ISSN (Online) : 2454 -7190 **ISSN (Print) 0973-8975**

J.Mech.Cont.& Math. Sci., Vol.-13, No.-5, November-December (2018) Pages 225-240

A MATHEMATICAL MODEL OF INTEGRATED CHAOTIC BASED HASH FUNCTION TO IMPROVE RANDOMIZATION AND BIT VARIATION PROPERTIES OF CHAOTIC MAPS

[1]B. Madhuravani, [2]B. Rama, [3]N. Chandra Sekhar Reddy, [4]B.Dhanalaxmi, [5]V. UmaMaheswari

[1,2,3]Department of CSE, MLR Institute of Technology, Dundigal, Hyderabad, Telangana, India.

[4]Department of Information Technology, Institute of Aeronautical Engineering, Dundigal, Hyderabad, India.

[5]Department of CSE , Vardhaman College of Engineering, Shamshabad, Hyderabad, Telangana.

madhuravani.peddi@gmail.com, bulusurama1967@gmail.com, naguchinni@gmail.com, dinnu18@gmail.com, umasridhar11@gmail.com

https://doi.org/10.26782/jmcms.2018.12.00018

Abstract

In the present internet world, Security is a prime vital concern and the secure hash function is one of the ideal alternative means to guarantee security. In this paper we made a study on different nonlinear dynamical systems – Chaotic maps and introduced a novel hash scheme based on integrated chaotic maps. The experimental outcomes shows that the proposed model satisfies all cryptographic properties of secure hash functions such as resistant to collisions, high level of sensitivity to initial conditions, high confusion and diffusion, high randomization etc. The suggested model is fast and accurate in terms of speed and security is concern. In this model, multiple chaotic maps are integrated as a single chaotic system to generate an n-bit digest value, where the length of digest is flexible in terms of security is concern.

Keywords: Access Control, Authentication, Chaotic Maps, Complex Chaotic Maps, Integrity

I. Introduction

The primary goal of Cryptography is to ensure integrity, authenticity, confidentiality and access control of transmitted data over insecure communication channel. Cryptographic hash function is one of the key feature to provide integrity and authenticity by compressing an input message of approximate size to a result with fixed size, the hash code. They are used to provide Digital signatures, generating random numbers, deriving keys and updating , One way functions , detecting fraudulent code, user authentication (storing passwords) and many more. Keyed and un keyed are two different classes of hash functions which are used to construct a Message Authentication Code (MAC) and ensures the origin and integrity of private information.

The most preferable properties of hash functions are:

1. The hash function could take a message M of any kind of size as input.

2. The returned hash function h(M) is of dealt with fixed size.

3. Provided a message M, it ought to be very easy to calculate the hash function h(M).

4. Preimage resistance: it needs to be incredibly tough to locate any type of message M with an offered hash function h(M).

5. 2nd preimage resistance: offered a message M it need to be difficult to discover an additional message M0 such that h(M) = h(M0).

6. Crash resistance: it needs to be difficult to discover 2 distinctive messages M as well as M0 such that h(M) = h(M0).

A regular method to construct a hash function is based on Merkle Damgard scheme which iteratively uses a fixed-input size compression function. In the MD technique a message of approximate size is separated right into m bit blocks (with an extra padding procedure on the last block), that are sequentially infused right into compression function with collision resistant. The compression function produces an n bit message digest utilizing the input block as well as the previous intermediate outcome. After all input blocks have been processed it produces a final n bit hash value. The iterated hash function based on Merkle damgard scheme is depicted in Fig 1

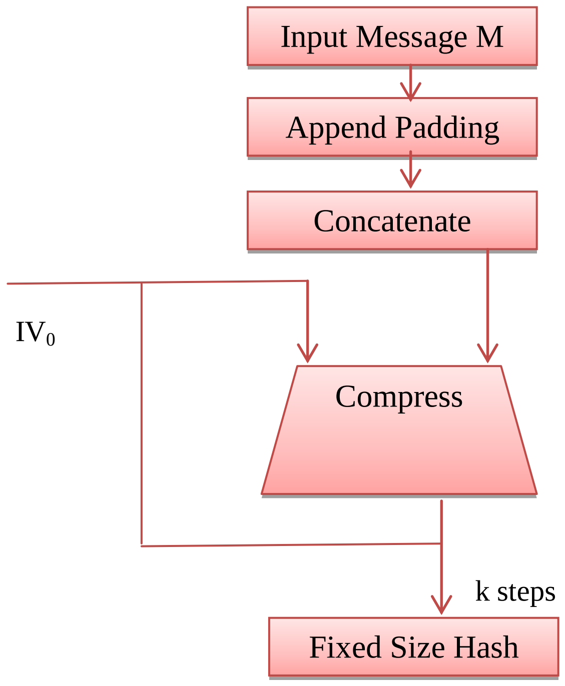

Figure 1. Merkle Damgard Scheme

The possible attacks on Merkle Damgard scheme are

Length-extension attack: The widely known weak point of the initial Merkle-Damgård building and construction is a size expansion residential or commercial property. If digests of messages M and also M ′ clash after that including a typical suffix likewise causes collision.

Multicollisions: Joux [VIII] proposed a multicollison attack. The suggestion is to construct collisions one after one more, which leads to 2k collision messages after just k tests of the collision search. If a hash feature has a repetitive framework, the attack could be constantly kept.

Fixed point attacks: It is specified by Dean[V] that for a repetitive hash function , if the solution factors of compression functions could be computed conveniently, after that locating second-preimages is less complicated compared to anticipated. Davies--Meyer building fits this problem well.
Many cryptographic hash functions are based on principles of MD4[XIV] proposed by Rivest in 1990. Rivest proposed MD5[V] in 2002 which produces 128 message digest similar to MD4 but it is slower compared to MD4. National Security Agency NIST has published Secure Hash Function SHA0[XI] in1993 with 160 bit hash. The collision attack with complexity 261 has discovered and published in 1998. NIST introduced another version of SHA as SHA1 with 160 bit hash and find a collision with the complexity of 269 was published in 2005. Another variant to SHA as SHA2[XII] with variable size hash bit as 224, 256, 384, 512 has introduced in 2001. SHA3[XIII] achieves variable length hash functions with Sponge Construction. All these hash functions are based on the basic Markle Damgard scheme which has proved insecure. Moreover, these hash functions are not keyed, multi-threaded and collision resistant. Hence a new hash function which is faster, secure and multi-threaded not based on Merkle Damgard scheme is desired in industry.

II. Chaotic Maps

Hash functions based on chaotic have actually obtained a great deal of attraction by the researchers in the area of big data, cloud computing, internet of things and also mobile and wireless communication. As a result of the restricted computing power, disorderly orbits will certainly come to be non-periodic. The majority of the chaotic based hash functions are refined in consecutive manner, which limits their implementation rate and also efficiency on the wireless communication. The major goal of a chaotic-hashing system is the chaotic systems convergence property which are complex. The features of these are:- a) Both are deterministic. b) Both are not foreseeable as well as complicated. c) In chaotic behavior a tiny modification in the first problems could influence and also show a substantial modification in the outcome. In hashing a small adjustment in the crucial or plain message will certainly customize the hashing result to a wonderful degree. As convergence, there are some attributes in charge of aberration of these 2 locations of study. Those are:- a) Chaotic systems are stood for by continual areas, yet hashing is stood for by limited as well as distinct spaces.

Di Xiao et.al[VI] introduced a Parallel keyed hash function construction based on chaotic maps. They used Piecewise Linear and 4D Cat Chaotic Maps and producing fixed Length Hash Value(128 Bit). This scheme is slow due to matrix multiplications and additions. Moreover, it is not applicable to Wireless Sensor Networks. as this model has poor diffusion and confusion probability, which uses Merkle Damgard Scheme. And also have high probability of collision. Di Xiao et.al[VII] proposed Improving the security of a parallel keyed hash function based on chaotic maps. This model has used Piecewise Linear and 4D Cat Chaotic Maps which consumes more time due to matrix multiplications and additions and producing a fixed Length Hash Value(128 Bit)where Output size less than 256 bits is insecure and is not applicable to Wireless Sensor Networks. This model has poor diffusion and confusion probability and high probability of collision. Shaojiang Deng et.al[XVI] Introduced Analysis and improvement of a chaos-based Hash function construction, which used 10D Cat Chaotic Maps and is slow due to matrix multiplications and additions. Moreover not applicable to Wireless Sensor Networks. The model has poor diffusion and confusion probability and high probability of collision.
A. Kanso et.al[I] Prosed a Keyed hash function based on a chaotic map, which uses a single chaotic map called Skew Tent Map, producing good statistical results for 128-bit hash values. This model is insecure as Merkle Damgard Scheme is involved and also not applicable to Wireless Sensor Networks. A. Kanso et.al[II] Introduced A fast and efficient chaos-based keyed hash function with 4D Cat Chaotic Maps, producing good statistical results for 128-bit and 160 bit hash values. Insecure due to Merkle Damgard Scheme having low probability of confusion and diffusion. Chenaghlu et.al,[IX] Introduced A novel key parallel hashing scheme based on a new chaotic system, which overcomes problems of one dimensional or multi dimensional maps (insecure/slow/attacked) with complex structure instead of Merkle Damgard Scheme. Not applicable to Wireless Sensor Networks as input parameter can easily predicted. Chandra Sekhar Reddy et.al[X] presented an empirical study on IDS and reviewed into his section provided insight related to security issues in cloud computing. It is found that stolen credentials are the major threat.

The research problems in the traditional models are depicted in the Fig 2. where all the models are having low bit variations and low randomization and moreover difficult to process the large data.

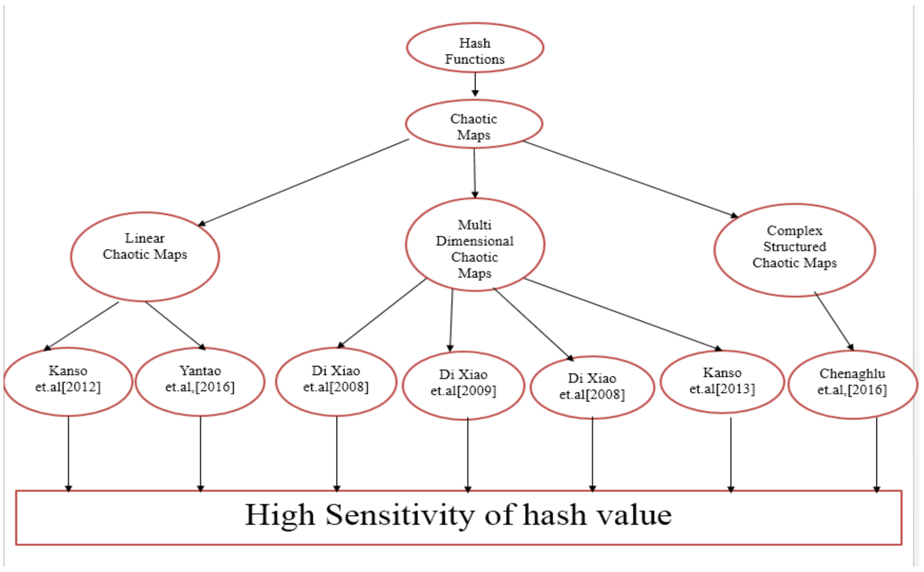

Figure 2: Traditional Chaotic Maps

III. Extended Chaotic Maps

Several chaotic maps have been proposed in the literature. They are sequential, multi-dimensional, parallel and structured. As most of the traditional chaotic maps listed in Table 1 are having low bit variation and low randomization. Moreover they can be easily guessed by frequency and statistical attacks.

Table 1. Types of Chaotic Maps

Chaotic Map	Domain	Dimensions	Parameters
Arnold's cat map	Discrete	02	00
Logistic Map	Discrete	01	01
Tent Map	Discrete	01	
Gauss Map	Discrete	01	
Henon Map	Discrete	02	02
Lorenze 96	Continuous	Arbitrary	01
Exponential Map	Discrete	02	01

We proposed a hash based chaotic model which performs a multi-threaded processing to handle massive amount of data. In this model we have extended three chaotic maps which are integrated and generates a single chaotic value to fed as input the model there by improves the bit variations and randomizations.
The extended nonlinear chaotic maps are:

Controlled Chebyshev Chaotic Map[III,IV]:

Let r be an integer from the set g onto g such that

$$f_r(x) : g \to g \text{ i.e. } [-1,1] \to [-1,1]$$

The equation is defined as $f_r = t \cos(r \arccos x)$

The recurrence relation is defined as

$$f_r(x) = (xf_{r-1}(x) - f_{r-2}(x))/t$$

Some of the possible equations in solving the recurrence relation with positive factor is given below. The basic condition for the recurrence relation is the degree of r and the bifurcation diagram is shown in the figure 3.

32x^5 - 40x^3 + 10x With positive factor 30248
64x^6 - 96x^4 + 36x^2 - 2 With positive factor 238142
128x^7 - 224x^5 + 112x^3 - 14x With positive factor 1874888
256x^8 - 512x^6 + 320x^4 - 64x^2 + 2 With positive factor 14760962
512x^9 - 1152x^7 + 864x^5 - 240x^3 + 18x With positive factor 116212808
1024x^10 - 2560x^8 + 2240x^6 - 800x^4 + 100x^2 - 2 With positive factor 914941502
2048x^11 - 5632x^9 + 5632x^7 - 2464x^5 + 440x^3 - 22x With positive factor 1386615384
4096x^12 - 12288x^10 + 13824x^8 - 7168x^6 + 1680x^4 - 144x^2 + 2
With positive factor 877037314
8192x^13 - 26624x^11 + 33280x^9 - 19968x^7 + 5824x^5 - 728x^3 + 26x
With positive factor 187020696
16384x^14 - 57344x^12 + 78848x^10 - 53760x^8 + 18816x^6 - 3136x^4 + 196x^2 - 2
With positive factor 1921764414
32768x^15 - 122880x^13 + 184320x^11 - 140800x^9 + 57600x^7 - 12096x^5 + 1120x^3 - 30x
With positive factor 1618733176
65536x^16 - 262144x^14 + 425984x^12 - 360448x^10 + 168960x^8 - 43008x^6 + 5376x^4 - 256x^2 + 2 With positive factor 1986727934
131072x^17 - 557056x^15 + 974848x^13 - 905216x^11 + 478720x^9 - 143616x^7 + 22848x^5 - 1632x^3 + 34x With positive factor 1390188408
262144x^18 - 1179648x^16 + 2211840x^14 - 2236416x^12 + 1317888x^10 - 456192x^8 + 88704x^6 - 8640x^4 + 324x^2 - 2 With positive factor 544844738
524288x^19 - 2490368x^17 + 4980736x^15 - 5447680x^13 + 3540992x^11 - 1391104x^9 + 321024x^7 - 40128x^5 + 2280x^3 - 38x With positive factor 1326397800
1048576x^20 - 5242880x^18 + 11141120x^16 - 13107200x^14 + 9318400x^12 - 4100096x^10 + 1098240x^8 - 168960x^6 + 13200x^4 - 400x^2 + 2 With positive factor 1728874750

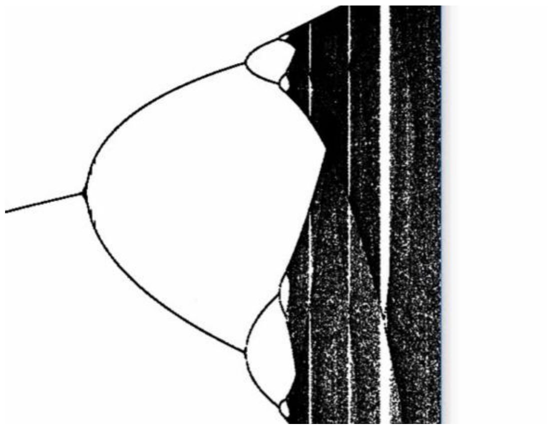

Figure 3: Bifurcation diagram of Controlled Chebyshev Map

Extended Quadratic Map[14, 15]:

The Quadratic Map is extended to increase the chaotic region there by improves the randomization.

The traditional equation $T_{n+1} = r - T_n^2$

Which is extended as $T_{n+1} = r - t.\ T_n^2$ where t lies between (0,2)

The cob web diagram and its corresponding wave graph is shown in Fig 4.

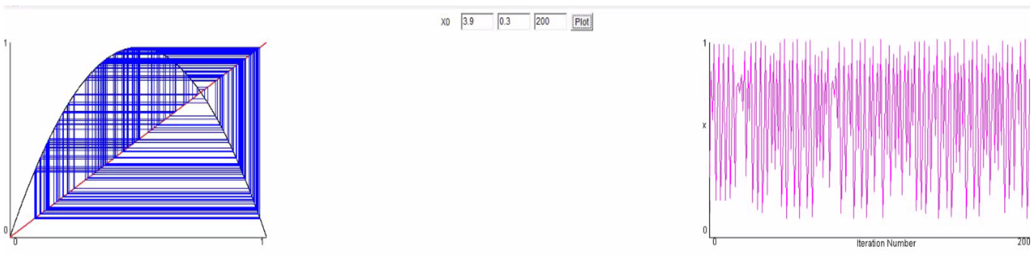

Figure 4: Cob Web diagram of Extended Quadratic Map

Extended Dynamic Chaotic Map [14,15]

The existing dynamic chaotic map used only two parameters α and β. Moreover the chaotic region can easily be predictable as the weighted parameters are fixed and as r is ranging from 0 to 16. This can be overcome by including a third parameter γ and defined as

$$EDCS = X_{n+1}(\alpha,\beta,\gamma) = w_\alpha \alpha(r.AX^2_n + w_\beta \beta(16-r).AX^2_n) \\ + w_\gamma \gamma(\frac{(w_\alpha \alpha + w_\beta \beta)}{2})AX^2_n$$

The random values for weighted parameters can be taken as
walpha 0.21 wbeta 0.69 wgamma 0.39 A 3301897.0 r 0.56 alpha 0.58 beta 0.43 gamma 0.18 X0 0.46 Final Value 503706.85333294404
walpha 0.77 wbeta 0.38 wgamma 0.22 A 1030469.0 r 0.4 alpha 0.49 beta 0.38 gamma 0.53 X0 0.35 Final Value 165558.42879634633
walpha 0.67 wbeta 0.84 wgamma 0.16 A 1138728.0 r 0.24 alpha 0.82 beta 0.72 gamma 0.8 X0 0.59 Final Value 2193645.2287514857
walpha 0.67 wbeta 0.59 wgamma 0.46 A 100938.0 r 0.27 alpha 0.47 beta 0.18 gamma 0.2 X0 0.21 Final Value 4230.091370564441
walpha 0.13 wbeta 0.2 wgamma 0.2 A 5527277.0 r 0.36 alpha 0.16 beta 0.2 gamma 0.83 X0 0.46 Final Value 40159.75210692122
walpha 0.18 wbeta 0.39 wgamma 0.62 A 455190.0 r 0.31 alpha 0.72 beta 0.13 gamma 0.27 X0 0.18 Final Value 5034.8096600566605
walpha 0.75 wbeta 0.46 wgamma 0.11 A 821026.0 r 0.81 alpha 0.21 beta 0.23 gamma 0.9 X0 0.15 Final Value 20628.005741207773
walpha 0.73 wbeta 0.21 wgamma 0.37 A 2374952.0 r 0.46 alpha 0.17 beta 0.34 gamma 0.49 X0 0.62 Final Value 225943.4083835771
walpha 0.3 wbeta 0.71 wgamma 0.55 A 4867136.0 r 0.64 alpha 0.41 beta 0.28 gamma 0.14 X0 0.18 Final Value 130147.66691627301
walpha 0.17 wbeta 0.45 wgamma 0.28 A 320942.0 r 0.72 alpha 0.89 beta 0.44 gamma 0.68 X0 0.18 Final Value 11398.883790162585
walpha 0.27 wbeta 0.72 wgamma 0.63 A 1726240.0 r 0.75 alpha 0.49 beta 0.57 gamma 0.16 X0 0.29 Final Value 173852.10708141018
walpha 0.64 wbeta 0.25 wgamma 0.48 A 4722467.0 r 0.83 alpha 0.56 beta 0.83 gamma 0.37 X0 0.35 Final Value 1173395.9742283646
walpha 0.41 wbeta 0.62 wgamma 0.37 A 1917336.0 r 0.46 alpha 0.56 beta 0.71 gamma 0.37 X0 0.49 Final Value 843373.4775181768
walpha 0.69 wbeta 0.17 wgamma 0.35 A 4352397.0 r 0.76 alpha 0.8 beta 0.87 gamma 0.46 X0 0.47 Final Value 2108585.7370586395
walpha 0.43 wbeta 0.47 wgamma 0.69 A 4599263.0 r 0.4 alpha 0.16 beta 0.48 gamma 0.65 X0 0.56 Final Value 515335.48997164815
walpha 0.86 wbeta 0.18 wgamma 0.29 A 25753.0 r 0.21 alpha 0.44 beta 0.46 gamma 0.9 X0 0.66 Final Value 7575.646764491245
walpha 0.4 wbeta 0.38 wgamma 0.51 A 1222387.0 r 0.77 alpha 0.63 beta 0.79 gamma 0.78 X0 0.22 Final Value 126845.91024895769
walpha 0.11 wbeta 0.13 wgamma 0.67 A 4119901.0 r 0.88 alpha 0.26 beta 0.69 gamma 0.85 X0 0.37 Final Value 79242.16686192679
walpha 0.82 wbeta 0.35 wgamma 0.51 A 971029.0 r 0.57 alpha 0.52 beta 0.58 gamma 0.11 X0 0.46 Final Value 386617.67971938004
walpha 0.34 wbeta 0.88 wgamma 0.76 A 595750.0 r 0.7 alpha 0.78 beta 0.66

gamma 0.89 X0 0.41 Final Value 310003.4851470876
walpha 0.78 wbeta 0.89 wgamma 0.79 A 545947.0 r 0.25 alpha 0.21 beta 0.82 gamma 0.38 X0 0.13 Final Value 21515.32126827173
walpha 0.33 wbeta 0.75 wgamma 0.78 A 3848562.0 r 0.25 alpha 0.45 beta 0.34 gamma 0.26 X0 0.17 Final Value 95175.0534185965
walpha 0.63 wbeta 0.6 wgamma 0.13 A 4970518.0 r 0.35 alpha 0.46 beta 0.57 gamma 0.58 X0 0.42 Final Value 1592631.8878236576

The bifurcation diagram for the above values is shown in fig 5.

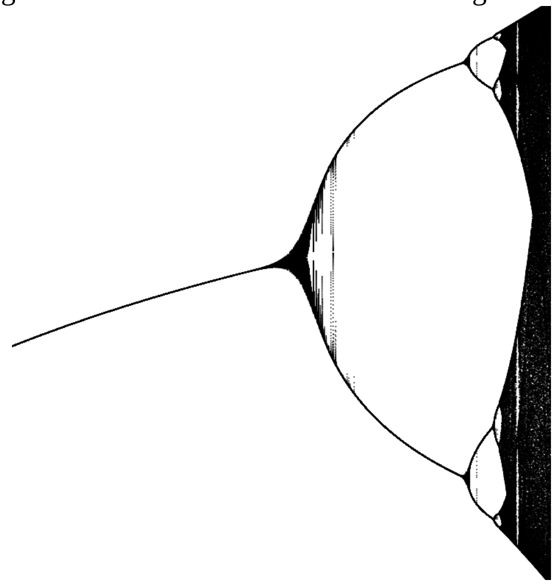

Figure 5: Bifurcation diagram of EDCS

IV. Proposed Model

Figure 6, represents the proposed extended chaotic model with parallel approach implemented on arbitrary input messages. The input text message M is divided into chunks of n- blocks, (b1, b2, b3…), each with b-length. Append padding bits $(10000 \ldots 00)_2$ with length ‘q’ at the end of the message M. After padding, each block is again divided into ‘m’ sub-blocks, each with 32-bit length and it is represented as P1, P2, P3… . Here, secret key is generated dynamically using the node-id. In this system, we have introduced an extended multi-chaotic system with three chaotic maps. Controlled Chebyshev Chaotic Map, Extended Quadratic Map and Extended DCS. In each round function, iterative multi-chaotic system generates output to the transformation box.

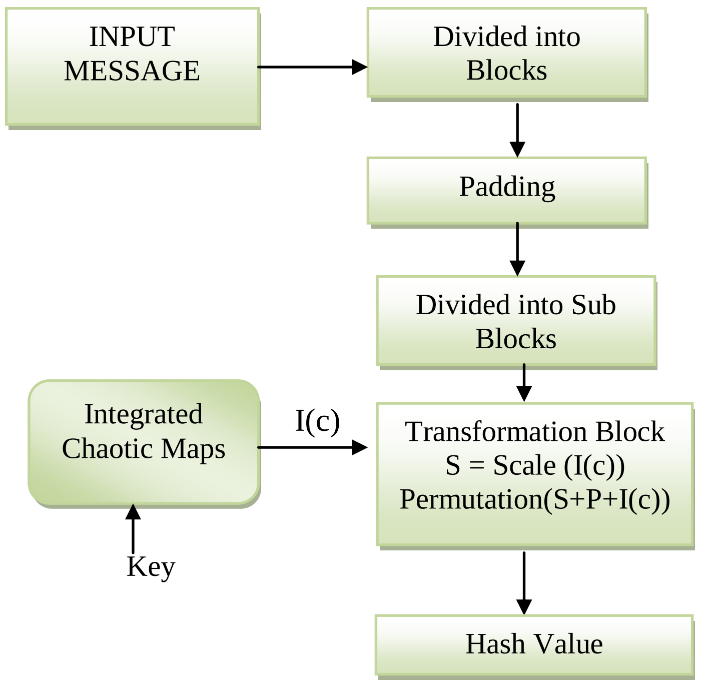

Figure 6: Proposed Integrated Chaotic Model

The permutation function is defined as follows for each 32-bits:

```
y |= (((int) mat_y.get(0)) << 3*8);
y |= (((int) mat_y.get(1)) << 2*8);
y |= (((int) mat_y.get(2)) << 1*8);
y |= (((int) mat_y.get(3)) << 0*8);
```

The overall pseudo code steps are summarized as follows:

INPUT:

- Secret Information.
- Key as Processor Id
- Initialization Parameters

OUTPUT:

Hash = H1+H2+H3…

Step 1: Take a message. If it is not in the desired multiple of r-length, append padding bits.
Step 2: divide the information into sub blocks.
Step 3: Generate chaotic values using integrated chaotic maps with secrete key as Processor ID.

Step 4: In each round function, iterative multi-chaotic system generates output to the transformation box.
Step 5: In the transformation box, the following operations are performed on the Y and chaotic output.
Step 6: Generates Hash value.

V. Experimental Results

The model is experimented and demonstrated for only 5 iterations. The procedure iterates till all sub blocks have been processed.

INPUT: Welcome

Proces block: [0x68, 0x65, 0x6c, 0x6c, 0x6f, 0x80, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05]

Partitioning block into sub block p[]: [0x68656c6c, 0x6f800100, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000005, 0x00000000, 0x00000005, 0x00000000, 0x00000005, 0x00000000, 0x00000005, 0x00000000, 0x00000005, 0x00000000, 0x00000005, 0x00000000, 0x00000005, 0x00000000, 0x00000005]

Iteration 1:

Processor ID eb94823e-9d55-4939-bf79-5c26c8d5da37
1100101110001011100111010011100011001011001111001011011011100111001
00110101110101101101110100111001110011111001101101110001011001101101
11111001101101110101110001111001011011011000111110001100100110101110
010011000011100111101111
Secret key
:110010111000101110011101001110001100101100111100101101101111001
C1 01101101
C2 00101001
C3 01000100

X[j] 00101001
Before scaling = 2147483647
After scaling 8388607
Y= 10101111
P[i]= 00100101
P'[i]= 10100011

Iteration 2:

Processor ID 85c537e7-6148-4c50-a98c-778cb447bf5d
1110001101011100011110101110011110111110010111011110110111011010001
1101001110001011011101001100011110101110000101101110000111100111100 0
1100011101101110111110111111000110001111000101101001101001101111100 0
1011001101101011100100

Secret key
:111000110101110001111010111001111011111001011101111011011101 10

C1 00000001
C2 00101001
C3 10010011
X[j] 10111011
Before scaling = 2147483647
After scaling 8388607
Y= 10000110
P[i]= 11011111
P'[i]= 11100010

Iteration 3:

Processor ID 16fba3ec-d180-4f09-8280-6c9f2174baec
1100011101101100110110001011000011100111100101110001110110111001001 1
0001111000110000101101110100110011011000011100110110111100011001011 1
0001100001011011101101100011111001110011011001011000111011111010011 0
0010110000111001011100011

Secret key
:11000111011011001101100010110000111001111001011100011101101110 0100

C1 00000001
C2 10111011
C3 11101111
X[j] 01010101
Before scaling = 2147483647
After scaling 8388607

Y= 11110101
P[i]= 01001101
P'[i]= 11101101

Iteration 4:

Processor ID fc089331-bd91-456b-97fe-43e13a276a43
11001101100011110000111000111001110011110011110001101101110001011001
00111001110001101101110100110101110110110001010110111100111011111001
10110010110110111010011001111001011100011100111100001110010110111110
1101100001110100110011
Secret key
:1100110110001111000011100011100111001111001111000110110111000010
C1 00000001
C2 01010101
C3 00111001
X[j] 01101101
Before scaling = 2147483647
After scaling 8388607
Y= 10010011
P[i]= 00010010
P'[i]= 11101100

Iteration 5:

Processor ID 14eee5b0-ee64-4cf4-bff3-54b4345c46de
11000111010011001011100101110010111010111000101100001011011100101110
01011101101101001011011101001100011110011011010010110111000101100110
11001101100111011011101011101001100010110100110011110100110101110001
111010011011011001001100101
Secret key
:11000111010011001011100101110010111010111000101100001011011100101
C1 00000001
C2 01101101
C3 00001001
X[j] 01100101
Before scaling = 2147483647
After scaling 8388607
Y= 01001100
P[i]= 01111011
P'[i]= 01010010

OUTPUT:

HashValue:618c3d5e3dd42cf23aaed42870c28ccb3395e4711513691d49b94ac80be75
1df392f98651a5af3e829a724f2664f398444b5a6070d0303fb36fa2e262544e6115a65d
9db6ad6b3385dcd59a726788e056fdd43386d25758655b8553c03ba5d1e7918765871a
a60ad6bf83519045baf597f1269df7c0f715a70bb992b5963892c

PERFORMANCE ANALYSIS

Proposed Model Vs Asgari[16]

Original Message: The cat jumps on the back of elephant.

Hash Value:
be412c4af3b1f7a3b29bde63db5d98e139dc3d1d083fcffb7e9d195e9c2d5b96

BinaryValue:
10000110010010010111000110101001010001101101101010010101000010011000
101101001001100100010001101110110010101100100111100000001010001100001
101000111011011111110000011110101010110101110011010000110101101001110
01111110000000000101011110000110111010010100001000101

Modified Text: The cat jumps on the back of elephant**?**

Hash Value
e5bf884791614cd17938e90e4feac752d966749cfa91552b7d33b77f10c711aa

Binary Value
01100011111101101100101011101110110101111011011111011001110111001111
00100111000101111000011101010111110101010100100011110100011011011011
11000101110000010101110100001111110010111010010110101101110010011101
1100010101110111011111100011000110110011100111111111

Proposed Bit Change:150
Asgari[16] Bit change:138

We perform different experiments to find the efficiency of our parallel hash function against traditional models. We also provide experimental results with traditional hash models in terms of hash sensitivity. Hash sensitivity of the input message to the changed message are evaluated using different conditions. Experimental results demonstrated in Table 2 and Fig 7 have the high sensitivity of the original to the changed one in different cases.

Table 2. Statistical Analysis of Proposed Model to Existing Models in term of Bit Variations

Models	B(min)	Avg(B)
Kanso et.al[19]	118	129
Yantao et.al[17]	117	127
Tian et.al[18]	120	131
Asgari et.al[16]	126	136
Proposed Integrated Chaotic Model	133	144

Table 2, describes the performance of the proposed model compared to the traditional models in terms of bit size, hash sensitivity. From the above results, it is clear that any change in the original message has a huge impact on the final output. From the table, it is clearly observed that the proposed model has high hash sensitivity compared to traditional models.

VI. Conclusions

In the proposed model the integrated chaotic model is presented to improve the random ness property by enhancing the chaotic region. The experimental results have proved that proposed model is having high sensitivity in terms of bit variations, thereby improving the randomization. The model has to be extended to support different kinds of messages.

References

I. A Kanso , H. Yahyaoui b, M. Almulla, “Keyed hash function based on a chaotic map”, Information Sciences 186 (2012) 249–264, Elsevier.

II. A Kanso, M. Ghebleh, “A fast and efficient chaos-based keyed hash function”, Commun Nonlinear Sci Numer Simulat 18 (2013) 109–123, Elsevier.

III. B Madhuravani, Dr. D.S.R. Murthy, “An Efficient Authentication Protocol to amplify collision resistance using Dynamic Cryptographic Hash Function & LSB Hop based Image Steganographic Technique”, International Journal of Applied Engineering Research, Volume 11, Number 7 PgNos: 5293-5296.(2016), ISSN 0973-4562

IV. B Madhuravani, Dr. D.S.R. Murthy, “A NOVEL NODE INTEGRITY BASED AUTHENTICATION MODEL FOR DYNAMIC WIRELESS COMMUNCATION NETWORKS”, Journal of Advanced Research in Dynamical and Control Systems, ISSN: 1943-023X Issue: 12-Special Issue, (2017), Pages: 1145-1169

V. Dean RD. Formal Aspects of Mobile Code Security. PhD thesis, Princeton University; (1999).

VI. Di Xiao a,b,, Xiaofeng Liao a, Shaojiang Deng , "Parallel keyed hash function construction based on chaotic maps", Physics Letters A 372 (2008) 4682–4688, Elsevier.

VII. Di Xiao a, Xiaofeng Liao a, Yong Wanga,' Improving the security of a parallel keyed hash function based on chaotic maps", Physics Letters A 373 (2009) 4346–4353, Elsevier.

VIII. Joux A. Multicollisions in iterated hash functions. In: CRYPTO'04, LNCS, vol. 3152; (2004). p. 306–16.

IX. Meysam Asgari Chenaghlu ∗, Shahram Jamali, Narjes Nikzad Khasmakhi," A novel keyed parallel hashing scheme based on a new chaotic system", Chaos, Solitons and Fractals 87 (2016) 216–225.

X. N. Chandra Sekhar Reddy, Dr. Purna Chandra Rao Vemuri, Dr. A. Govardhan, Ch. Vijay, "An empirical study on feature extraction techniques for Intrusion Detection system", Journal of Advanced Research in Dynamical and Control Systems, ISSN: 1943-023X Issue: 12-Special Issue, (2017),Pages: 1118-1130.

XI. NIST, Secure hash standard (SHS), federal information processing standards 180; (1993).

XII. NIST, secure hash standard (SHS), federal information processing standards 180-1; (1995).

XIII. NIST, secure hash standard (SHS), federal information processing standards 180-2; (2002).

XIV. Rivest R. The MD4 message digest algorithm. In: CRYPTO'90, LNCS, vol. 537; 1991. p. 303–11.

XV. Rivest R. The MD5 message digest algorithm, request for comments (RFC) 1321. Internet engineering task force; (1992).

XVI. Shaojiang Deng, Yantao Li *, Di Xiao, «Analysis and improvement of a chaos-based Hash function construction", Commun Nonlinear Sci Numer Simulat 15 (2010) 1338–1347, Elsevier.

XVII. Tian-Fu Lee, Efficient three-party authenticated key agreements based on Chebyshev chaotic map-based Diffie–Hellman assumption, Nonlinear Dynamics ,2014, vol 10,pp 23-43.

XVIII. Yantao Li ,Di Xiao, Parallel chaotic Hash function construction based on cellular neural network, Neural Comput & Applic (2012) 21:1563–1573.