Int. J. Advance Soft Compu. Appl, Vol. 9, No. 2, July 2017
ISSN 2074-8523

Design of Secure Cryptographic Hash Function Using Soft Computing Techniques

S.Santhanalakshmi, Sangeeta K, G K Patra

Department of Computer Science and Engineering, Amrita School of Engineering, Bengaluru,
Amrita Vishwa Vidyapeetham, Amrita University,India.
s_lakshmi@blr.amrita.edu
Department of Computer Science and Engineering, Amrita School of Engineering,Bengaluru,
Amrita Vishwa Vidyapeetham, Amrita University,India.
k_sangeeta@blr.amrita.edu
Council of Scientific and Industrial Research, Fourth Paradigm Institute, Bangalore,India.
gkpatra@csir4pi.in

Abstract

Data integrity is a crucial part of any secure system. Cryptographic hash functions serve as a basic building block of information security for ensuring data integrity and data origin authentication. They are used in numerous security applications such as digital signature schemes, construction of MAC and random number generation. A hash function takes an arbitrary amount of input and produces an output of fixed size. Many of the widely used cryptographic MD-5 and SHA-1 hash functions have been shown to be vulnerable to attacks. The non linear behavior of the neural network model which takes multiple inputs to produce single output makes it a perfect entrant for cryptographic hash design. The paper describes the construction of a cryptographic hash function using a multi layer Tree Parity Machine neural network. Although in our simulations we have considered 512 bit message blocks which generate 128 digit hash value, the proposed algorithm can be used flexibly to generate a hash function of arbitrary length. Simulations show that this hash function satisfies the security requirements of confusion, diffusion, and collision attack

Keywords: *Hash Function, Tree Parity Machine, Neural Synchronization, Collision, Message Authentication Code*.

1 Introduction

Data integrity provides assurance of data non- alteration either in transit or in storage, which is essential in business involving electronic commerce. This type of data assail can be prevented using a verifiable mechanism which strengthens integrity and provides an assurance that the data transactions, communications have not been intervened. Data integrity in cryptography is achieved by means of cryptographic hash functions, which can detect any modification and hence guarantee the integrity and source of information [1].

Cryptographic hash functions may be unkeyed or keyed. Unkeyed hash function also known as *Modification Detection Code (MDC)* take a single parameter i.e., message as an input. The most popular Modification Detection Code are SHA-1 and MD5 [2] .SHA-1 is useful in "security applications and protocols" and MD5 is used for "verification of integrity of electronically transmitted files". However, successful collision attacks on the above cryptographic hash functions as discussed by Wang et al. [3] makes it unsecure and weak. This weakness of MD5 apparently allows attackers to create multiple input sources to MD5 mapping that results in the same output. Wang et al. [4,5] have found hash collisions in the 160 bit SHA-1 in approximately 2^{63} operations rather than 2^{80}. Unkeyed hash functions without a key only provide data integrity [6,7] but no authentication.

Keyed hash functions, on the other hand depends on two inputs – a message and a secret key to construct the *Message Authentication Code (MAC)*, which provides data integrity as well as data origin authentication and is denoted by

$$\text{MAC} = h\,(m \,||\, k)$$

Bellare et al [8-9], proposed "Keyed-Hashing for Message Authentication" (HMAC) which was shown to be secure against a number of subtle attacks. HMAC was defined here as $H\,(key||H\,(key||message))$, which performs an extra computation, i.e., the transitional result of the internal hash is masked by the outer application of the hash function which leads to enhanced security. Also HMAC-MD5 does not endure from the same weakness of hash collision that has been found in MD5. It is summarized in RFC 6151 [10] that even if the security of the MD5 hash function itself is sternly compromised, collisions are not an issue in HMAC-MD5, which use inputs (the key) unknown to the attacker in construction of message digest. The most frequent attack in opposition to HMACs is brute force, which depends on the size of the key. Hence, the cryptographic power of the HMAC increases with the dimension of the secret key. Also HMAC calls for secret key management [11], i.e., during the operation the system must either keep the secret key all the time or access from its secured storage during the process.

The challenge of key management can possibly be overcome by dynamic key generation at the time of communication. A number of dynamic key generation procedures have been proposed by using the concept of neural network. Neural networks due to its confusion, diffusion and non linearity properties [12] are extensively used in the design of cryptographic protocols and hash functions [13,

14]. Shiguo et al., [15,16] described the construction of hash function based on neural networks. The authors demonstrated how one-way and diffusion property of neural network ensures reasonable output. The hash function proposed by Shiguo requires more arithmetic operations (mul/div/add/sub), which can be drastically reduced in a parallel-realization.

Neural Network based secure hash function that uses "*Piecewise Linear Chaotic Map (PWLCM)"* and key generator, as an activation function and for initialization of parameters was proposed by Lian et al. [17]. This could execute only in sequential mode, and a parallel keyed hash function based on the chaotic neural network (CNN) was later proposed by Xiao et al. [18]. Limitation of this algorithm lies in the secret key (nonce numbers) i.e., the keys lose their security level if used more than once and would lead to some security flaws. These flaws have been prevailed by Z Huang [19], where he uses hash mixer which consists of two parts. The first part to determine the intermediate hash value and in the second part to establish the transverse connections to the different part of the hash value by using yet another double layer CNN. Using *PWLCM* and a "*4-Dimensional One-Way Coupled Map Lattices (4D OWCML)*" as an activation function and key generator a new hash algorithm based on two-layer CNN was proposed by Li et al. [20]. ln all of these above hash algorithms based on neural network, key generation and hash value generation are two distinct functions.
In our work, we are proposing using of a similar multi layer neural network called Tree Parity Machine (TPM) for both key and hash generation. The following section explains in detail how the key is generated using neural key exchange process. The hash generation algorithm described in section 3 uses this generated key to construct the hash of a given message. The projected hashing scheme make a secure hash function which satisfies the required confusion and diffusion properties. This has been described as a part of performance analysis in section 4. Comparison and Computational analysis has been discussed in section 5. With the help of simulations, conclusions are described in section 6.

2 Key Generation Using Neural Cryptography

Exchange of secret keys over public channels using a variety of learning rules offer an pleasing alternative to number theory based cryptography algorithms. Neural key exchange, discussed by Kanter et al. [21] is based on the harmonization of two neural networks.

The Tree Parity Machine (TPM) shown in Fig 1 is an exceptional class of multi-layer neural network which contains one output neuron (τ), *'K'* hidden neurons ($\sigma_{k,\ k=\ 1\ldots K}$) and *'N'* input neurons ($X_{ik}, i = 1 \ldots n$).Inputs to the network take bipolar values: $X_{ki} \in \{-1,1\}$. The discrete random weights W_{ki} are initialized linking

input and hidden neurons in the range (-L, L). The value of each hidden neuron is computed as $\sigma_k = \mathrm{sgn}\left(\sum_{i=1}^{N} X_{ki}\, W_{ki}\right)$ and the final output $\tau = \prod_{k=1}^{K} \sigma_k$.
Training two such TPM'S on their common output coordinate to an identical time dependent weight vector. The identical weights obtained from the synchronization are the keys. This process has been used for construction of secure cryptographic secret-keys on a public channel. Substitution of random weights with optimal weights generated by genetic algorithm leads to a faster synchronization of the TPM as discussed by us [22]. The TPM's using the genetic approach, was able to coordinate faster because of the best optimal weights achieved from genetic process. These optimal weights not only fasten the synchronization process but also considerably reduce the probability of attackers. This method of key generation is efficient in the sense that for each message if needed a fresh key can be generated without any data been stored internally. The efficiency of the network can be further improved by increasing the number of neurons either in the input or hidden layer [23] to get a secure key. The key so generated is tested for its randomness [24] and the same can be used to define a hash function which will be described in the next section.

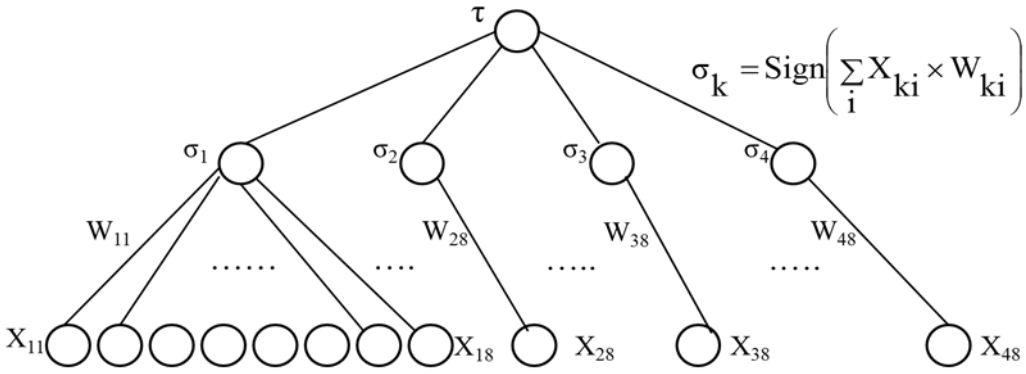

Fig 1: Tree Parity Machine for Key generation

3 Proposed Cryptographic Hash Function

The HMAC uses the cryptographic hash function in association with the secret key.

Keyed Hash (MAC) = Hash (Key || Message) Eq 1.

"Hash functions take in data of arbitrary length as input called message M and produce an output of fixed size often referred to as message digests" (RSA Laboratories) represented by *H = h(M).* The properties that good hash functions should possess are as follows [28].

1. *"Pre- image resistance*: Given *H(x),* it should be tough to find *x.*
2. *Second pre- image resistance*: Given message m_1, it should be hard to find another message m_2 such that

$$H(m_1) = H(m_2)$$

3. *Collision resistance*: It is hard to find any two distinct messages m1, m2 such that

$$H(m_1) = H(m_2)"$$

In this section we describe the generation of hash function using the TPM network. Here the TPM network is used for generating key as well as for the computation of Hash.

The generation of keyed hash comprises of two stages, viz. pre processing stage and computation stage.

In pre-processing stage, the message is split into blocks (B_i, *i = 1 to N*) of size 512 bits each. The length of the message expressed as a 64 bit binary, is appended to the message. Between the end of the message and the length field, a pad is inserted so that the message padding length is a multiple of 512, the block size. Each block (B_i) is further split into an array of 16 words, each 32 bit wide as shown in Fig 2.

Procedure of Hash Computation using the TPM network can now be described as follows:

For each block B_i, of 512 bits with *i = 1 to N*,

The neural network shown in Fig 3 is initialized at time t =1 with input neurons M_{ij} (i= 1 to 8, j= 1 to 4) obtained from the first 32 bits of the message M_t. The weights (K_{ji}), (i=1 to 8, j=1 to 4) in the TPM corresponds to the key generated from the synchronization of two TPM'S as described in section 2 . Following steps are now repeated iteratively from t=1 to t=16

- The hidden values $\sigma_j = (\sum K_{ji} M_{ji})$ are computed and hex values of $(\sigma_j, j = 1..4)$ are saved as H_t.
- The output $\tau = \prod_{j=1}^{K} \sigma_j$ of the network is computed and compared with $sgn(\sigma_j)$
- Weight vectors are now updated at next time step for those branches of the TPM where $\sigma_j = \tau$ as per the relation

  $K_{ji}(t+1) = K_{ji}(t) + M_{t+1}$

  $H = H_t \mid H_{t+1}$

Repeat this until the whole message is consumed from B_i , Finally 512 bit MAC value is obtained by performing AND operation on all B_i, *i=1 to N*.

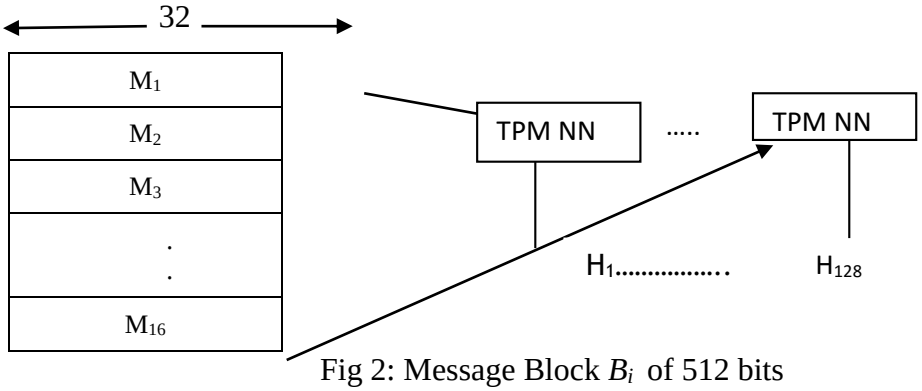

Fig 2: Message Block B_i of 512 bits

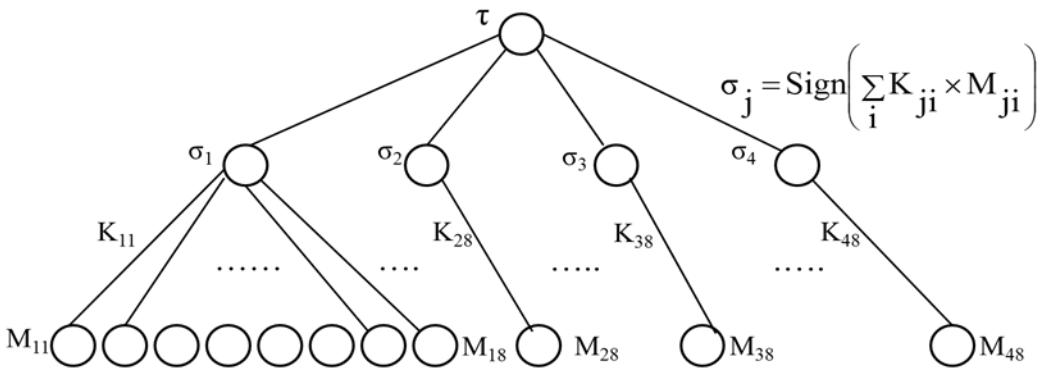

Fig 3 TPM for Hash Computation

4 Performance Analysis

Hash function should satisfy the properties like one way and Collision resistance. We extremely assess the proposed algorithm, and the results exemplify that it has good potential of confusion and diffusion, and strong collision resistance. The performance analysis of the proposed hash function is discussed in this section.

4.1 Diffusion Analysis:

A hash function has to be sensitive to any small modification in the input message, so that it will be difficult to predict the original message from its hash value. It is expected that any small change in the message or secret key input should lead to 50% change in its hash value, where the difference between two hash values is measured in terms of their Hamming distance. To test the sensitivity the following input message has been considered “The quick brown fox jumps over the lazy dog”.

Diffusion is associated with the dependency of the output bits on the input bits. Here we have considered the messages with the following cases, such as adding null space at the end of the message, replacing a word by another word and so on and their hash values are computed. The changes in the computed hash value for each case is calculated and shown in Table 1.

Table 1: Sensitivity of the Hash Algorithm for the input message “The quick brown fox jumps over the lazy dog”

Case No.	% Change in Original Message	Hash Value (512 bits)	% Change in the Hash Value
1	0	00f6f2f7fe06f8010404e0fdf4fe0201e2eefeff020606f7fc0804f 1e808f0e90a08f2f5fe00fcfbeef4faedfe06f8f7fe06f6f5fc06f8f3	0

		fa06fe6ff606fc2f	
2	1-2	0e090410f50010c02eb06ffe2050603e010c0e03f8f5f2ede2e3e8edfefff83f4f1f605feef0eff00ff0ef7f0ebf4f112e95df60902edf60700edf609feedf40	>80
3	2-3	f8801020502eb0607e0090603f403f6fbf6ede0e5e8ed0007f3f8f1ec05feef0c0f3eee9eced0ce3fc036e9fc0906e9fc0704e9f30f0ff02c0902e9fc0ffe110	>90
4	3-4	1fde800bedf2f8e3ebf6fa5bf7e0c090cfe0ef207e5f80621fffe041df3f2f4120c0908fac090afe0c0906f80c0906fa0c0908fc0c0908fc0c0904fc c20904fc	>80
5	4-5	b9f7ecfa11fffcec30ff504007ede4f807e7f8fc37f5e6001d0304f2e50100204e32100f08719fa06a50df60423fa00a111b02060ffb02040ff904060ff9080c	80-90
6	5-6	eccfa2f7c558a047269ca3b0a61dd2c546ac0d79fcc903d0f6435d6a63fc74daba413a06d627a96620a2625148c57c79a099cf9c502273a8759ebec4f8572fed	>90

For ideal diffusion, any slight modification in the plain text should lead to 50% change in the bit series of the message digest. To get a deeper insight into the diffusion property of the discussed hash function the input message M of 1024 bit length is taken and its corresponding hash value is computed. By modifying the i^{th} *(i=1...1024*) bit of M, the new message M′ and its corresponding hash value is generated. It has been observed that the hamming distances vary between 163 and 350 for above messages. The average hamming distance is 256.5. Ideally the hamming distance between the original message and the modified message is expected to be 256 which agree with our observed result.

In order to analyze the diffusion effect [25, 26] we consider the binary form of text given in previous case (1) as original message and compute its 512-bit hash value. This message is now modified by selecting a random bit and flipping it. The hash value of the modified message obtained after flipping one bit is then computed. The Hamming distance is used to measure the change in the hash value arising from any change in the original binary message. The Hamming distance between this hash value and the hash value of the original message is found. The above procedure is repeated N times and the following measures are computed:

- *"Minimum Hamming Distance* : $B_{min} = min(\{B_i\}_1^N)$
- *Maximum Hamming Distance* : $B_{max} = max(\{B_i\}_1^N)$
- *Mean Hamming Distance* : $\bar{B} = \sum_1^N \frac{B_i}{N}$
- *Mean probability* : $P = \frac{\bar{B}}{512} * 100\%$
- *Std. dev of the Hamming Distance* : $\Delta B = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(B_i - \bar{B})^2}$
- *Standard deviation*

$P = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\frac{B_i}{512} - P)^2} \; X100\%$"

Table 3: Statistics of number of changed bits.

Statistical measures	N-256	N=512	Mean
B_{min}	162	162	162
B_{max}	350	352	355
$\bar{B}$	256	257	256.5
$P\%$	50	50.19	50.1
ΔB	10.31	10.25	10.28
ΔP	4.72	4.86	4.79

It is clear from the resultant data in Table 3 that the mean altered bit number (B) and the mean altered probability (P) are both very close to the ideal value 256 bits and 50 %. Also, ΔB and ΔP indicate little deviations about the mean value, which in turn indicates, the projected hashing method has very strong potential for confusion and diffusion. Thus, the above proposal is safe against diffusion attacks.

4.2 Distribution of hash value

The most significant properties, that is honestly associated to the security of the hash function is the uniform distribution of hash value. To show this the ASCII values and the message digest for the message *"Hash function takes a message as input and produces an output referred to a hash value. A hash value serves as a compact representative image of input string. Hash function takes a message as input and produces an output referred to a hash value. A hash value serves as a compact representative image of input string"* is generated and it has been observed in Fig 4(a-b) that Hexadecimal values for the message are confined to a small area, while the hash value in hexadecimal spreads around the entire area.

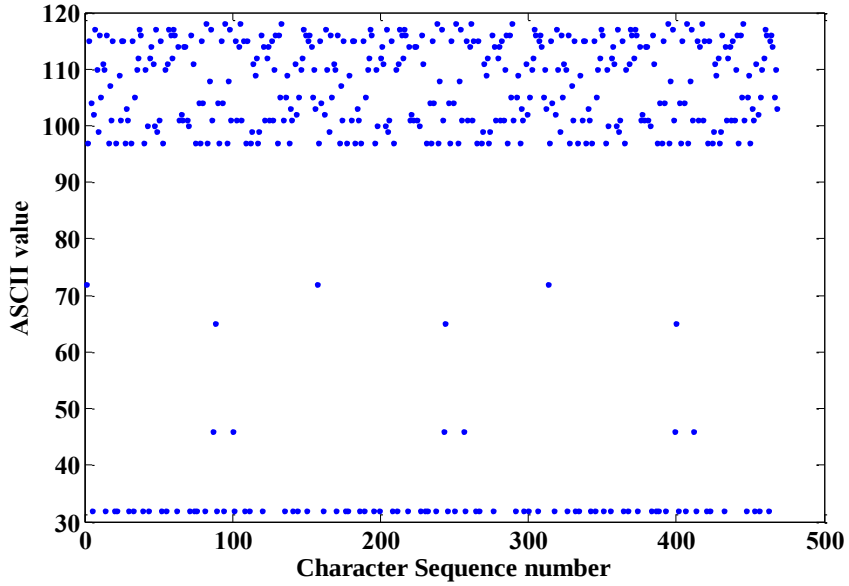

Fig 4a Distribution of the message in ASCII code

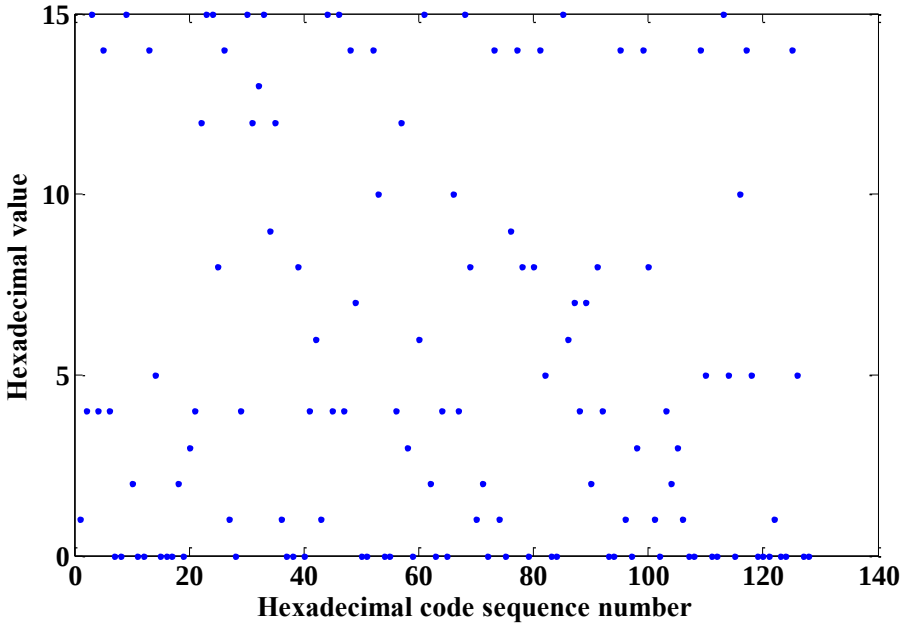

Fig 4b Distribution of the hash value in hexadecimal format.

Another similar experiment shown in Fig 5(a-b) is done with a message equal to 0. Even in this intense condition we can see the spread of the hash value of the message "0" is still uniform

Spread of message and hash value: (a) distribution of the message in ASCII code and (b) distribution of the hash value in hexadecimal format.

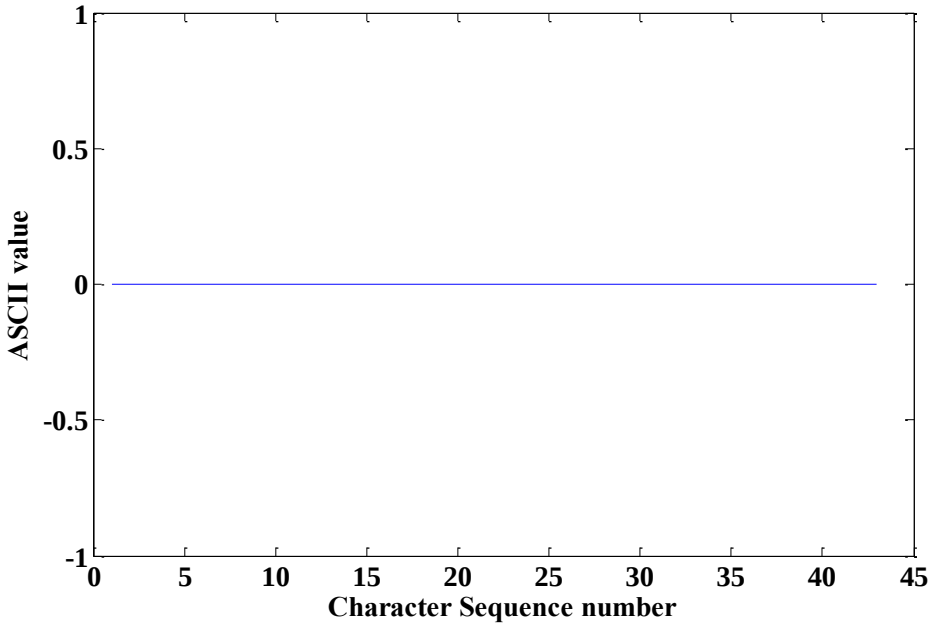

Fig 5a: Distribution of message '0' in ASCII code

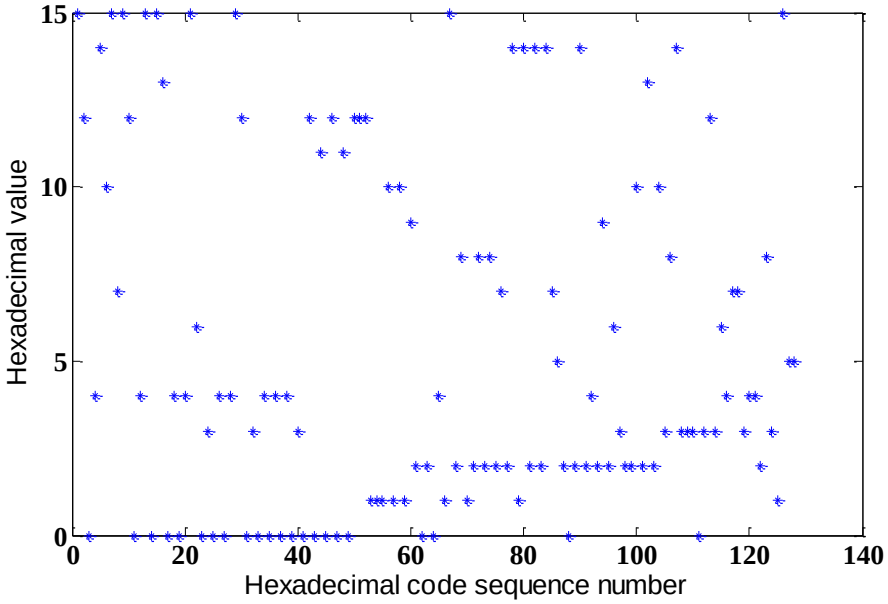

Fig 5b: Distribution of hash value of '0' message

4.3 Collision analysis

"A hash function is collision resistant if it is computationally infeasible to find two messages M, M' that will get hashed to the same output i.e., h(M) =h(M')".

Birthday attacks are similar in idea, where *two random input data have to be found such that they hash to the same output.* The safety of the hash function for birthday attacks is resolute by the length of the hash value, which is 512 bit in our proposed function, hence the attack difficulty is 2^{256} [27]. Further by modifying the size of neural network, the hash value for larger bits can be obtained. Therefore, even with the advancement in computing power, the proposed hash function can defy this kind of attack.

4.3.1 Collision test

In turn to investigate the collision resistance capability of the hashing approach the following collision test [28] is carried out: First, the input message given in section 4 is chosen randomly and the message digest is produced and stored in ASCII format [29-30]. Then a bit series in the binary form of the message is chosen arbitrarily and changed and new digest is then generated and stored in ASCII format. The absolute difference (ad) of the two hash results is computed by using the following formula:

$$ad = \sum_{1=1}^{N} |dec(e_i) - dec(e_i')|$$

here e_i and e_i represent the i^{th} ASCII character of the message digest for the original message and the modified message, respectively, and dec( ) maps these ASCII values to their corresponding decimal values. This simulation is run for 1000 times, each time changing the random bit of the original message. The absolute difference of the hash values is recorded every time. From this sequence of hash values the maximum, minimum and mean values of the absolute difference are obtained as shown in Table 3.

Table 3: Absolute difference for hash values of length 512

Maximum	Minimum	Mean
7421	4671	2603

The above results suggest that a small change in input leads to a considerable change in the output message so having the probability of having two messages with same output is very low. Fig 6 shows the number of positions shared by the 512 bit hash values where the ASCII values are same for N = 1000.

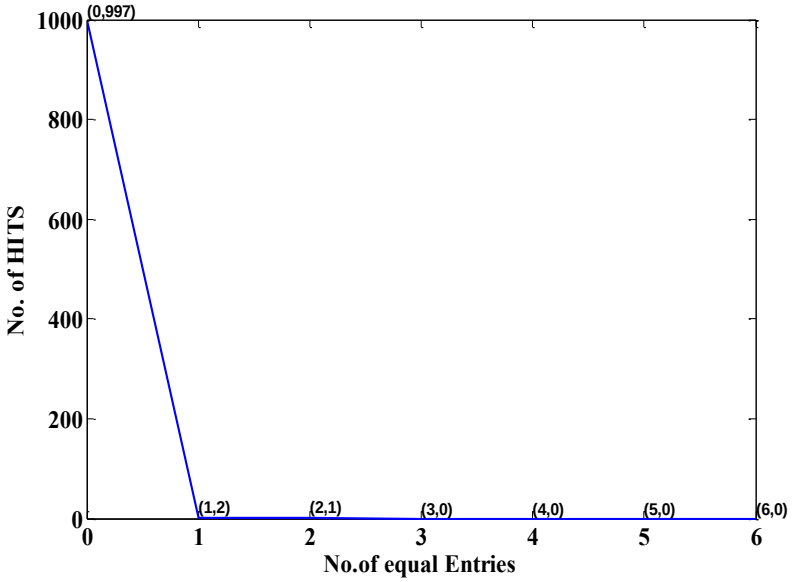

Fig 6: Positions where the ASCII characters are identical.

4.4 Security Analysis

Attacking strong collision resistance is analogous to the Birthday Paradox. The Birthday attack can be launched in time $O(n^{1/2})$ where $n = 2^w$. Hence, a 64 bit hash value (w=64) is extremely vulnerable. Partly in response to this concern, hashes of length 128 and above are widely used. Unfortunately, even these have

recently been shown to be vulnerable. For example, Wang et al. found hash collisions in the 160 bit SHA-1 in approximately 2^{63} operations rather than 2^{80}. For one of that demand stringent security, one of the proposed neuro- hash functions of length 512 can be recommended.

5 Comparison Analysis

Here, we compare our proposed method with some obtainable hash functions, particularly those based on neural networks. Table 4 and 5 present the numerical results for 128 bit and 512 –bit hash values of some known obtainable hash functions [18, 26, 31-33]. It can be concluded from these tables that the projected hashing method is very close to that of an ultimate cryptographic hash function.

Table 4: Absolute difference for the 128 bit hash values generated when N=2048

Hashing Method	Max	Min	Mean	Mean/character
Proposed Method	2453	832	1530.5	95.65
Ref. [31]	2221	696	1506	94.125
Ref.[26]	2230	731	1368	85.5
Ref. [18]	2156	658	1431.3	89.456
Ref. [33]	2224	573	1401.1	87.569
Ref. [32]	1952	605	1227.8	76.738

Table 5: Absolute difference for the values generated when n=N=512

Hashing scheme	Maximum	Minimum	Mean
Our Scheme	7421	4671	5830
Ref.[26]	7081	4658	5778

The performance will vary not only between algorithms, but also with the specific implementation and hardware used. Table 6 shows the number of rounds and operators required for different hash algorithm [34].

Table 6: Comparison of Hash functions

Algorithm and variant	Output size (bits)	Rounds	Operations
MD5	128	64	And, Xor, Rot,Add (mod 2^{32}), Or
SHA-1	160	80	And, Xor, Rot,Add (mod 2^{32}), Or
SHA-256	256	64	And, Xor, Rot, Add (mod 2^{32}), Or,Shr
SHA-512	512	80	And, Xor, Rot,Add (mod 2^{64}), Or,Shr
Proposed Hash	512	16	Mul, Add, And

6. Conclusion

In this paper, a hash function based on TPM neural network is proposed, which uses the general iterative structure of Hash function. In the TPM neural network the output feedback model is employed, its output not only depends on the input and parameters of the neural network, but also its status. This dependence is enhanced by iterating the NN, which is very useful to improve the efficiency of Hash function. The proposed hash function has been analyzed for its randomness and security using numerical simulations. . Results indicate that the hash function is simple, competent, realistic, and trustworthy and hence can be a good aspirant for data integrity. It holds high message sensitivity and good statistical properties. Furthermore, this algorithm provides the litheness to increase the hash length to any random length, which makes the system opposing to birthday attack for hashes greater than 512 bits.

References

[1] A. Menezes, P. Van Oorschot, S. Vanstone. (1996). *Handbook of applied cryptography*. CRC Press.

[2] [Harshvardhan Tiwari , Krishna Asawa (2012). A secure and efficient cryptographic hash function based on NewFORK-256. *Egyptian Informatics Journal* (2012) 13, (pp. 199–208).

[3] X. Wang, H. Yu. (2005). How to break MD5 and other hash functions. in: *Proceedings of Eurocrypt'05*. (pp. 19–35). Aarhus, Denmark

[4] Wang X, Yin YL, Yu H.(2005). Finding collisions in the full SHA-1. In: *Advances in cryptology-crypto 05 proceedings. Lecture notes in computer science*, 3494. (pp. 17–36). Springer-Verlag.

[5] Wang X, Feng D, Lai X, Yu H. (2004). Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD, *IACR Cryptology* . ePrint Archive, (pp. 199).

[6] A. Kanso , M. Ghebleh, (2013). A fast and efficient chaos-based keyed hash function",*Commun Nonlinear Sci Numer Simulat* 18 (pp. 109–123)

[7] Zhang J, Wang X, Zhang W.(2007). Chaotic keyed hash function based on feedforward-feedback nonlinear digital filter. *Phys Lett A; 362*. (pp. 439–448).

[8] M Bellare, R. Canetti, H Krawczyk,(1996). Keying hash function for message authentication, *LNCS, 1109*, ,.(Advances in Cryptology-CRYPTO ’96). (pp.1-15). Springer Verlag.

[9] Bellare, Mihir (June 2006). New Proofs for NMAC and HMAC: Security without Collision-Resistance. In Dwork, Cynthia. Advances in Cryptology – Crypto 2006 Proceedings. *Lecture Notes in Computer Science 4117*. Springer-Verlag.

[10] "RFC 6151 (March 2011).Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms. *Internet Engineering Task Force*. Retrieved 15 June 2015.

[11] Secure Hash Standard. Federal Information Processing Standards Publications (FIPS PUBS) 180-2,2002.

[12] Douglas R Stinson, .(1996) Cryptography Theory and Practice. *CRC Press*.

[13] D.A. Karras and V. Zorkadis.(2003). On neural network techniques in the secure management of communication systems through improving and quality assessing pseudorandom stream generators. *Neural Networks*, Vol. 16, No. 5-6, (pp. 899-905).

[14] C.-K. Chan and L.M. Cheng.(March 2001) The convergence properties of a clipped Hopfield network and its application in the design of key stream generator, *IEEE Transactions on Neural Network.* Vol. 12, No. 2(pp. 340-348).

[15] Shiguo Lian, Zhongxuan Liu, Zhen Ren, Haila Wang. (2006). Hash Function Based on Chaotic Neural Networks.(pp.237-240). IEEE.

[16] ShiguoLian,Jinsheng, Sun,ZhiquanWang(2006).One-wayHashFunction Based on Neural Network. *Journal of Information Assurance and Security*.(pp.1-7)

[17] S. Lian, J. Sun and Z. wang,. (2006). Secure hash function based on neural network. N*eurocomputing*, vol. 69. (pp. 2346-2350).

[18] D. xiao, X. Liao and Y. wang.(2009). Parallel keyed hash function construction based on chaotic neural network", *Neurocomputing*, vol. 72. (pp. 2288-2296).

[19] Zhongquan Huang.(2011). A more secure parallel keyed hash function based on chaotic neural network. *Commun Nonlinear Sci Numer Simulat*,vol.16.(pp. .3245-3256).

[20] Y. Li, S. Deng and D. Xiao.(2011). A novel hash algorithm construction based on chaotic neural network", *Neural Comput & Applic*, vol.20. (pp. 133-141).

[21] I.Kanter, W.Kinzel, E.Kanter(2002). Secure Exchange of Information by Synchronization of Neural Networks", *Europhys Lett 57*, (pp. 141-147).

[22] S .Santhanalakshmi,TSB Sudarshan,G K Patra (2012). Neural Synchronization by Mutual Learning Using Genetic Approach for Secure Key Generation. *Recent Trends In Computer Networks and Distributed System CCIS* Volume 335, (pp. 422-430). Springer-Verlag,

[23] S.Santhanalakshmi, Sangeeta K,G K Patra (2015). Analysis of Neural Synchronization using Genetic approach for Secure Key Generation. *Third International Symposium on Security in Computing and Communications (SSCC) CCIS* Volume356,.(pp. 207-216). Springer-Verlag

[24] S .Santhanalakshmi, Sangeeta K,G K Patra(2014).cSecure Key Stream Generation using Computational Intelligence Methods. IJCTA-Volume 5 Issue 3 (pp. 967-972).

[25] Harshavardhan Tiwari,Krishna Asawa (2012) A Secure and efficient cryptographic hash function based on New FORK-256. *Egyptian Informatics Journal* 13,.(pp. 199-208).

[26] A. Kanso , M. Ghebleh. (2013). A fast and efficient chaos-based keyed hash function. *Commun Nonlinear Sci Numer Simulat* 18 (pp. 109–123).

[27] Yong Wang, Xiaofeng Liao, Di Xiao, Kwok-wo Wong. (2008). One-way hash function construction basedon 2D coupled map lattices. *Information Sciences 178* (pp. 1391–1406)

[28] Zhang J, Wang X, Zhang W. (2007). Chaotic keyed hash function based on feedforward-feedback nonlinear digital filter. *Phys Lett A* 362. (pp. 439–448).

[29] Maokang Du,Bo HE, Yong Wang,Jianjun Wu, Di Xiao(2009). Parallel Hash function Based on Block cipher", *International conference on E-business and Information System Security* .

[30] D.Xiao,Xiaofeng Liao,Kwok-Wong. (2006). Improving the Security of dynamic Look-Up Table Based Chaotic Cryptosystem. *IEEE transactions on Circuits and SystemsI,* Vol 53 issue 6.

[31] Xiao D, Liao X, Deng S. (2005). One-way hash function construction based on the chaotic map with changeable-parameter. Chaos Solitons.Fract2005:24(386) pp. 65–71.

[32] Xiao D, Liao X, Deng S. (2008). Parallel keyed hash function construction based on chaotic maps. *Phys Lett A* 372. (pp. 4682–8).

[33] Xiao D, Shih F, Liao X. (2010). A chaos-based hash function with both modification detection and localization capabilities. *Commun Nonlinear Sci Numer Simul* 15(9) (pp. 2254–61).

[34] Comparison of Hash cryptographic hash function .Wikipedia