advanced Encryption Standard", Lecture Notes on "Computer and Network Security © Avinash Kak 2016, Purdue University, West Lafayette, Indiana, USA.