International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org, editorijettcs@gmail.com
Volume 2, Issue 1, January – February 2013 **ISSN 2278-6856**

A Survey on Recent Cryptographic Hash Function Designs

Richa Purohit (Arya) [1], Upendra Mishra[2] and Abhay Bansal[3]

[1,2] Amity School of Engineering. & Technology
Amity University Rajasthan, Jaipur, India

[3]Amity School of Engineering. & Technology
Amity University Uttar Pradesh, Noida, India

Abstract- ***Hash Functions are important tool in information security over the internet. The hash functions that are used in various security related applications are called cryptographic hash functions. They accept arbitrary length of input and produce a usually small, fixed size output, known as message digest. They are designed to provide message integrity, i.e. if the message has been changed after transmission from sender and before it may be received by the corresponding receiver, can be traced by the receiver, and thus, such a modified message can be discarded. This property is also useful in may other applications such as creation of digital signature and random number generation etc. Most of the hash functions are based on Merkle-Damgard construction, such as MD-2, MD-4, MD-5, SHA-1 etc, which are not hundred percent safe from attacks. The paper discusses some of the attacks that are possible on this construction, and thus on all of these hash functions also face same attacks. Today, the aim of researchers is to find new hash functions with either modification in Merkle-Damgard construction or with totally new designs. This paper discusses few modifications in the popular Merkle-Damgard construction, and few of recent hash function designs such as Whirlpool, TAV, JH Hash, RC-4 Hash, BLAKE-256 and CHA-, that are using the new modified design architecture.***

Keywords: BLAKE-256, CHA-1, JH Hash, Merkle-Damgard construction, RC-4 Hash, TAV, Whirlpool.

1. INTRODUCTION

Cryptographic Hash Functions are important building blocks in computer security. They provide message integrity that is a surety that the receiver is receiving the same message that was sent by sender, and has not been modified by adversary in the path. Typically hash functions are the mathematical functions that take arbitrary length input and produce a small output of fixed size. This output is known as message digest or hash code or hash result or simply hash. The hash value is digital fingerprint of the message or file, that is as two persons can not have same fingerprint and one person can have only one fingerprint, similarly no two different message can have same hash value and one message can have only one hash value associated with it. The hash output depends on each bit or each character of input, that's why a small change in the message will lead to definite change in the digest value. This unique property of hash makes the hash functions useful for different applications such as- digital signature, password protection, software integrity, pseudo-random number generation, commitment schemes such as message authentication etc. [1- 3].

Conventionally, all Hash Functions are designed to have following three properties: [4]

(a) Collision Resistance- It is computationally infeasible to find x, y , $x \neq y$ such that $H(x) = H(y)$.

(b) Pre-image Resistance- Given an output value y, it is computationally infeasible to find x such that $h(x) = y$.

(c) Second Pre-image Resistance- Given an input x', it is computationally infeasible to find x such that $H(x) = H(x')$.

There is a wide variety of cryptographic hash functions but MD5 and SHA-1 are being used in majority. Both hash functions are derived from MD4, which has long been known to be weak [5-6], thus leading to concerns that they might have common weaknesses. These issues got highlighted in late 2004, when techniques for efficiently finding collisions in MD5 [7] and SHA-0 [8] were announced. Subsequently, Wang [8] announced a technique for finding collisions in SHA-1 in 2^{69} operations, rather than the 2^{80} for which it was designed, and Lenstra et al. [10] demonstrated a pair of X.509 certificates with the same distinguished name, different public keys, and identical signatures, though no extension is known which can generate such a pair with different distinguished names. Now, as it is clear that MD-5 and SHA-1 were not as strong as their target security levels and so there was a requirement to shift towards new hash functions with improved designs.

Lucks and Daun have shown the attack on hash functions using following property-

$$H(x) = H(y) \Rightarrow H(x \parallel \sum) = H(y \parallel \sum)$$

where $\sum$ is an arbitrary string and x and y are messages of same length [11]. They proved that collisions are still possible in hash functions, no matter we have specified above mentioned three properties for them.

When first hash function was designed for cryptographic security purpose, it used iterative approach, in which the whole message is broken down into small similar sized blocks and each block is processed individually taking the CV from the output of previous block processing. This is

done for all blocks in circular fashion. Today also, almost all new designs include same iterative method. For iterative design we refer to Merkle - Damgard structure. But Merkle-Damagard construction also faces a class of generic attacks, that is, all hash functions based on this structure are also prone to this type of attack. There are following few such variants of such attacks [12]:-

Collision Attack- A collision is trivially found as- H(M‖pad ‖ x) = H(H(M) ‖ x), where, pad is padding appended to message M before hashing.

Second Collision Attack- If we have H(M) = H(N) while M ≠ N and |M| = |N| a second collision can be obtained by extending M and N with an arbitrary string S, H(M ‖ S) = H (N ‖ S).

Related Message Attack- Only by knowing length of message M that is, L and H(M) , one can easily compute a related or extended message M' for an unknown message M. H(M ‖ L ‖ x) is hash of extended message M by suffix L ‖ x. If attacker knows length of message L, he may trace out how M has been padded before being hashed.

MAC forgery Attack- In this, one may compute a valid message without knowing the secret key K, used by a MAC algorithm based on Merkle-Damgard that generates a tag T from M. He can update message M by appending a suffix Y and obtain a tag T' matching the new message M‖Y without even knowing K, that is, MAC(K,M‖Y)=H(K‖M‖Y).

(Length) Extension Attack- The length extension attacks are a type of attack on certain types of hashes which allow inclusion of extra information. This attack can be done on hashes with construction H(key ‖ message) when the message is known and the length of the secret is known.

Multi-collision Attack- A multicollision for a function is a set of inputs whose outputs are all identical. It generate a series of messages m1, m2, ... mN, such that hash(m1) = hash(m2) = ... = hash(mN). It is defined as 2t messages that all have the same hash value – costs only about t times a single collision attack [13].

2. FEW MODIFICATIONS IN MERKLE-DAMGARD CONSTRUCTION

In an attempt to overcome the attacks, described in previous section (and many more), few new hash functions are being designed with modifications in Merkle-Damgard structure. The paper discusses few modified structures in this section of the paper.

2.1 Wide and Double Pipe- Lucks has shown that if the size of the chaining variable of hash algorithm is increased, then the size of final hash value will also be increased and this will lead to more secure hash function [14]. It will solve the problem of extension attack, as it will now become difficult to guess unknown discarded bits to be able to append an extension, because in this method, the final hash value is truncated. Moreover, this method will also make it difficult to find collisions for compression function. But apart from these advantages over Merkle-Damgard construction, this method also has drawback of lesser efficiency due to larger input-output for compression function.

2.2 3C Construction- This is another modification over Merkle-Damgard construction. In this, a variable is maintained, that contains a value produced by repeatedly XORing the chaining variable while hashing the message (refer to Figure-1). The variable is then processed in an extra finalization call to compression function [15]. This method is also prone to multi collision attack, but developers have shown that it is safe from long messages second pre-image attack and herding attack. But, later in 2008, Praveen Gauravaram and John Kesley [16] shown that 3C construction is also not safe from second pre-image attack and herding attack .

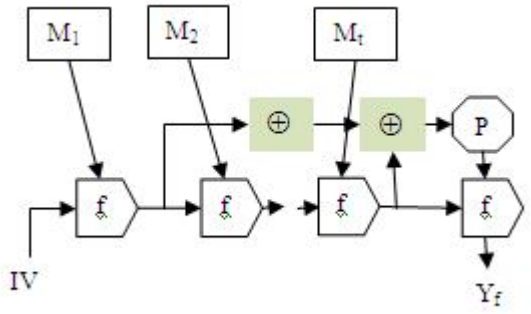

Figure 4: The 3C Construction

Algorithm works as follows:

$M \rightarrow M_1 \ldots M_\ell$, $y_0 = IV$, $t=0$
For ($i=1$ to ℓ do)
$Y_i = f(M_i, y_{i-1})$
$t = t \oplus yi\text{-}1$
return $Y_f = f(P(t), y_\ell)$

2.3 Prefix, Chop, NMAC and HMAC Constructions- Prefix algorithm ensures that message is padding free. It may be done by prepending or appending length of whole message to each block. But this padding becomes a reason for poor efficiency and it also wastes bits while representing length in each block.

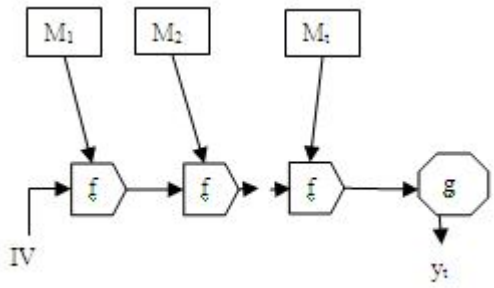

Figure 5: NMAC Method of hashing

The another construction is Chop, in which few non-

trivial bits are removed from final digest. This technique is not suitable as it also weakens the security of hash function. One more technique is NMAC, in which output of last block processing is provided as input to an independent function g as figure2:

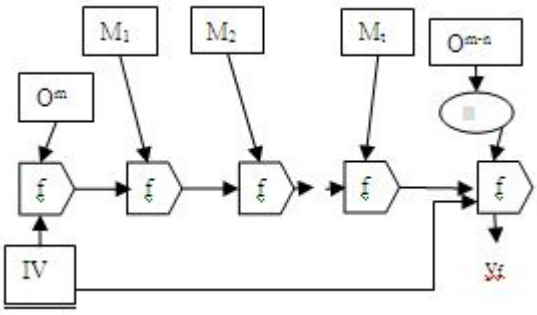

Figure 6: NMAC method of hashing

HMAC uses extra compression function like nested MACs, as shown in figure 3.

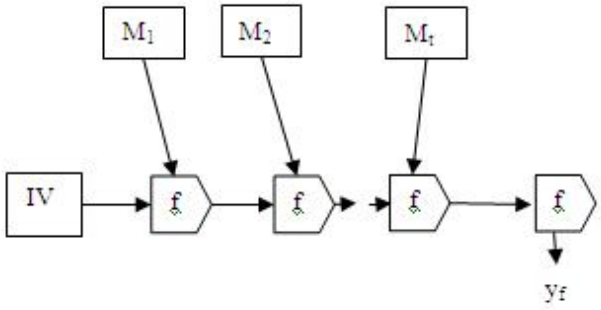

Figure 7: Merkle-Damgard with Permutation

Apart from these modifications there are lot more modifications in Merkle-Damgard construction such as- use of permutation [17] in chaining variable input of last compression function (as figure 4), use of salt for randomizing hash function [18] (designers proved that the salt and randomization can turn even the weak hash algorithm into a stronger one), nested iteration mode of operation [19], which makes use of two keys k_1, $k_2 \in \{0,1\}^k$ etc.

Now, we will discuss few popular recent cryptographic hash techniques in the following section.

3. FEW RECENT HASH TECHNIQUES

Beginning from Merkle-Damgard construction, research is moving ahead towards refinements and modifications in this construction. This section of paper describes few such recently developed hash function that are still based on logic given by Merkle-Damgard.

3.1 Whirlpool Secure Hash Function-

Whirlpool is block cipher based secure hash algorithm [20] proposed by Vincent Rijmen and Paulo Barreto endorsed by NESSIE (New European Schemes for Signatures, Integrity and Encryption). Although it is block cipher based technique, where block ciphers have own disadvantages for this purpose, such as- they show certain weaknesses, they are slower than compression function based hash functions etc., yet it provides security and performance almost as good as hash functions based on non-block ciphers. Here, output is 512 bit and makes use of block cipher W which is similar to AES in some aspects to provide better performance [21]. Because output length is more than that of SHA-1, it promises a stronger result.

The Whirlpool hash function is given as-

H_0= initial value

$H_i = E(H_{i-1}, M_i) \oplus H_{i-1} \oplus M_i$

H_i = hash code value

The block cipher used in Whirlpool is known as W, and is based on AES, but W takes a block and a key of 512 bits instead of 128 bits. It operates on a state of 4*16 bytes of Rijandeal, but due to more number of rounds, it becomes slow in speed.

Whirlpool exhibits good performance in terms of execution speed in both hardware and software and can work with lesser memory requirements. Moreover, the hash function W is also resistant against linear and differential cryptanalysis because of its substitute byte function. Here, S-Box consists of two non-linear layers, each containing two 4*4 S-boxes separated by a 4*4 randomly generated box. SB function introduces nonlinearity into algorithm, thus, it should exhibit no correlations of output bits and small changes in input results in large change in output.

3.2 JH Hash Function-

JH Hash Function was designed by Hongjun Wu to be submitted to NIST hash competition. It processes message blocks of 512 bits and generates hash of 224, 256, 384 and 512 bits. It is also based on Merkle-Damgard mode of hash function. Its compression function can be given as-

$f(H_{i-1}, M_i) = E(H_{i-1} \oplus M_i \| 0^{512}) \oplus 0^{512} \| Mi +$

Where, E is a permutation of 1024 bits and 0^{512} means the string of 512 zero bits. For $c \leq n/2$, JH Hash Function using an ideal n bit permutation and producing c-bit outputs by truncation is collision resistant up to $O(2^{c/2})$. This bound implies that JH function provides the optimal collision resistance in the random permutation model [22]. JH Hash functions are very efficient in S/W. With bit slice implementation using SSE2, the speed of JH is about 16.8 cycles/byte on Intel Core2 Duo microprocessor, running 64-bit Operating System, with Intel C++ compiler [23]. It is also based on a generalized AES design methodology. It has a 1024 bit state and works on an input block of 512 bits. It performs in three steps- First of all, the left half of the state is XORed with input block. In second step, a bijunctive function E8 is applied to the state. As last part, the right half of the state is XORed with the input block.

The simple JH compression function structure reduces the cost of security evaluation with respect to differential cryptanalysis. Because of enough confusion and diffusion

after message modification, it is secure against differential attacks. It is also resistant to second pre-image attack because we use 1024 bit hash value for JH-512 while the reversible property of compression function is being taken into consideration.

3.3 TAV Hash Function-

It is 128 bit light weight hash function designed by Peris-Lopez et. al. for an RFID authentication protocol. The mappings of f and g are defined by-

$f: \{0,1\}^{160} \times \{0,1\}^{32} \dashrightarrow \{0,1\}^{160}$ and

$g: \{0,1\}^{160} \dashrightarrow \{0,1\}^{128}$ respectively.

(RFID (Radio Frequency Identification) systems aim to identify tags in an open environment where neither visual nor physical contact is needed for communication. RFID tags are also useful in authentication applications such as in passport, access control cards for public transportation and location tracking system etc.)

Many RFID protocols make use of cryptographic hash functions for implementing security. TAV-128 is one such function. Designers of this function performed only few statistical tests for randomness, and thus it has not gone through security analysis. And they proved that TAV_128 does not posses any trivial weaknesses. But later after carrying third party analysis, it got proved that this hash function is neither collision resistant nor second pre-image resistant [24].

TAV produces a 128 bit digest and is also based on Merkle-Damgard Construction. Its compression function f works on words of 32 bits. TAV should take 2^{64} compression function calls to find a collision and number of compression function calls is 2^{128} to mount a second pre-image attack, so that it may be considered as a collision resistant hash function. However, its internal state is of 160 bits but output is of 128 bits. Moreover in order to avoid the attacker to have direct access to any bit of internal state, two more functions have been included as a filter phase, which actually lead to improve the security.

3.4 RC4 Hash Function-

RC4 was designed by Donghoon Chang, Kishan Chand Gupta and Mridul Nandi in 2006. It is based on RC4 stream cipher, given by Ron Rivest in 1987. RC4 is most widely used software stream cipher and is used in popular protocols such as SSL and WEP. Although it is simple and executes with fast speed yet, in new systems it is not that worthy to use, especially when nonrandom or related keys are used [25]. RC4 uses a state change function which is invertible. Invertibility leads to long cycle lengths, on an average of about $2^{m/2}$ where m is the number of bits in initial state, and this long cycle length is one of the reasons of security of RC4. The whole processing of RC4 is so simple that it automatically reduces possibility of errors. It makes use of lesser number of operations and so is faster in execution. But RC4 faces many attacks, such as shortcut attack, related and weak key attacks, Knudsen's attack etc [26]. One more attack is IV weakness; to avoid which, at least N words of output should be dropped from the beginning of each stream. RC4 hash is denoted by RCH_ℓ , $16 \le l \le 64$ where, RCH_ℓ: $\{0,1\}^{<2^{64}} \rightarrow \{0,1\}^{8\ell}$. message size is at most 2^{64}-1. It is a wide pipe hash function that uses an initial value and a variant of padding rule which provides a dynamic hash function.

3.5 CHA-1-

This hash function was proposed by Mohmoud Maqableh, Azman Bin Samsudin and Mohammad A. alia in 2008 based on chaos theory [27]. CHA-1 takes an input with less than 2^{30} bits and produces a digest of 160 bits. It uses a logistic map for implementation, which is simple, fast , sensitive to initial conditions, unpredictable and a one way function. Logistic map is a recursive function which takes a real number (X_n), $0 \le X_n \le 1$ as an input and produces a real number $X_{n+1,}$ between 0 and 1 as indicated by-

$$X_n = r\,(1- X_{n-1})\,X_{n-1}\,;$$

Where $r \in [0,4]$, $X_n \in (0,1)$, and $n \in N$.

This logistic map is used in two phases of CHA-1 and due to this the final output of CHA-1 is highly influenced by initial conditions and original message.

CHA-1 consists of four parts- (i) X_0 function- it produces 160 bit output which is further given as input to logistic map in phase 3. (ii) R function- it produces an incremental value r which is also used as argument for logistic map. (iii) Next function finds the chaos region of logistic map by finding a real value between 0.33 and 0.59. (iv) In last phase, X_n is produced recursively by using a 64 bit value at a time from original message.

The designers compared the CHA-1 with SHA-1 in terms of execution time for same input message, which proved that SHA-1 is faster than CHA-1 but they claimed that CHA-1 is more secure than SHA-1. They mentioned that with advances in cryptanalysis, the security factor for SHA-1 has gone down to only 2^{63} (hash-operation-brute force) while the security factor for CHA-1 is still 2^{80} (hash-operation-brute force). Further CHA-1 can accept any message less than 2^{80} bits instead of 2^{64} bits as in case of SHA-1 [27]. They also depicted that randomness test average and standard deviation of CHA-1 is also better than that of SHA-1.

3.6 BLAKE-256-

It was developed by Jean-Philippe Aumasson, Luca Henzen, Willi Meier and Raphadel Phan in 2008 to be submitted as a competitor in NIST SHA-3 competition [28].

The core BLAKE-256 compression function takes, as an input, 512 bits/16 words/64 bytes of message data, 256 bits/8 words/32 bytes of chaining value, 128 bits/4 words/ 16 bytes of salt, and additionally a counter that is 64 bits/2 words/8 bytes. A series of XORs, rotations and modular additions are used for generating new chaining values. Its compression function takes 512 bit input and 128 bit salt to produce 128 bit output by applying an invertible nonlinear transformation composed of 14 rounds, and each round uses a non-linear permutation G. here G accepts four of 32 bit words, two message words and two constant words.

It leads the simplicity of algorithm and performs fastly on software and hardware. As it is based on an intensively analyzed component (ChaCha) and proved to be resistant to generic second pre-image attacks, side channel attacks and length extension. But it can work for message less than 2^{64} bit (BLAKE can work up to 2^{128} bit message) [27].

4. CONCLUSION

Cryptographic hash functions are important tool in the area of network security, and since beginning MD-2, MD-4, MD5, SHA-1 etc are very popular hash functions. But, after finding collisions in popular hash functions as MD-5 and SHA-1, focus got shifted towards designing new secure and faster hashes functions. TAV-128, BLAKE-256, RC-4, Whirlpool, CHA-1, JH Hash etc. are products of such new generation of hash functions. Although they used to follow, more or less, the same Merkle-Damgard construction, but each of design has modified this construction for better security and improved performance results, for example JH hash function has included bit slicing, TAV-128 has included a filter phase in Merkle-Damgard design, RC-4 includes long cycle length and so on.. The designers are also working on modification of these new designs for better performance, so that, the attacks which are possible as of now, may not perform in coming times and we may get full-proof hash functions.

BIBLIOGRAPHY

[1] H. Tiwari, K, Asawa. Cryptographic Hash Function: An Elevated View. European Journal of Scientific Research, 2010, vol.43 issue-4, pp.452-465.

[2] S.Bakhtiari, R. Safavi-Naini and J. Pieprzyk. Cryptographic hash functions: A Survey. Technical Report 95-09, Department of Computer Science, University of Wollongong. 1995.

[3] J. Walker, M. Kounavis, S. Gueron and G. Graunke. Recent Contribution to Cryptographic Hash Functions. Intel Technology Journal, vol-13, issue-2, 2009, pp- 80-95.

[4] S.M. Bellovin, E.K. Rescorla. Deploying a New Hash Function. Presented at first NIST Workshop, 2005. Available at http://www.csrc.nist.gov/pki/HashWorkshop /2005/Oct31_Presentations/Bellovin.new-hash.pdf.

[5] H. Dobbertin. Cryptanalysis of MD4 (Third Workshop on Cryptographic Algoruthms, Cambridge 1996). Lecture Notes in Computer Science, pages 55–72, 1996.

[6] H. Dobbertin. The First Two Rounds of MD4 are Not One-Way. Lecture Notes in Computer Science, 1372, 1998.

[7] X. Wang and H. Yu. How to Break MD5 and Other Hash Functions. In Proceedings of Eurocrypt ’05, 2005.

[8]E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, and W. Jalby. Collisions of SHA-0 and Reduced SHA-1. In Proceedings of Eurocrypt ’05, 2005.

[9] X. Wang, Y. Yin, and H. Yu. Collision Search Attacks on SHA1, 2005.
Available at: http://theory.csail.mit.edu/˜yiqun/shanote.pdf.

[10] A. Lenstra, X. Wang, and B. de Weger. Colliding X.509 Certificates. In Proceedings of ACISP, 2005. To appear. Online: http://eprint.iacr.org/2005/067.

[11] http://th.informatic.uni-mannheim.de/peopl/lucks/ HashCollisions.

[12] S.Al Kuwari, J.H. Davenport, R.J. Bradford. Cryptographic Hash Functions: Recent Design Trends and Security Notations. In short paper proceedings of Inscrypt’10. Science Press of China, 2010, pp- 133-150.

[13] A. Joux. Multicollisions in Iterated Hash Functions: Application to Cascaded Constructions. CRYPTO (Matthew K. Franklin, ed.), Lecture Notes in Computer Science, vol. 3152, Springer, 2004, pp. 306–316.

[14] S. Lucks. A Failure-Friendly Design Principle for Hash Function. In Asiacrypt’05, vol 3788 of LNCS, pp- 474-494. Springer-Verlag, 2005.

[15] P. Gauravaram, W. Millan, Ed. Dawson, K. Viswanathan. Constructing Secure Hash Functions by Enhancing Merkle-Damgard Construction. In CISP’08, vol. 4058 of LNCS pp- 407-420. Springer-Verlag, 2006.

[16] P. Gauravaram, J. Kesley. Linear-XOR and additive checksums don’t Protect Damaged Merkle Hashes from Generic Attacks. In CT-RSA ’08, vol 4964 of LNCS, pp- 36-51, Springer-Verlag, 2008.

[17] S. Hirose, Je H. Park, A. Yun. A Simple Variant of Merkle Damgard Scheme with Permutation. In Asiacrypt ’08, vol 4833 of LNCS, pp- 113-129, Springer-Verlag, 2008.

[18] S. Halevi and H. Krawczyk. The RMX Transform and Digital Signatures. In second NIST Hash Workshop, 2006.

[19] M. Bellare and T. Ristenpart. Hash Functions in the Dedicated Key-Setting; Design Choices and MPP Transforms. In ICALP ’07, vol 4596 of LNCS, pp-399-410. Springer-Verlag, 2007.

[20] W. Stallings. The Whirlpool Secure Hash Function. Cryptologia, vol- 30, pp- 55-67, 2006.

[21] http:// paginas.terra.com.br/informatica/paulobarreto/ whirlpoolpage.html

[22] R. Bhattacharya, A. Mandal and M. Nandi. Security Analysis of the Mode of JH Hash Function. In Proceedings of FSE, LNCS 6147, pp- 168-191, Springer 2010.

[23] H. Wu. The JH Hash Function. January 2011. Available at: http://www3.ntu.edu.sg/home/wuhj/research/jh/

[24] A. Kumar, S.K. Sanadhya, P. Gauravaram, M. Safkhani and M. Naderi. Lecture Notes in Computer Science, vol 6498. In Proceedings of Indocrypt 2010: 11th International Conference on Cryptology in India, pp- 118-130.

[25] M.E. McKague. Design and Analysis of RC4-like Stream Ciphers. Ph.D. Thesis presented to University of Waterloo, 2005.

[26] D. Chang, K.C. Gupta and M. Nandi. RC4- Hash, A New Hash Function based on RC4. In Proceedings- INDOCRYPT '06, Proceedings of 7th International Conference of Cryptology in India, 2006, pp- 80-94.

[27] M. Maqableh, A.B. Samsudin amd Mohd. A. alia. New Hash Function Based on Chaos Theory (CHA-1). International Journal of Computer Science and Network Security, vol- 8 (2), 2008, pp- 20-26.

[28] J.P. Aumasson, L. Henzen, W. Meier and R.C. W. Phan. BLAKE Cryptographic Hash Function, SHA-3 Proposal Finalist 2010. Available at: http://131002.net/blake/