Cellular Automata Based Hashing Algorithm (CABHA) for Strong Cryptographic Hash Function

Kartik Rajeshwaran,
School of Computer Science and Engineering
Vellore Institute of Technology, Vellore, India
kartikrajeshwaran.kr@gmail.com

Dr. Kakelli Anil Kumar
Associate Professor
School of Computer Science and Engineering
Vellore Institute of Technology, Vellore, India
anilkumar.k@vit.ac.in

Abstract— Cryptographic hash functions play a crucial role in information security. Cryptographic hash functions are used in various cryptographic applications to verify the message authenticity and integrity. In this paper we propose a Cellular Automata Based Hashing Algorithm (CABHA) for generating strong cryptographic hash function. The proposed CABHA algorithm uses the cellular automata rules and a custom transformation function to create a strong hash from an input message and a key.

***Index Terms* — Cryptographic hash function, Cellular Automata, Digital Signature, One-way Function, Permutation**

I. Introduction

Cryptographic hash functions are an important part of modern cryptography. The hash functions took an important role in computer security with the development of MD5 algorithm having digest size of 128bits [1]. Although now considered an insecure hashing algorithm, SHA1 laid the foundation of further SHA based algorithms. SHA2 and SHA3 are considered extremely secure and are widely used in SSLs. Cellular Automata [2, 3] are relatively old concept and there have been numerous researches on its applications in cryptography from encryption algorithms to hash algorithms. M. R. Z. Baraa Tareq Hammadet al presented a hashing algorithm used in cellular automata (Rule 30 and Rule 134) along with a custom Omega-Flip Network [4]. This explored the potential of the use of hybrid Cellular Automata rules along with Omega-Flip networks which could help achieve a strong avalanche effect among the bits. Further in 2016, Parashar et al [5] proposed a Cellular Automata based hashing algorithm which worked serially working on blocks of bits along with bit rotation to develop a strong hash. The proposed algorithm in this paper uses the concepts of cellular automata along with bit shuffling and logical operations to form an extremely random hash which would make it computationally infeasible to get back to the original input, as well as byte collision resistance for highly secure hashing algorithm.

II. Mathematical Background

A. Cryptographic Hash Function

Cryptographic hash functions have a crucial role to play in the modern cryptography. A cryptographic hash function is a type of one-way function that takes an arbitrary input of any no. of bits and outputs a fixed n number of bits. This output of one-way function must possess properties like, it must be computationally infeasible to get back the input, and it must be computationally infeasible to find a pair of messages which has same hash value, also known as collision resistance property of a hash function. Also, for two messages with minimal difference, hash function must give a completely different, unrelatable output. A hash function is considered good if it possesses these three properties.

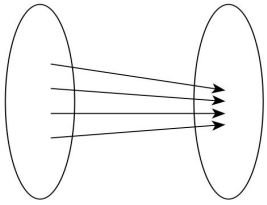


Figure 1: A basic representation of hash function showing its many to one mapping property from domain to co-domain.

Figure 1 shows the many-one property of the hash function, i.e. an input of arbitrary size will output a fixed size hash. The Cryptographic hash functions at their highest level can be classified into two broad categories namely – whose output is dictated by a single input parameter i.e. is a message known as unkeyed hash functions, then there are those hash functions whose outputs are dictated by two input parameters i.e. a message and a corresponding secret key also known as keyed hash functions [6] as shown in *Figure 2*. Modern keyed cryptographic Hash functions also have the properties of diffusion which essentially means the hiding of statistical properties of the input message and confusion which means that the final hash has dependency on the input key [7, 9, 10].

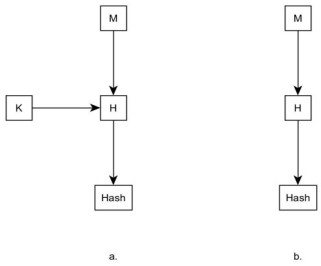


Figure 2: a) A keyed hash function b) A unkeyed hash function

B. *Elementary Cellular automata*

Cellular automata are a discrete model that is extensively studied in mathematics, computer science, physics etc. At its foundation, cellular automata consist of a grid of finite number of cells. In case of Elementary cellular automata, the grids of cells are of a single dimension as shown in *Figure 3*. In a cellular automaton, there is a set of cells called neighborhood, for each cell, which is defined relative to the specified cell. An initial state is a state where states are assigned to each cell in the automata at say a time t. Now, a new generation of cells are created when 't' is incremented by 1, i.e. a new generation for t=t+1 as shown *Figure 6.*

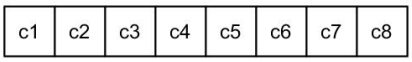


Figure 3: A basic representation of an elementary cellular automata. Here c_i represents a cell.

The formation of new state depends on the Rule which is being applied to the given automaton state. The Rules are derived from the neighborhood of the cell. In this paper 3-neighborhood rules for the cellular automata are used. These rules have been defined in A New Kind of Science [8]. The rules which are going to be used in this paper are given below.

Table 1: Here C_s is the center cell, C_{s-1} is left neighbor and C_{s+1} is right neighbor

Rule No.	Logical Expression
Rule 30	C_{s-1} XOR (C_s OR C_{s+1})
Rule 134	(C_{s-1} AND (C_s OR C_{s+1})) XOR C_s XOR C_{s+1}

Rule 30: Rule 30 is one of the rules introduced in A New Kind of Science. This Rule is known to possess aperiodic chaotic behavior. The rule can be represented by its neighborhood as following.

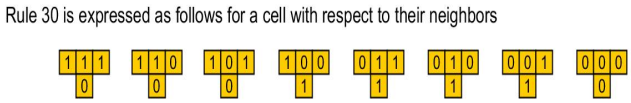


Figure 4: Rule 30

The Rule 30 produces a complex and seemingly-random pattern. This property of chaotic behavior is extremely useful when used in cryptography.

Rule 134: Rule 134 is one of the rules in A New Kind of Science. This rule in combination with Rule30 is proven to provide good avalanche effect and hence is considered to be suitable for the use in the proposed cryptographic hashing algorithm [8]. Rule 134 is represented as follows.

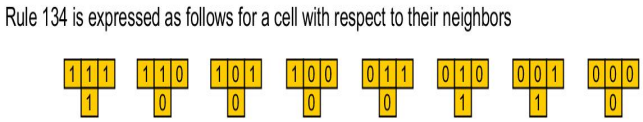


Figure 5: Rule 134

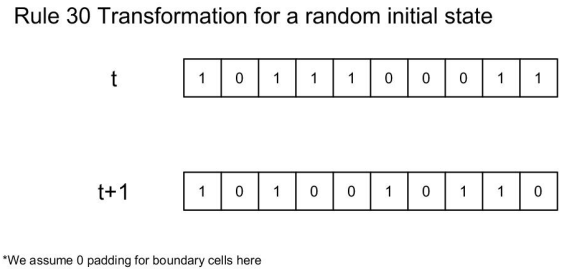


Figure 6: Representing Rule 30 Applied on the elementary cellular automata with an initial state at a time t.

III. PROPOSED HASHING ALGORITHM

The proposed algorithm takes a message 'm' of an arbitrary size, along with the key, of any standard size of 128, 192, 256 bits etc. and gives a key hashed value. The high-level structure of the hashing algorithm is as described below

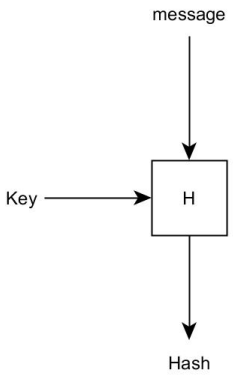


Figure 7: A high Level diagram for the algorithm to be proposed

A. *Internal Operations*

1. The input message must first be padded in such a way that

   size(m) modulo 512 = 0
   and
   size(m) >=512

   where size(m) is the size of the message 'm' and 512 is the number of bits.
   Let the Padded message be represented by M. Its recommended to use repetitive padding, for better security.

2. Key must also be padded in such a way that

size(k) modulo 512 = 0
and
size(k) >=512

where size(k) is the size of the key 'k' and 512 is the number of bits.
Let the Padded message be represented by K. Its recommended to use repetitive padding, for better security.

3. Now the message is then divided into the blocks of 512 bits as shown below.

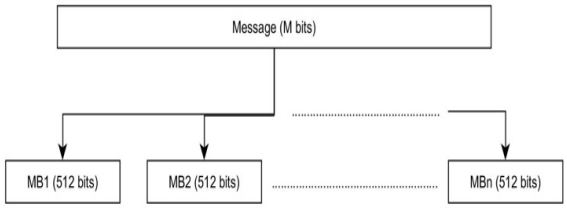


Figure 8: Padded message being divided into Message Blocks of 512 bits.

4. Now each Message Block of 512 bits is subdivided further into 8 Sub-Blocks of 64 bits each.

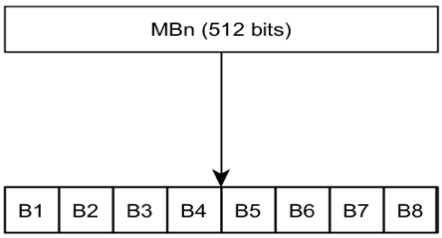


Figure 9: Each Message Blocks of 512 bits are divided into 8 Sub-Blocks of 64 bits each.

5. Cellular automata Rule 30 is applied to each Message Block (512-bits block) before passing into the next step as shown in *Figure 10*.

6. The output of Step 5 along with the 512 bits key is passed into the TF (Transformation Function).

7. This output is then XORed with second array of 512 bits after Rule 30 has been applied onto it.

8. The Key for subsequent rounds is derived via Key Rotator in the previous iteration which performs the circular shift of bits by 1 as shown in *Figure 10.*

9. Step 6 ,7 and 8 are repeated until there are no more Message Blocks of 512 bits left i.e. the whole message has been hashed.

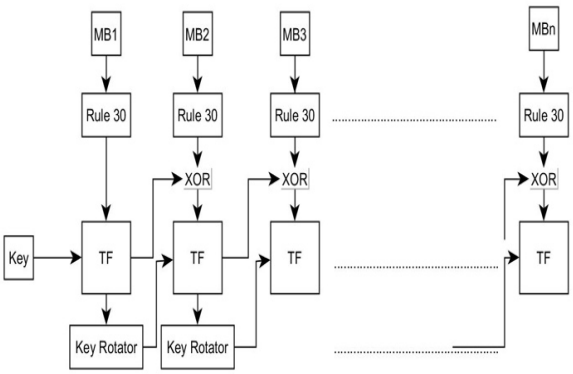


Figure 10: The diagrammatic representation of overall operation taking place in the algorithm.

B. Transformation function

Now further describing the operations taking place in the Transformation Function (TF). Here certain Cellular Automata rules are used along with some logical operations in order to generate a complete random output from the given input. Below given are the mapping rules for a single round of operation in Transformation Function for an input Message Block. Each Mapping takes place for a Sub-Block (64-bit each) using custom functions as shown in *Figure 11*

1. a=e
   This means, bytes from 'e' are directly mapped onto 'a'.

2. b= J(g, h, K_1) or a=J(g, h, K_5)
   a=J(g, h, K_1) is used when the round number is odd and a=J(g, h, K_5) is used when the round number is even.
   The function J(x, y, z) is described as following

   $$J(x, y, z) = ((ROTN^{47}(x) \text{ XOR } Rule30(ROTN^{37}(y')) \text{ AND } (ROTR^{17}(z)))' \quad \ldots. (1)$$

3. c=G(e, f, K_2) or c=G(e, f, K_6)
   c=G(e, f, K_2) is used when the round number is odd and c=G(e, f, K_6) is used when the round number is even.
   The function G(x, y, z) is described as following: -

   $$G(x, y, z) = (Rule134(Rule30(x')) \text{ OR } y) \text{ XOR } (Rule30(z') \text{ AND } x) \quad \ldots. (2)$$

4. d=F(a, c)
   The function F(x, y) is described as following: -

   $$F(x, y) = Rule30(x) \text{ XOR } y' \quad \ldots. (3)$$

5. e=J(a, d ,K_4) or e=J(a, d ,K_8)

e=J(a, d ,K_4) is used when the round number is odd and e=J(a, d ,K_8) is used when the round number is even. The function J(x, y, z) is defined in (1).

6. f=H(b, d)
   The function H(x, y) is described as following: -

$$H(x, y) = ROTN^{17}(x) \text{ XOR } ROTN^{59}(y) \quad .... (4)$$

7. g=I(c, f)
   The function I(x, y) is described as following: -

$$ROTN^{41}(x') \text{ XOR } Rule134(Rule30(ROTN^{31}(y'))) \quad .... (5)$$

8. h=H(a, K_3) or h=H(a, K_7)
   h=H(a, K_3) is used when the round number is odd and h=H(c, K_7) is used when the round number is even. The Function H(x, y) is described (4).

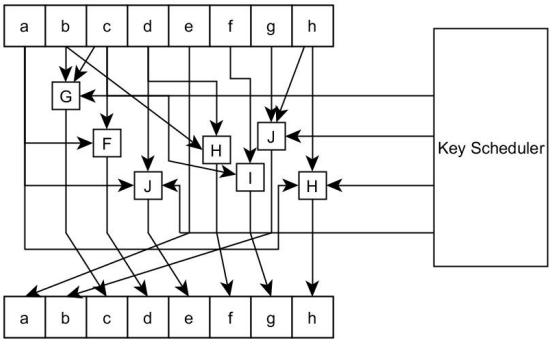


Figure 11: The diagrammatic representation of mapping inside a single round of Transformation Function

Inside this Transformation function the number of rounds are dynamically allocated. They are calculated by the formula:

rounds = mod(count of '1's in the message block (512 bits) + count of '0's in key (512 bits),512)

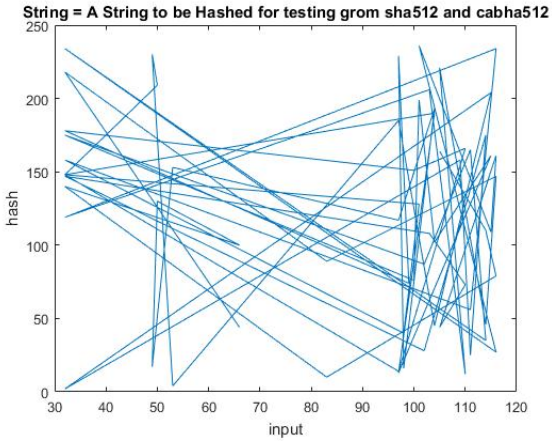
This is bound to change for new blocks and creating more randomness in the output, hence more confusion and more diffusion.

IV. Security Analysis

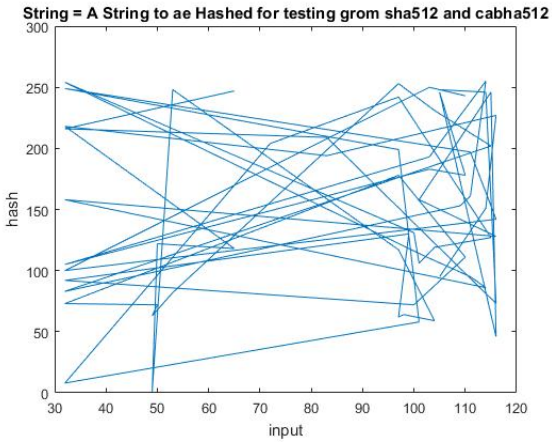
For a hash function to be secure, as discussed earlier, it must have strong collision resistance, for closely relatable inputs hash function must produce completely new random outputs and hash outputs must completely hide the properties of the input. These properties are essentially satisfied by the proposed algorithm. Since its tested that Rule 30 along with Rule 134 provides basic randomness suitable for hash functions[1], it has been incorporated into the proposed algorithm. Further, there are proposed custom functions for custom mapping in a single round of the Transformation Function. This is to make sure that a great avalanche effect takes place so as to introduce diffusion in the output.

A. Analysis for same key but closely related messages.

Figure 12 illustrates an essential feature of hash function which is the avalanche effect of bits. We have converted the input characters into their ASCII representation and plotted it with the hash function's output. X-axis represents the input of message that has been converted to their ASCII representation and Y-axis is the hash output. We can later obtain the hexadecimal representation of the hash output.



a.



b.

Figure 12a, b: The following shows a demonstration of randomized output generated from two tested string with only difference of a single character.

$String_1$ = "A String to be Hashed for testing grom sha512 and cabha512"
$String_2$ = "A String to ae Hashed for testing grom sha512 and cabha512."

Notice that, String1 and String 2 differ by a character in the fourth word as 'be' and 'ae' from String1 and String2 which means they differ from each other by a byte, if a standard encoding scheme is used. When $String_1$ and $String_2$ were hashed with the proposed algorithm, completely different hashes were obtained as evident from the *Figure 12*. This is

one of the fundamental features of the hash function. This feature illustrates the strength of the proposed hashing algorithm and its capability to avoid collisions as well as hide the details of input message.

B. Analysis for same Message but closely related keys.

Figure 13 illustrates the hash values plotted, for the same string but with different keys with only one character different, completely different and random output was obtained. Since there is an added advantage of the usage of the key, it can be utilized to produce strong hashes even with closely related keys. We have plotted the input message with its ASCII representation numbers and the hash output to study the randomness. X-axis represents the input message's ASCII representation and Y-axis represents the hash output.

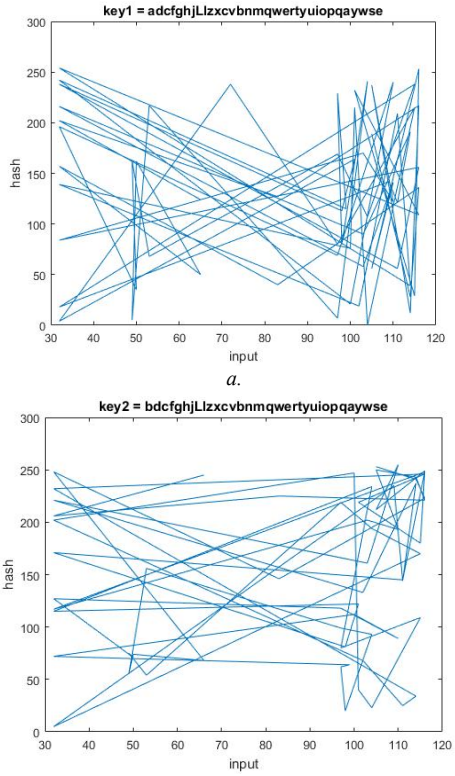


Figure 13: The following shows a demonstration of randomized output generated from two tested keys with only difference of a single character

String = "*A String to be Hashed for testing grom sha512 and cabha512"*

Key1 = "*adcfghjLIzxcvbnmqwertyuiopqaywse*"
Key2 = "*bdcfghjLIzxcvbnmqwertyuiopqaywse*"

The Message, Key pair was hashed, with message being same but keys having a byte of different from each other, still completely random hash was obtained. This shows that the key plays a significant role in the algorithm while hashing a sequence of bits. This illustrates the property of confusion in the given the algorithm, i.e. there is dependency on several parts of key in the final hash. These properties together can be exploited and we can use different keys for different messages to produce hash, making it extremely secure to use and infeasible to break or manipulate.

C. Analysis of Byte Collision Occurances

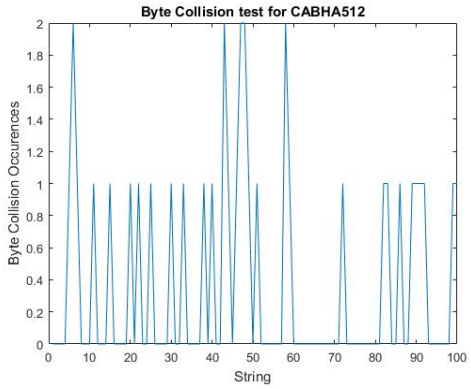


Figure 14: The CABHA512 algorithm test result for 100 random message-key pair

Hundred random message, key pairs were hashed, and byte collisions were observed, i.e. the occurrences of same byte at same position in two hash values. The above result clearly shows that such occurrences never occur more than 2. This means in the 100 strings tested, two random message-key pair had at most 2 bytes in the same positions. This result is considered good for a hash function. But this result on its own cannot prove the strength of the hashing algorithm and hence, probability of byte occurrence needs to be analyzed.

D. Probabilistic analysis of Byte Collision Occurrences

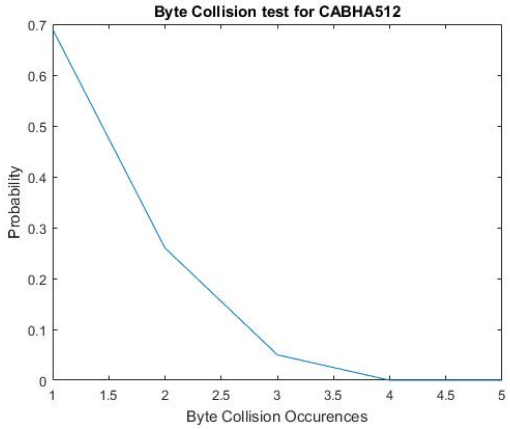


Figure 15: Probability for Byte Collision occurrence in hash of same 100 random message-key pairs

Figure 15 illustrates the probability of the Byte Collisions which was explained previously. It can be observed from the graph that probability of no byte collision occurrence is around 0.7, and as higher number of Byte Collisions are

observed, a decrement in Probability of byte collision occurrences and almost gets to 0. All the tested strings were of strictly 256 bits with 256 bits keys and hence the probability of a collision occurrence was observed to be 0.0425 which is extremely low for a hash function making this function strong. This essentially shows that the probability of complete collision for two strings is hard to find satisfying the fundamental requirement of the hash function.

Our proposed CABHA uses Transformation Function which produces much stronger effect compared to Omega-Flip Network proposed by M. R. Z. Baraa Tareq Hammad et al [4], since Omega-Flip simply relies on bit shuffling and doesn't do any transformation of bits unlike Transformation Function where bits are shuffled as well as transformed. CABHA has given better results in terms of low probability for collision. Since in the proposed CABHA algorithm we have a method of dynamic allocation of rounds unlike approach used by M. R. Z. Baraa Tareq Hammad et al [4], the hash is even stronger in terms of collision resistance and avalanche effect.

V. Conclusion

The proposed Cellular Automata Based Hashing Algorithm (CABHA) has helped to design a strong cryptographic hash function using Cellular Automata rules. The CABHA has proven itself to be a hashing algorithm that introduces confusion and diffusion into the hash output using cellular automata based on non-linear rules and custom transformation. These functions together are capable of achieving a great avalanche effect which is an essential feature of any hash function. The proposed CABHA has an extremely low probability to have a collision for any given two message-key pairs M, K and M', K', and it is infeasible from a given hash H to find the message-key pairs M and K.

The proposed CABHA helps to realize the potential of hybrid cellular automata rules with custom transformation rules for secure hash generation.

VI. Reference

[1] R. L. Rivest, "RFC 1321: The MD5 Message-Digest Algorithm," *MIT Laboratory for Computer Science and RSA Data Security, Inc.*, 1992.

[2] C. Hanin, B. Echandouri, F. Omary, and S. El Bernoussi, "L-CAHASH: A novel lightweight hash function based on cellular automata for RFID," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2017.

[3] S. Tripathy and S. Nandi, "LCASE: Lightweight cellular automata-based symmetric-key encryption," *International Journal of Network Security*, 2009.

[4] M. R. Z. Baraa Tareq Hammad, Norziana Jamil, Mohd Ezanee Rusli, "A survey of Lightweight Cryptographic Hash Function," *International Journal of Scientific & Engineering Research*, 2017.

[5] D. Parashar, S. Roy, N. Dey, V. Jain, and U. S. Rawat, "Symmetric key encryption technique: A cellular automata based approach," in *Advances in Intelligent Systems and Computing*, 2018.

[6] NIST, "The Keyed-Hash Message Authentication Code," *Federal Information Processing Standard Publication*, 2008.

[7] L. H. Nguyen and A. W. Roscoe, "Short-output universal hash functions and their use in fast and secure data authentication," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012.

[8] S. Wolfram, "A new kind of science. Champaign, IL: Wolfram Media," *Author Notes*, 2002.

[9] J. Machicao and O. M. Bruno, "A cryptographic hash function based on chaotic network automata," in *Journal of Physics: Conference Series*, 2017.

[10] Mohamed Mahmoud ElRakaiby, "Cryptographic Hash Function using Cellular Automata," in International Journal of Computer Applications Technology and Research, Vol 5, Issue 5, 238 - 240, 2016.