International Journal of Computer Applications Technology and Research
Volume 5– Issue 5, 238 - 240, 2016, ISSN:- 2319–8656

Cryptographic Hash Function using Cellular Automata

Mohamed Mahmoud ElRakaiby

Communications and Electronics Department
Alexandria University
Alexandria, Egypt

Abstract: In this paper we make use of statistical properties of applying elementary cellular automata on a block of bits to generate a fixed size digest of that block to use it as hash function which can be use in different cryptographic applications.

Keywords: file digest; hash; one way function; cellular automata;

1. INTRODUCTION

Hash functions **H** plays an important role in different cryptographic applications which rely on hash function only or combine it with other cryptographic standards to produce a certain protocol suite a desired application

Data integrity is the most important application of hash functions. By using the message digests generated by a cryptographic hash function a system administrator can detect unauthorized changes in files.

In this paper we will introduce a new type of Hash functions using Cellular Automata which will be explained briefly as well

2. Hash Functions

Hash functions are mathematical computations that take in a relatively arbitrary amount of data as input and produce an output of fixed size. The output is always the same when given the same input (Figure 1). The inputs to a hash function are typically called messages, and the outputs are often referred to as message digests

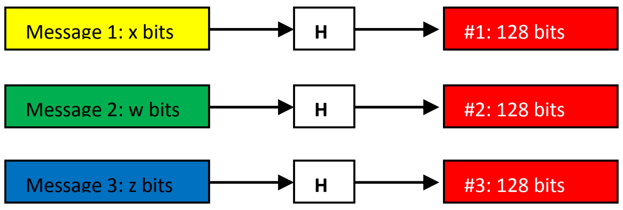

Figure 1. Hash Function Basic Description

All hash functions have the property that it is impossible to determine the input knowing only the output

Currently we have two main popular hash functions Message Digest 5 MD5 [1] which produce 128 bit digest for any input length and Secure Hash Algorithms SHA [2] which produces variety of digest sizes according to used standard (160 – 512) bits

Since two distinct messages are extremely unlikely to generate identical message digests, one can use this property of cryptographic hash functions to detect when a message has been altered. If one takes a binary file and computes a digest of the file, one can record this baseline digest. In the future, the digest can be recomputed on the file. If the new digest differs from the original baseline digest, then one can be assured that the file has been altered in some way

3. Cellular Automata

3.1 Elementary Cellular Automata

A cellular automata (CA) consists of a regular grid of cells, each in one of a finite number of states [3], such as on and off. The grid can be in any finite number of dimensions. For each cell, a set of cells called its neighborhood is defined relative to the specified cell. An initial state (time t = 0) is selected by assigning a state for each cell. A new generation is created (at t=t+1), according to some fixed rule that determines the new state of each cell in terms of the current state of the cell and the states of the cells in its neighborhood. Typically, the rule for updating the state of cells is the same for each cell and does not change over time, and is applied to the whole grid simultaneously

The simplest cellular automata system is one dimensional with two possible states per cell which we call it Elementary Cellular Automata with two adjacent neighbors per cell and according to cell status (Figure 2), neighbors status at time t=T , we get the cell status at time t=T+1 controlled by transition rule

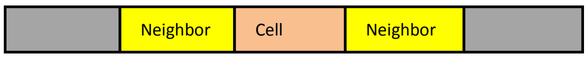

Figure 2. Cell and two neighbors

Cellular Automata also can be two dimensional [4] or three dimensional [5] which can be used in different applications but in this paper we use the simplest model of cellular automata

Figure 4 shows example of Rule 90 (Figure 3) in elementary cellular automata with 2 neighbors per cell (90=01011010), also Figure shows graphical representation of applying Rule 90 on 10010110000001000010010000000011 for 16 iteration

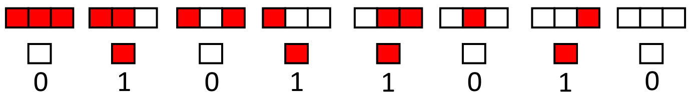

Figure 3. Rule 90

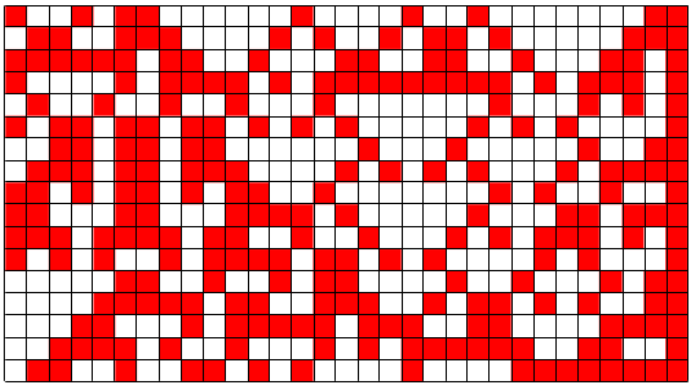

Figure 4. 16 Iterations using Rule 90

3.2 Cell Neighbors and Rule expansion

As we learnt so far that rule can be represented as the decimal equivalent of truth table output whose inputs are the all combinations of cell-neighbors states , it is easy in case of three neighbors rules

In our proposal in paper we will expand the neighbors to be six neighbors three on left of cell and three on right of cell (Figure 5) which will produce 128 combinations

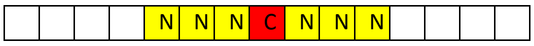

Figure 5. Cell and 6 neighbors

Rule is composed from bits of the right column of table 1 which represent the result of truth table of all combinations of cell and its neighbors

Table 1. Truth Table of Cell and 6 neighbors

N^{-3}	N^{-2}	N^{-1}	C^t	N^1	N^2	N^3	C^{t+1}
0	0	0	0	0	0	0	C^{127}
0	0	0	0	0	0	1	C^{126}
0	0	0	0	0	1	0	C^{125}
0	0	0	0	0	1	1	C^{124}
0	0	0	0	1	0	0	C^{123}
0	0	0	0	1	0	1	C^{122}
			. .				
1	1	1	1	0	1	1	C^4
1	1	1	1	1	0	0	C^3
1	1	1	1	1	0	1	C^2
1	1	1	1	1	1	0	C^1
1	1	1	1	1	1	1	C^0

4. Proposed System

We propose a main building block which the file to be hashed should be multiples of its size which is 8 * 128 bit

In order to maximize the diffusion, we first take each 128 bit block to be processed as shown in below block using elementary Cellular Automata using r=3

We use each block as a rule for elementary cellular automata for another block of same size then, use the result as next rule for next block and so on till finishing the remaining blocks of bits (7 blocks) .This process result of one block of 128 bits for each input block as shown in Figure 6.

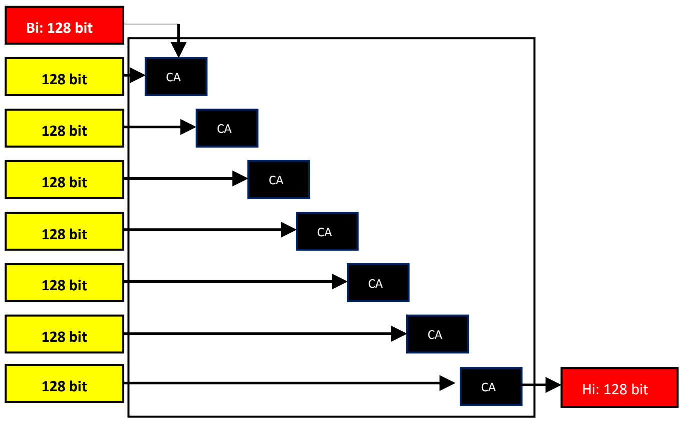

Figure 6. Initial Hashing of one block

Repeating this process for all 8 blocks can be presented as 8*8 initial hashing block with 8 inputs and 8 outputs (Figure 7).

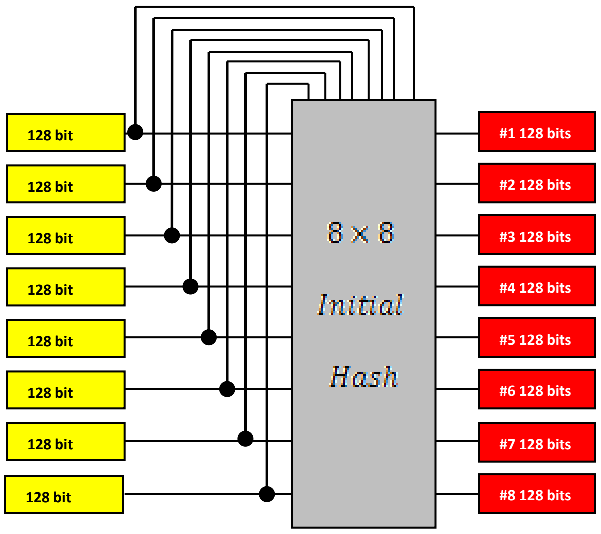

Figure 7. Initial Hashing 8 blocks

At that point, we produced 8*128 bit blocks as a result of the same size input so; we need to proceed in another step which is reduction step by processing each two successive blocks of 128 bits to produce only one block of 128 bit as shown in Figure 8.

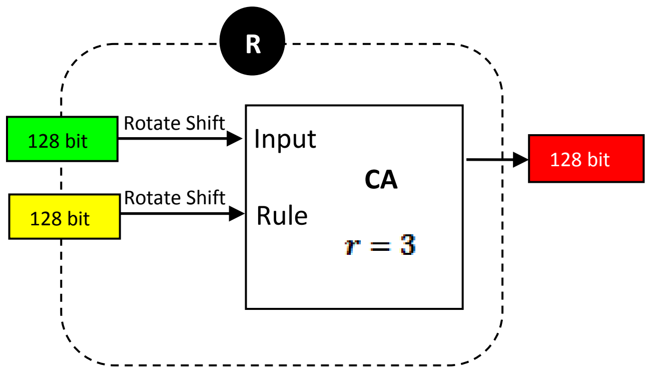

Figure 8. Single reduction round

Applying the reduction process on 8*128 bit blocks will result in 4*128 bit blocks and so on till we reach only one block of 128 (Figure 9) which will be considered as the digest of 8*128 bit blocks

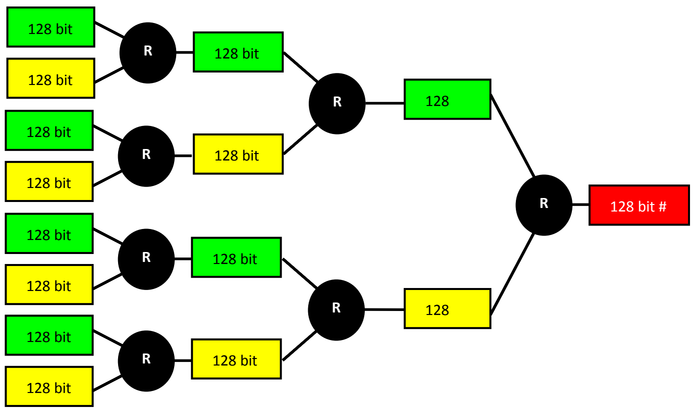

Figure 9. Reduction step of 8 x 128 bits to 128 bits

Last step is applying the same reduction process for resulting digests of building blocks for larger file sizes till we finally get fixed size hash for the whole file which is 128bit

Figure 10 is summary of hashing 4048 bits file

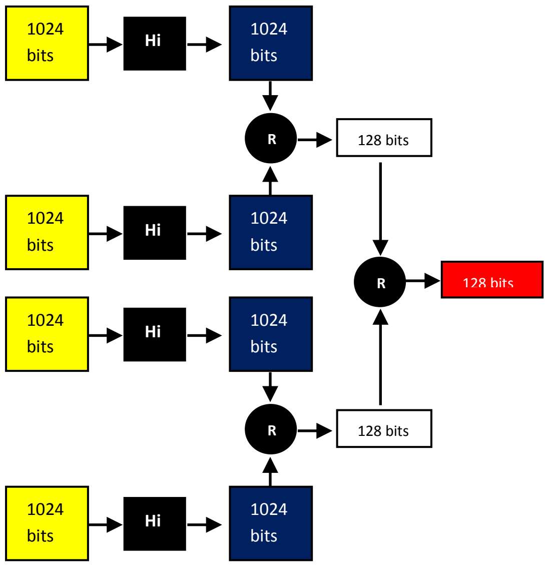

Figure 10. Summary of hashing 4048 bits file

5. Conclusion

We made use of cellular automata to prove that it can be used as a message digest function which produces a fixed size digest of a variable sizes files .It is sensitive to input file variation as shown in Figure 11

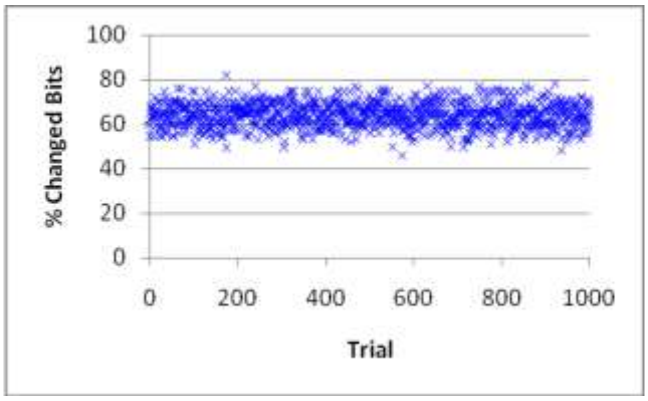

Figure 11. Changed bits after changing input by one bit

6. ACKNOWLEDGMENTS

Deep thanks to Prof. Hassan ElKamchouchi and Dr Fatma Ahmed for their continuous and endless support

7. REFERENCES

[1] Rivest RL (1991) The MD4 message-digest algorithm. Crypto, LNCS 537:303–311

[2] Secure Hash Standard (SHS), Federal Information Processing Standards Publication FIPS, PUB 180-4.

[3] Wolfram, Stephen (2002). A New Kind of Science. Wolfram Media.

[4] Norman H. Packard 1 and Stephen Wolfram, Two-Dimensional Cellular Automata , Journal of Statistical Physics, Vol. 38, Nos. 5/6, 1985

[5] R.W. Gerling , Classification of three-dimensional cellular automata , Physica A: Statistical Mechanics and its Applications , Volume 162, Issue 2, 1 January 1990, Pages 187-195