SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

JBEMi JOURNAL OF BIOMEDICAL ENGINEERING AND MEDICAL IMAGING

ISSN: 2055-1266
VOLUME 2 ISSUE 6

Robust Hashing Algorithm for Data Verification

Rushdi A. Hamamreh
Computer Engineering Department, Faculty of Engineering, Al-Quds University
rhamamreh@eng.alquds.edu

ABSTRACT

This paper presents a method for data authentication. Data authentication is the process of being able to verify the source of data. With data authentication, one can distinguish messages originating from the intended sender and an attacker. Data authenticity verification procedure uses cryptographic hash functions as the core algorithm. This algorithm based on linear combination of matrices to find non-invertible matrix, DILH takes advantage about of the compact representation of a set of numbers in a matrix, we design a robust method by employing strong collision resistance and reduces the hashing time.

Key Words— Hash function, Data authentication, Collisions, one-way cryptography, Data integrity DILH, non-invertable matrix.

1 Introduction

Data security is going to be a more and more important issue since various industries use information technology infrastructure for the great benefits. There is a huge number of data in different sectors (Cloud system, Health information system, Internet protocols, Information Retrieval Systems, Transport information network, and etc) . The information is very sensitive, requires high level and long-term preservation, and demands of data sharing. How to effectively ensure the authenticity of documents is a vital technology related to data security, information integrity and the interests of target groups (patients, students, and etc) [18].

Hash functions are important security primitives used for authentication and data integrity. The data integrity as a service based on the principles of security mechanism, that releases the data received are the same as send by an authorized entity using hash functions[1].

Hash functions H are one-way functions with as input a string of arbitrary length of the message m and as output a fixed length string of the hash value v. One-way functions work in one direction, meaning that it is easy to compute the hash value from a given message and hard to compute a message that hashes to a given hash value [1]. Hash algorithm is considered as the foundation algorithm of message security, identity, authentication, and message integrity [2].

The security of these applications depends on the cryptographic strength of the underlying hash function. Therefore some security properties are required to make a hash function suitable for such cryptographic uses: Pre-image resistance: Given a hash value v it should be hard to find any message m

DOI: 10.14738/jbemi.26.1644
Publication Date: 5th December 2015
URL: http://dx.doi.org/10.14738/jbemi. 26.1644

such that $= H(m)$, and a message $m1$ it should be hard to find another message $m2 \neq m1$ such that $H(m1) \neq H(m2)$ and Collision resistance: It should be hard to find different messages $m1, m2$ such that $H(m1) = H(m2)$ [2][8].

A hash collision is a pair of different messages $m1 \neq m2$ having the same hash value $H(m1) = H(m2)$. Therefore pre-image resistance and collision resistance are also known as weak and strong collision resistance, respectively. Since the domain of a hash function is much larger (can even be infinite) than its range, it follows from the pigeonhole principle that many collisions must exist. A brute force attack can find a pre-image for a general hash function with n-bit hashes in approximately 2^n hash operations. Because of the birthday

A brute force approach to generate collisions will succeed in approximately $\mathbf{2^{n/2}}$ hash operations. Any attack that requires less hash operations than the brute force attack is formally considered a break of a crypto graphical hash function[3][10].

2 Hash function properties

Hash functions $H(M)$ 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. The inputs to a hash function are typically called messages *M* , and the outputs are often referred to as message digest v. Hash function H has next security properties:

1. H should accept a block of data of any size as input.
2. H should produce a fixed-length output no matter what the length of the input data is.
3. H should behave like random function while being deterministic and efficiently reproducible. H should accept an input of any length, and outputs a random string of fixed length. H should be deterministic and efficiently reproducible in that whenever the same input is given, H should always produce the same output.
4. Given a message H, it is easy to compute its corresponding digest v; meaning that v can be computed in polynomial time $O(n)$ where n is the length of the input message, this makes hardware and software implementations cheap and practical.
5. Given a message digest v, it is computationally difficult to find M such that $H(M) = v$. This is called the one-way or pre-image resistance property. It simply means that one should not be capable of recovering the original message from its hash value.
6. Given a message $m1$, it is computationally infeasible to find another message $m2 \neq m1$ with $H(M1) = H(M2)$. This is called the weak collision resistance or pre-image resistance property.
7. It is computationally infeasible to find any pair of distinct messages $(m1, m2)$ such that $H(m1) = H(m2)$. This is referred to as the strong collision resistance property[1].

3 Matrix Hash Algorithm

Matrix K is a square matrix, the inverse is written K^{-1}. When K is multiplied by K^{-1} the result is the identity matrix I. Non-square matrices do not have inverses.

Not all square matrices have inverses. K square matrix which has an inverse is called invertible or nonsingular, and a square matrix without an inverse is called non-invertible or singular [12].

$$KK^{-1} = K^{-1}K = I$$

We call I here identity matrix. As we previously mentioned, not all matrices have Inverses but most of them have [12], so here we proposed an algorithm that converts any invertible matrix non-invertible one.

3.1 Hill Cipher

The core of Hill-cipher is matrix manipulations. It is a multi-letter cipher, is a type of monoalphabetic polygraphic substitution cipher. Hill cipher requires inverse of the key matrix while decryption. In fact that not all the matrices have an inverse and, therefore they will not be eligible as key matrices in the Hill cipher scheme [6][9]. Moreover, Hill cipher has several advantages such as disguising letter frequencies of the plaintext (M), its simplicity because of using matrix multiplication and inversion for enciphering and deciphering, its high speed, and high throughput [11].

3.1.1 Hill Cipher (Encryption, Decryption)

To Encrypt Plaintext Block size of m [9], we need key matrix **(**$K_{n\times n}$**)** with entries are between $(0, q - 1)$ included, but the determinant must be relatively prime to p, each entry in the plaintext block is between $(0, q - 1)$, included each block of plaintext is then an n-dimensional vector m .We encrypt vector m simply to produce the cipher text vector $\boldsymbol{c}$ using the following linear algebra equation:

$$c = m \times k \; mod \; N$$

To Decrypt cipher text vector $\boldsymbol{c}$ [9], we need first to find the inverse matrix k^{-1} to k , where that matrix must be invertible . Then can calculate m from the mathematical model

$$m = c \times k^{-1} \, mod \, N$$

In [5][9] they proposed new technique to convert any non-invertible matrix's to invertible ones. As a result, Hill cipher being a efficient algorithm because any encrypted text will decrypted using the key matrix [5][10].

3.1.2 Hill Cipher (Hashing algorithm)

The main point of one-way hash algorithm is that any encrypted text cannot be decrypted [11]. From this point, we need to choose the non-invertible matrix from the hill cipher to use it inside the practical one- way hash algorithm.

$$H(m) = m \times k \; mod \, N$$

$$H(m_i) = v_i$$

Where k is the non-invertible matrix. In [11] author works on an algorithm that generate non-invertible matrix and multiply it by plaintext as column vector with modular value N to generate the hash value V.

4 Proposed Algorithm (DILH)

Data integrity using linear combination for Hash algorithm (DILH) uses non-invertible matrix to produce hash value V. This algorithm selects and generates non-invertible matrixes using linear combination of rows or columns of a matrix to ensure that the hash values are collision-free and one-way properties [1].

In this section, we plot the diagram for our proposed algorithm which showing each step inside it and how it works , It is also showing the mathematical proof of our work. Besides that, we have the DILH algorithm analysis and result and its comparison with other hashing algorithms including SHA-1 ,MD5 [17].

4.1 Programmable Hash algorithm

According to figure 2, the step of DILH algorithm structured as the followings:

Step1 (Input): input M.

Step2 (Padding): Pad (P) , $P(M)$.

Step3 (Splitting): M is split into.q is the number of blocks. blocks $(\mathrm{m}_1 , \mathrm{m}_2 ,.., \mathrm{m}_\mathrm{i})$,each of length $n \times n$ suitable for the hashing block.

Step4 (Key generation): Key matrix generation k_i :

4.1: Generate a random matrix (R) with size $(\mathrm{n}-1, \mathrm{n})$ and value in a interval (0,1).

4.2. Casting (R)

4.3. Introduce a new row in the matrix R so as the size of the resulting matrix k_i is $n \times n$. The added row is obtained by ***linear combination*** of row i and row j of matrix R according to:

$$\boldsymbol{K(N,:) = c_1 \times R(i,:) + c_2 \times R(j,:)}$$

where c_1, c_2 are arbitrary integer constants. This ensure that the inverse of K denoted by K^{-1} does not exist.

Step5 (GenerationV_i): Hash value V_i generation using this formula:

$$\boldsymbol{H_{xi} = m_i \times k_i = v_i}$$

Step6 (Digest): Digest V_i.

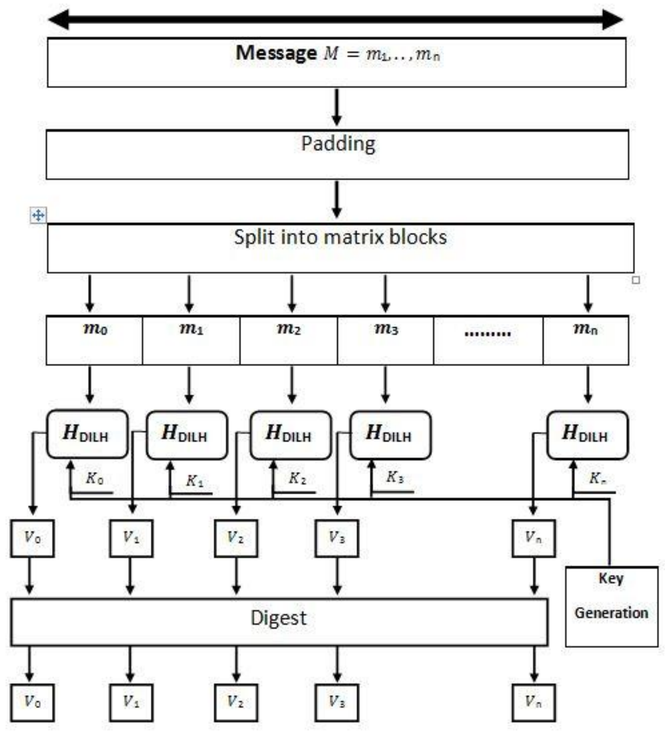


Figure.1 DILH diagram

4.2 Mathematical Model Proof

The first step is to define *linear independence*. Given a set of vectors[12]:

$$v_1, v_2, \ldots, v_k$$

we look at their linear combinations:

$$c_1v_1 + c_2v_2 + \ldots\ldots + c_kv_k$$

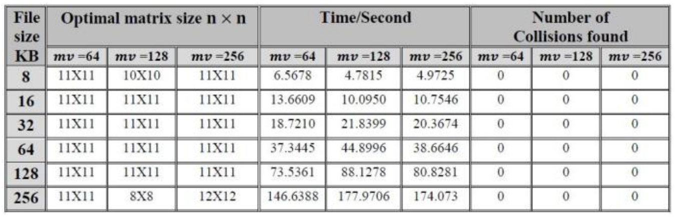
Where $c_1, c_2, \ldots, c_k$ are arbitrary constant weights. The trivial combination, with all weights $c_i = 0$, obviously produces the zero vector $0v_1 + \cdots + 0v_k = 0$. The question is whether this is the only way to produce zero. If so, the vectors are independent. If any other combination of the vectors gives zero, they are dependent [12].

It is well known that a matrix A is invertible if and only if its rows and columns are linearly independent [12]. This mean that if rows and columns of a matrix are linearly independent then the matrix is invertible, otherwise it's not invertible.

5 Simulation and Results

A different file sizes is randomly selected and is fed into the algorithm to generate different hash Values with three modular values N=64,128,256 to analyze the time delay and to find if there is any collisions.

Table 1: Optimal time in seconds required for the proposed DILH Algorithm with number of collisions found



File size KB	Optimal matrix size n × n			Time/Second			Number of Collisions found		
	mv =64	*mv* =128	*mv* =256	*mv* =64	*mv* =128	*mv* =256	*mv* =64	*mv* =128	*mv* =256
8	11X11	10X10	11X11	6.5678	4.7815	4.9725	0	0	0
16	11X11	11X11	11X11	13.6609	10.0950	10.7546	0	0	0
32	11X11	11X11	11X11	18.7210	21.8399	20.3674	0	0	0
64	11X11	11X11	11X11	37.3445	44.8996	38.6646	0	0	0
128	11X11	11X11	11X11	73.5361	88.1278	80.8281	0	0	0
256	11X11	8X8	12X12	146.6388	177.9706	174.073	0	0	0

From table 1, we can observe that there are no collisions found over all file sizes with its hash values. DILH shows that hash value 968 bit is repeated more than one time compared to other hash values, beside that there is no collisions found over it .So we can consider the hash value 968 bit as a good option to deal with it DILH algorithm.

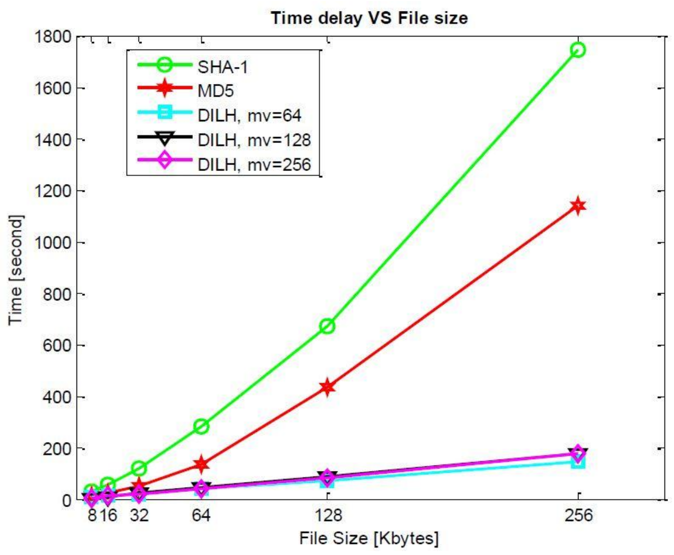


Figure 2: Hash algorithms comparison (SHA-1, MD5, and DILH) in term of time delay versus file size

In this simulation we examine different hash values length started from 2x2 which also equal 32 bits to 12x12 which equal 1152 bits. DILH algorithm find collisions only in 32 bit hash value length.

Fortunately MD5 and other common hash functions have substantially larger key lengths than 64-bits. For MD5 the key length is 128 bits, for SHA-1 the key length is 160 bits, SHA-256 the key length is 256 bits.[17]

The MD5 algorithm breaks a file into 512 bit input blocks. Each block is run through a series of functions to produce a unique128 bit hash value for the file.

Wang,Yu find two 128 byte messages with same MD5 hash value[10].

In SHA-1 Collisions found in 2^{80} operations of reduced version of SHA-1--53 out of 80 rounds[16].

We implement our hash scheme in a Dell vostro 1015 laptop, 2.10GHz 2 core(s), 2GB RAM/ Microsoft windows7/MATLAB 7.9.0 simulator.

6 Analysis and Conclusion

Based on the analysis of the data in Table 1, our algorithm shows its strength in dealing with the collisions. As shown DILH algorithm didn't find any collisions in hash values with different file sizes from

8KB to 1024KB after 3X3 matrix size. DILH shows collisions just in 2X2 with all file sizes, as shown the algorithm find 474 collisions in 2X2 matrix with size 64KB and $N = 128$ but this number of collisions is just 2.921% of overall hash values generated that's where number of hash values generated is 16223 hash values. Also in the same case when $N = 256$ number of collisions is just 0.5178 % where number of hash values generated is 16223. From our analysis when you increase N the number of collisions is decreased in an Inverse relationship.

We check collisions in the new one-way data integrity hash algorithm DILH that based on special kinds of non-invertible metrics. Particularly, DILH shows its strength in dealing with collisions, as shown in table 1 DILH didn't find any collision after 72 bit hash value length. In our experiment $\mathrm{N} = 64, 128, 256$.

In our proposed algorithm DILH [2] , the time delay have been tested , DILH shows that the best time for calculating hash value in 8KB is 800bit when $\mathrm{N} = 128$ and 968 bit when $\mathrm{N} = 256$.

REFERENCES

[1] William Stallings. Cryptography and Network Security Principles and Practices, 5th Edition, January 24, 2010.

[2] Rushdi Hamamreh, Mohammed A. Jamoos, Raid Zagha.l DILH: Data Integrity using Linear Combination for Hash Algorithm, ICITeS-Edas-1569740315-18.

[3] Shangping Wang, Yaling Zhang, Youjiao Zou, Jin Sun. A New Hash Algorithm Based on MQ Problem and Polymorphic, International Conference on Information Science and Technology,March 26-28, 2011 Nanjing, Jiangsu, China.

[4] Schneir, Bruce. Applied Cryptography Protocols, Algorithms and Source code in C. s.l. : John Wiley & Sons, 1996. 0471128457.

[5] Songsheng Tang, Fuqiang Liu. A one-time pad encryption algorithm based on one way hash and conventional block cipher, Qingdao, P.R.China, Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on , 21-23 April 2012 .

[6] Sarat Kumar Patra, Invertible. Involutory and Permutation Matrix Generation Methods for Hill Cipher System, International Conference on Advanced Computer Control, National Institute of Technology Rourkela, Orissa-769008, India.

[7] Artan Berisha, Behar Baxhaku. A Class of Non Invertible Matrices in GF (2) for Practical One Way Hash Algorithm. International Journal of Computer Applications (0975 – 8887), VOL. 54– No. 18, September 2012, China.

[8] ZHOU Tian-shu, LI Jing-song. Development of Data Authenticity Verification System in Regional Health, IEEE International conference ITIME, August 2009.

[9] James H. Burrows. Secure hash standard, federal information processing standards publication (Supersedes FIPS PUB 180 – 1995 May 11).

[10] Wang X Y, Yu H B, Yin Y Q. Efficient collision search attacks on SHA-0,2005, Lecture Notes in Computer Science 3621 1.

[11] Rushdi A. Hamamreh, Mousa Farajallah. Design of a Robust Cryptosystem Algorithm for Non-Invertible Matrices Based on Hill Cipher, International Journal of Computer Science and Network Security; pp 11-16, 2009.

[12] Jianqiu Ji, Jianmin Li, Shuicheng Yan. Min-Max Hash for Jaccard Similarity. 13th International Conference on Data Mining, Dallas, Texas, December 7-10, 2013.

[13] Gilbert strang. Linear Algebra and Its Applications, 4th Edition, May 9, 2011.

[14] Ahmed Y. Mahmoud and Alexander G. Secure Hill Cipher Modifications and Key Exchange Protocol, Chefranov. IEEE International Conference on Automation Quality and Testing Robotics (AQTR), 28-30 May 2010, VOL. 2 .

[15] Danilo Gligoroski, Smile Markovski and Svein J. Knapskog. A Secure Hash Algorithm with only 8 Folded SHA-1 Steps", IJCSNS International Journal of 194 Computer Science and Network Security, 2006, VOL.6 No.10.

[16] X. Wang, Y. Lisa Yin, H. Yu. Finding Collisions in the Full SHA-1, Crypto 2005, LNCS ,3621 , pp. 17-36, 2005.

[17] X. Wang, H. Yu. How to break MD5 and Other Hash Functions, Advances in EUROCRYPT2005, LNCS 3494, pp. 19–35, 2005.

[18] Jingkuan Song, Yi Yang. Robust Hashing with Local Models for Approximat Similarity Search. IEEE Transaction on Cybernetics VOL. 44, NO. 7, July 2014.