BHA-160: Constructional Design of Hash Function based on NP-hard Problem

Ali AlShahrani
Faculty of Computing Studies
Arab Open University, Riyadh, Kingdom 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 are recognized by the message digest produced by the hash function. Hash function is mainly concerned with data integrity, where the 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 a secure hash function that is 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. This problem has proved to be secure against cryptanalysis attacks.**

Keywords—Hash function; integrity message; cryptanalysis; attack; NP-hard problem

I. 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 that receives any variable length input and produces a fixed length output. A 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 the hash value h cannot be distinguished from the values given by a random oracle [1]. Apart from hash functions, some cryptosystems are dependent on mathematically hard problems. An example of a mathematical hard problem is the braid theory. Generally, Braid Groups had been widely used as a tool to create various cryptographic primitives. There are a few of them such as a 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 reveal no information about the argument of the hash function. The objective of this paper, therefore, is 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 importance of this research is related to the capability of our designed hash function to be attached to any cryptosystem for message integrity purposes with a high level of security. The rest of the paper is organized in six sections. The related works are discussed in Section 2. The proposed hash function is presented in Section 3. In Section 4, the algorithm performance is analyzed. The discussions and conclusion are presented in Sections 5 and 6, respectively.

II. Literature Review

Hash functions have 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 a high level of security, we have to design a secure hash function against existing attacks. Let us now discuss the following scenario to understand the usage of a hash function: Alice wants to send a message *m* to Bob. Alice needs to use the hash function F_h to calculate the hash value *h* of her message such that $F_h(m) = h$, and appends 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 operation [5]. Recently, cryptosystems aimed to use some mathematical NP-hard problems in order to increase the complexity of their 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 a nondeterministic oracle machine. Therefore, if we built a hash function based on a NP-hard problem, we will certain that the attackers cannot attack this function since it is based on a "hard-to solve" mathematical problem.

A. Hash Function

A hash function F_h, is a transformation that takes an arbitrary size input *m*, and returns with a string of 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 that 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 applies 32 bytes to 16 bytes compression function. Researches done by [8] and [9] proved that the MD2 is not a one-way function, therefore, it is not collision-free and they also 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; however, it is aimed at 32-bit machines instead of 8-bit machines in MD2. The algorithm consists of four distinct rounds with a similar structure, but each uses a different primitive logical function. According to the research 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 a bit-operations based hash function category, since they depend on crossing, shifting and addition to 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. Indeed, 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 producing a 160-bit message digest. The overall process of SHA-1 consists of five steps, starting from appending some of the padding bits to make the message congruent to a 448 modulo 512, ending with a 160-bit message digest. Through these steps, the message length must be appended to the message as well as XOR operations being applied to the message's bits. Research done by Chinese researchers showed that SHA-1 has been broken [12]. They presented new collision search attacks on the hash function SHA-1 and showed that the collision of SHA-1 can be found with a complexity of less than 2^{69}.

B. *Braid Group*

Number equations consecutively: Equation numbers, within parentheses, are to position flush right, as in (1), using a right the second category of our hash function's classification is the hash function based on the NP-hard problem. In this category, the heart of the hash function depends on a mathematical nondeterministic polynomial-time hard problem. Braid groups had been widely used as a tool to create various cryptographically primitives. There are a few of them, such as a 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 braid is the Artin generators σ_i, these have a single crossing between *i*-th and *(i+1)*-st strand as in Fig 1. Besides, the *n*-braid group B_n 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 $|i - j| = 1$.

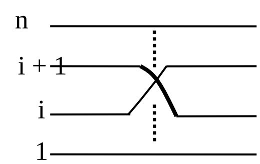


Fig. 1. Artin Generator σ_i

Many operations can be applied on two braids. For example, braid multiplication is the most used operation over braid. The multiplication of braids *a* by *b* where $(a,b) \in B_n$ results in a new braid which is unique. The process of ascertaining the original two braids (braid *a* and *b*), given the resulted braid after the multiplication, is known to be a hard-problem. The multiplication of two braids is carried out by placing the braid *a* under the braid *b*. As we previously mentioned, 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 are done in the braid group, and most of these researches showed the strength of this theory against attacks.

III. Proposed BHA-160 Hash Functuion Architiecture

Currently, most of the existing hash functions are focusing on scrambling and shifting of the bits in the input blocks. With the intention of randomizing the bits of the input blocks, usually they are using the exclusive OR (XOR) operation and some additions in their implementation. For our work, 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 group's theory is the best way to do this, as it provides mathematical 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 inputted into the hash function. We apply a braid operation (multiplication) on the braid groups to concentrate two different braids that then produce a completely new unique braid. By repeating this process, we will 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 will follow the steps as follows as in Fig. 2:

- 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 a multiplication operation on the initial braid and the braid generated beforehand.
- Repeat the iteration until the last text blocks.

A. BHA-160 Processing Stages

The algorithm of BHA-160, takes as input a message of arbitrary length thereby producing as output, a 160-bit message digest. The input is processed in 192 bit blocks. The combined braid, as illustrated in the architecture, is achieved from the braid multiplication and 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 as shown in Fig. 3.

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. Fig. 4 presents the input setup of BHA-160 in the first stage.

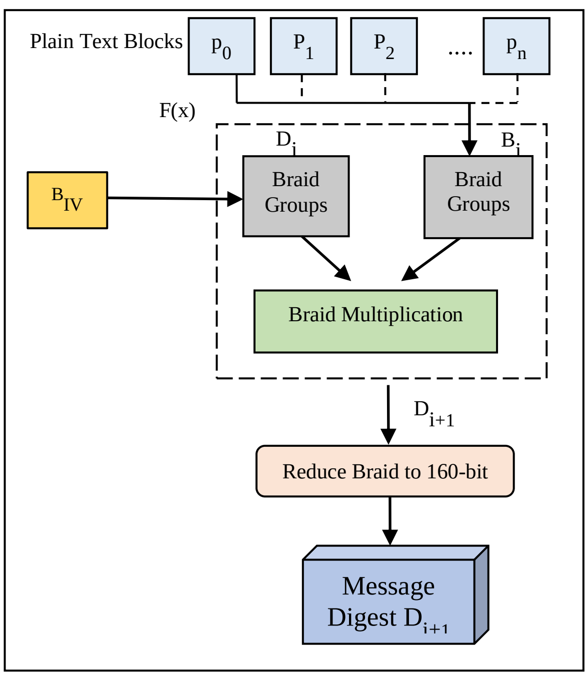


Fig. 2. Architecture of the Proposed Hash Function (BHA-160).

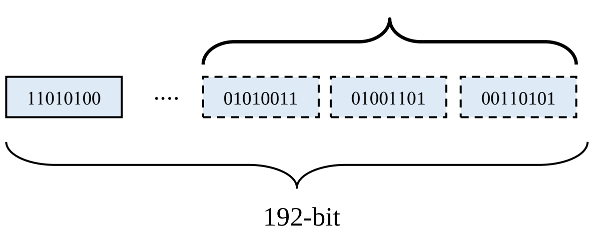


Fig. 3. Padding Process for BHA-160.

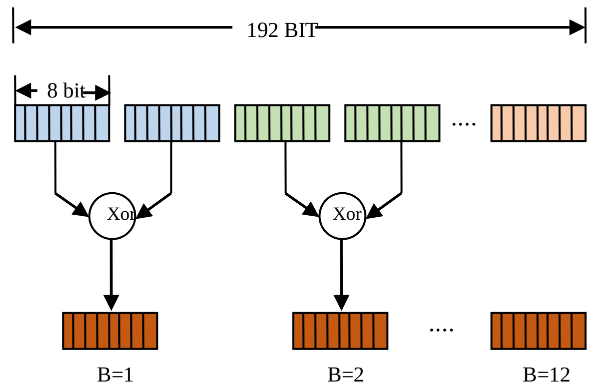


Fig. 4. Input Setup of BHA-160.

- 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 Fig. 5, 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 that can be negative or positive. The positive sign indicates a positive crossing and the negative sign indicates a negative crossing.

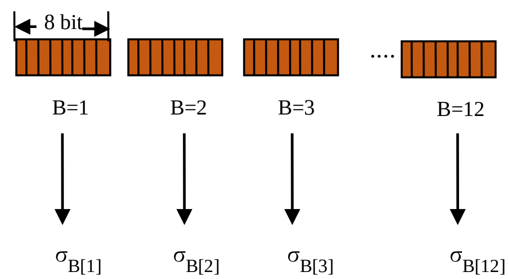


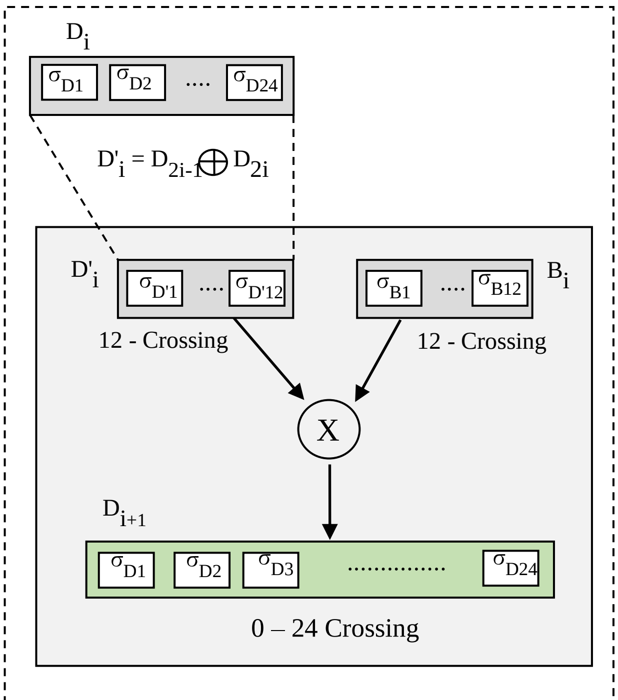
Fig. 5. Relation between Blocks of Bits with Artin Generators.

- 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} that is represented as braids D_i with 24-byte (B_{IV} that 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 a 12-byte braid size which is represented as D'_i. The combined braid, resulted from multiplying the two braids D'_i and B_i as shown in Fig. 6 will therefore 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 remaining bits as portrait in Fig. 7.



X: Braid Multiplication

Fig. 6. Braid Multiplication.

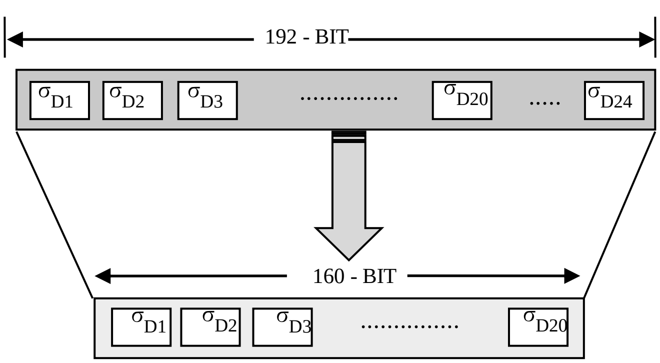


Fig. 7. Message Digest Reduction.

IV. Performance Analysis

The performance of BHA-160 is examined against well-known hash functions, including: MD2, MD5, SHA-1, SHA-256, SHA-512. The experiment at each point is the mean of 10 measurements. The experiments of all hash functions are implemented on Core i7-4500U of a CPU of 2.4GHz, running Java 6 under Windows 7. The performance results are presented in Fig. 8.

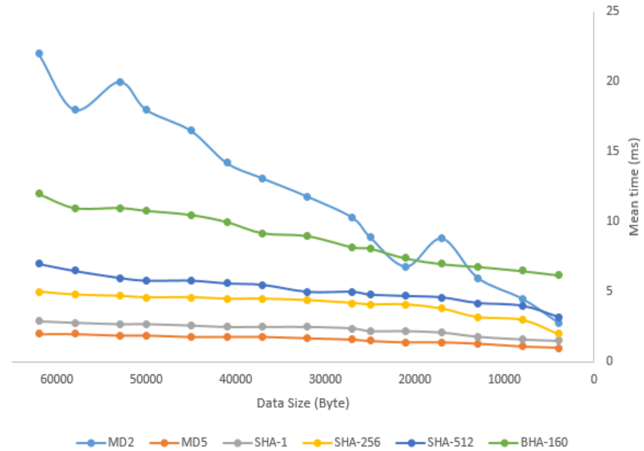


Fig. 8. Performance of Standard Hash Functions Against BHA-160.

The performance result shows that our BHA-160 is in the middle class, where it could outperform MD2 and it is almost close to the performance of other functions. However, we realized that a tradeoff between security and performance exists. Manipulating braids includes performing complex mathematical operations. In addition, the key size of BHA-160 is relatively larger than the key size used by most of the hash functions included in this study.

V. Discussion

Two important parameters should be discussed, they are: the security and the performance of the proposed architecture. In terms of security, the 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 purposes is an 80 strings braid. Mathematically, the braid theory proved to be secure since it is virtually impossible to retrieve one of the multiplied braids after a braid multiplication operation. In terms 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 a minimal operation that needs to be applied on every 192-bit plain text block.

VI. Conclusion

In conclusion, there is a presentation of less security level by bit-operations harsh functions that have a dependence on bits of XORing message as compared to the hash functions that are based on problems that are NP-hard. However, the proposed hash function is proved that it is secure due to the fact that it is based on mathematical problems that are hard to solve hence it is worth to be evaluated. The proposed hash function's internal stages tend to depend on the mapping of bits into braid representation as well as multiplying the resulted braids to each other. These stages form the core of the entire architecture of the hash function that is proposed and they could be able to fulfill the significant features of a hash function that is secured.

REFERENCES

[1] P. Hofmann and B. Schneier, Attacks on Cryptographic Hashes in Internet Protocols, facs.org. November 2005. [Online]. Available: https://tools.ietf.org/html/rfc4270 [Accessed May 20, 2019].

[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.