Alternative Hash Function based on NP-hard Problem

Ali Alshahrani
Faculty of Computer Studies, Arab Open University, Saudi Arabia

Abstract

Secure hash function is used to protect the integrity of the message transferred on the unsecured network. Changes on the bits of the sender's message is recognized by the message digest produced by the hash function. Hash function is mainly concerned of data integrity, where data receiver needs to verify whether the message has been altered by eavesdropping by checking the hash value appended with the message. To achieve this purpose, we have to use secure hash function able to calculate the hash value of any message. In this paper we introduce an alternative hash function based on NP-hard problem. The chosen NP-hard problem is known as Braid Conjugacy problem. The braid conjugacy problem forms the core of the hash function. The aim of using this hard problem is to make sure that the hash function is irreversible and secure against cryptanalysis attacks. The results shows that the proposed hash function is secure mathematically, and it provides huge key search space.

Keywords: *Hash function, Cryptography, MD5*

1. Introduction

Hash function is the core of any cryptosystem. It is used for message integrity or for authenticating the data exchanging process between the connected parties. The design of a secure hash function consists of a special one way function which receives any variable length input and produces a fixed length output. One way function is defined as a function that can simply take the input message and compute (generate) the corresponding hash value, but it is computationally infeasible to recover the original message using the hash value. A hash function is called ideal if its hash value *h* cannot be distinguished from the values given by a random oracle [1].

Apart from hash functions, some cryptosystems are depending on mathematically hard problems. An example of mathematical hard problem is the braid theory. Generally, Braid Groups had been widely used as a tool to create various cryptographical primitives. There are a few of them such as public key cryptosystem, key exchange, authentication and digital signature [2] [3]. Creating an ideal hash function using braid groups is connected to the general question of finding a function to map the braid groups to the sequence of {0,1}. The result of the secured hash function must be random enough and reveals no information about the argument of the hash function.

The objective of this paper is therefore to create a secured hash function based on braid group's theory. The mechanism used in the core of this function is the braid multiplication, by which we multiply a pre-defined braid by the braid generated from message transformation (transformation of the message's content to a braid form). However, the designed hash function then can be attached to any cryptosystem for message integrity purposes with high level of security.

2. Literature Review

Cryptography is one of the major branches of Computer Security sciences. Cryptosystem is the collection of all procedures, protocols and cryptographic algorithms used for encrypting and decrypting messages. It consists of

integrated assembly of cryptographic primitives (e.g. encryption algorithm, hash algorithms, etc.), protocols, operational procedures which together make effective security platform.

Hash function has been applied for many security applications and protocols such as PGP, SSL, SSH, IPsec, TLS and S/MIME [4]. In order to provide these applications with high level of security, we have to design a secure hash function against existing attacks. Let us discuss the following scenario to understand the usage of hash function: Alice want to send a message m to Bob. Alice needs to use hash function F_h to calculate the hash value h of here message such that $F_h(m) = h$, and append the hash value h with the message. On the other hand, Bob (the receiver) needs to recalculate h using the same hash function. By comparing the two hash values, Bob can judge whether the message has been altered or not. The message considered "Altered" if $F_h(m)\text{Alice} \neq F_h(m)\text{Bob}$.

The strength of any hash function can be measured by the complexity of its calculation and operations [5]. Recently, cryptosystems aim to use some mathematical NP-hard problems in order to increase the complexity of its structure against the attackers. A problem is assigned to the NP (Nondeterministic Polynomial time)-hard problem class if it is solvable in polynomial time by nondeterministic oracle machine. Therefore, if we build a hash function based on NP-hard problem, we will be sure that attackers can not attack this function since it is based on "hard-to solve" mathematical problems.

2.1 Hash Function

A hash function F_h is a transformation that takes an arbitrary size input m and returns a string with a fixed size, which is called the hash value h (where $h = F_h(m)$) [6].

A cryptographically secure hash function should have the basic requirements in its design, which are:

F_h can be applied to an input of data of any size.

- F_h produces a fixed-length of output.
- F_h (m) is relatively easy to compute for any given m.
- F_h (m) is one-way.
- F_h (m) is collision-free.

MD2, MD5 and SHA [7] are good examples of well-known hash functions. In 1989, Ron Rivest introduced the MD2 Message Digest Algorithm which takes as input a message of arbitrary length and produces as output a 128-bit message digest, by appending some redundancy to the message and then iteratively applying 32 bytes to 16 bytes compression function. Researches done by [8] and [9] proved that the MD2 is not one-way function, therefore it is not collision-free, and they showed that it does not reach the ideal security level of 2^{128}. However, the use of MD2 for new applications is discouraged.

Similarly, MD5 takes as input a message of arbitrary length and produces a 128-bit message digest, but it is aimed at 32-bit machines instead of 8-bits machine in MD2. The algorithm consists of four distinct rounds with similar structure, but each uses a different primitive logical function. According to the researches done by [10] and [11], MD5 is not secure to be used in security applications since it is not collision-free. Therefore, MD5 in no longer recommended for new applications where collision-resistance is required.

MD2 and MD5 are meant for digital signature applications where a large message has to be "compressed" in a secure manner. They are classified in bit-operations based hash function category, since they depend on crossing, shifting and addition of the message's bits.

The Secure Hash Algorithm (SHA-1) is another example of hash algorithms. It is one of the most widely used hash functions in the world. Besides, four more variants have since been issued with increased output range and a slightly different design: SHA-224, SHA-256, SHA-384 and SHA-512 (sometimes they are collectively referred as SHA-2). However, SHA-1 takes a message with a maximum less than 2^{64} as an input produces 160-bit message digest.

The overall process of SHA-1 consists of five steps, starting from appending some padding bits to make the message congruent to 448 modulo 512, ending by the 160-bit message digest. Through these steps, the message length must be appended to the message as well as XOR operations applied to the message's bits.

A research done by Chinese researchers showed that SHA-1 has been broken [12]. They present new collision search attacks on the hash function SHA-1 and showed that collision of SHA-1 can be found with complexity less than 2^{69}.

2.2 Braid Group

Braid groups had been widely used as a tool to create various cryptographical primitives. There are a few of them such as public key cryptosystem, key exchange, authentication and digital signature. However, Conjugacy Problems are NP-hard problems in braid group theory. We say that braid *a* and *b* are conjugate if we have $a = s\ b\ s^{-1}$ for some braid *s*. Conjugacy Search Problem is one of the conjugacy problems in braid theory. This problem lies, that for some braids $(a,b) \in B_n \times B_n$ (where B_n is braid group) such that *x* and *y* are conjugate, find $r \in B_n$ such that $b = rar^{-1}$.

Any braid can be decomposed as a product of simple braids. One type of simple braids is the Artin generators σi that have a single crossing between *i*-th and *(i+1)*-st strand as in Figure 1. Besides, the *n*-braid group *Bn* can be presented by the Artin generators $\sigma_{1,\ldots,}\ \sigma_{n-1}$ and relations $\sigma_i\ \sigma_j = \sigma_j\ \sigma_i$ for $|i - j| > 1$ and $\sigma_i\ \sigma_j\ \sigma_i = \sigma_j\ \sigma_i\ \sigma_j$ for $|i - j| = 1$.

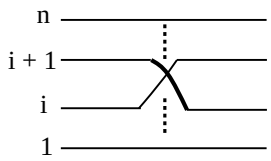

Figure 1: Artin generator σi

Many operations can be applied on two braids. For instance, braid multiplication is the most used operation over braid. The multiplication of braids *a* by *b* where $(a,b) \in B_n$ results a new braid which is unique. The process of figuring out the original two braid (braid *a* and *b*) given the resulted braid after the multiplication is known to be a hardproblem. The multiplication of two braids is carried out by placing the braid *a* under the braid *b* as in Figure 2.

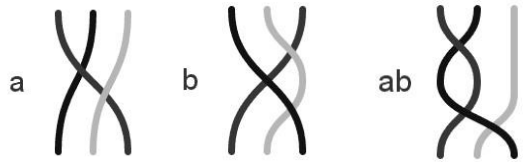

Figure 2: Braid Multiplication

As we mentioned previously, many cryptosystem's primitives have been built on braid group theory, but no hash function based on braid has been implemented yet. However, many researches done in braid group, and most of these researches showed the strength of this theory against attacks.

3. The Proposed Hash Function Architecture

Currently, most of the existing hash functions are focusing on the scrambling and shifting the bits of the input blocks. In intention of randomizing the bits of the input blocks, usually they are using the exclusive OR (XOR) operation and some addition in their implementation. For our works, we proposed a new approach of hash function architecture. In our opinion, hash function is not just scrambling or shifting the bits but should also include the mathematical hard problems. We have found that the braid groups' theory is the best way to do so as it provides mathematically hard problems and also some advantages in computational aspects.

The proposed structure consists of an initial vector called initial braid and blocks of text (represented as braid) to be the input to the hash function. We apply a braid operation (multiplication) on the braid groups to concatenate two different braids which then produce a new unique braid. By repeating the process, we get a random braid that cannot be traced back to get the initial value of the hash function. This condition is able to fulfill the important properties of a secured hash function.

The architecture of the proposed hash function (see Figure 3) will carry the steps as follows:

- Generate a random braid B_{IV}, to be as an initial vector of the hash function.
- Generate another braid by manipulating the bits from the text blocks.
- Do the multiplication operation on the initial braid and the braid generated before.
- Repeat the iteration until the last text blocks.

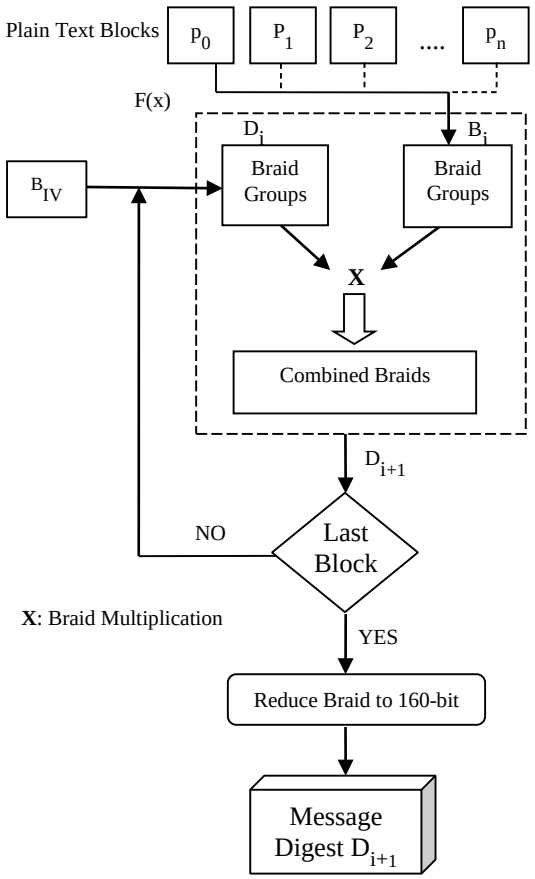

Figure 3: Architecture of the proposed hash function

Our algorithm takes an input message of arbitrary length and produces an output digest message of 160-bit. The input is processed in 192 bit blocks. The combined braid as illustrated in the architecture, which is resulted from braid multiplication will be processed in order to reduce the size of the digest to 160-bit. The overall processing of a message to produce a digest consists of four stages. The stages are:

- Stage 1: Append Padding Bits
  The 192 bit block is padded to make sure the length is always in the desired length. The padding process is done by taking the first 8-bit block from the input message (which is less from the desired length) and then cyclic left shift the bits of the block by 2 bits. After appending the padding bits, we will XOR every 8-bit in 192-bit block. The result of this stage is 12 8-bit blocks.

- Stage 2: Convert to Artin Generators, σ
  This process is the beginning of mapping the input into a braid representation. As we can see in Figure 4, B[i] represents the braid index or in other words, the location where crossing occurs in braid groups.

By mapping the input into a braid representation, we need to calculate the value of the crossing. With the number of strands n = 128, we convert the first 7 bits from binary to positive decimal. The 8th bit indicates the sign of the number which can be negative or positive. The positive sign indicate a positive crossing and the negative sign indicate negative crossing.

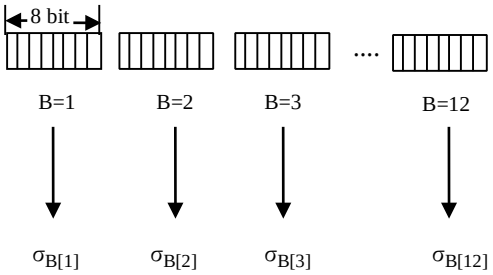

Figure 4 : Relation between blocks of bits with Artin generators

- Stage 3: Braids Multiplication
  The inputs of this stage are two braids with 12-byte size (12 crossing). The first braid represents the plain text block after the transformation (transforming the plain text block to Artin representation B_i. The second braid will be the initial value of B_{IV} which is represented as braids D_i with 24-byte (B_{IV} will be used for one time only, in the beginning of this stage). However, the initial value of D_i will be reduced to 12-byte size to be multiplied by B_i. The braid reduction occurs by XORing D_{2i-1} and D_{2i} for all values of $1 \leq i \leq 12$. Therefore, the resulted braid after reduction is 12-byte braid size which is represented as D'_i.

  The combined braid resulted from multiplying the two braids D'_i and B_i will be in the size ranging from 0-24 crossing since there is a possibility for zero crossing. However, the combined braid then will be multiplied by the next input block after reduction to 12-byte instead of using the same value of B_{IV}.

- Stage 4: Message Digest Reduction and Production
  This is the final stage where we produce the message digest of the corresponding plain text. However, this stage will be executed when we reach the last input block. The output will be in the size of 192-bit, meanwhile we are looking for 160-bit size. Therefore, we will reduce the output size to 160-bit by keeping the first 160-bit and ignoring the rest bits as portrait in Figure 5.

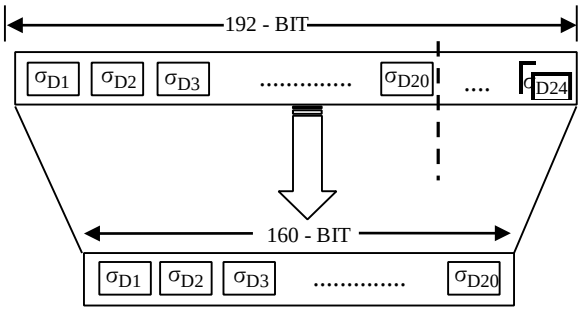

Figure 5: Message Digest Reduction

4. Discussion

The two important parameters should be discussed in this part are: the security and the performance of the proposed architecture. In term of security, 8-bit block of plain text will produce an Artin representation of a string in a 128 braid (2^7=128), which is big enough for security purposes, since the advice size for braid to be used for cryptography purpose is 80 strings braid. Mathematically, braid theory proved to be secure since it is virtually impossible to retrieve one of the multiplied braids after braid multiplication operation.

In term of performance, a block of 192-bit will require one braid multiplication of two 128-strings braids, and 24 XOR operations. This can be considered minimal operations that need to be applied on every 192-bit plain text block.

5. Conclusions

In conclusion, bit-operations based hash functions which depend on shifting or XORing message's bits present less security level than presented by hash functions based on NP-hard problems. However, the proposed hash function is worth to be evaluated, because it is proved that it is secure since it is based on hard to solve mathematical problems. The internal stages of the proposed function depend on mapping the bits into braid representation and multiplying the resulted braids by each others. These stages which form the core of the overall architecture of the proposed hash function can fulfill the important properties of a secure hash function.

References

[1] P. Hofmann and B. Schneier, Attacks on Cryptographic Hashes in Internet Protocols, facs.org. November 2005. [Online]. Available: http://www.faqs.org/rfcs/rfc4270.html [Accessed March 20, 2018]

[2] K. Ko, S. Lee, J. Cheon, J. Han, J. Kang and C. Park, *New Public-Key cryptosystem using braid groups*, In advance in Cryptology: Crypto 2000.

[3] I. Al-Siaq, Public Key Cryptosystems based on Numerical Methods, Global Journal of Pure and Applied Mathematics, Vol.13, No.7 pp (2017) pp 3105-3112.

[4] *R. Dobai, J. Korenek*, L. Sekanina, Evolutionary design of hash function pairs for network filters, Applied Soft Computing, Volume 56, July 2017, Pages 173-181.

[5] G. Yu, Y. Zhao, C. Lu, J. Wang, HashGO: hashing gene ontology for protein function prediction, Computational Biology and Chemistry, Volume 71, December 2017, Pages 264-273.

[6] *Y. Cui, J. Jiang, Z. Lai, Z. Hu, W.* Wong, Supervised discrete discriminant hashing for image retrieval, Pattern Recognition, Volume 78, June 2018, Pages 79-90.

[7] W. Stalling, *Cryptography and network security: principles and practices*, Prentice Hall 2nd ED. 1999.

[8] N. Rogier and Chauvaud, *The Compression Function of MD2 is not Collision Free*, Selected Areas in Cryptography '95, 1995.

[9] F. Muller, *The MD2 Hash Function is not One-Way*, Advanced in Cryptology-Asia Crypt '2004, 2004.

[10] V. Klima, *Finding MD5 Collision-a toy for a Notebook*, Cryptology ePrint Archive, Report 2005/075.

[11] X. Wang, D. Feng, X. Lai, H. Yu, *Collision for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD*, Cryptology ePrint Archive, Report 2004/199.

[12] *X. Wang, Y. Yin, H. Yu*, Finding Collisions in the Full SHA-1, *Crypto 2005*.