Comparison of Hash Function Algorithms Against Attacks: A Review

Ali Maetouq, Salwani Mohd Daud, Noor Azurati Ahmad, Nurazean Maarop, Nilam Nur Amir Sjarif, Hafiza Abas
Advanced Informatics Department
Razak Faculty of Technology and Informatics
Universiti Teknologi Malaysia
Kuala Lumpur, Malaysia

***Abstract*—Hash functions are considered key components of nearly all cryptographic protocols, as well as of many security applications such as message authentication codes, data integrity, password storage, and random number generation. Many hash function algorithms have been proposed in order to ensure authentication and integrity of the data, including MD5, SHA-1, SHA-2, SHA-3 and RIPEMD. This paper involves an overview of these standard algorithms, and also provides a focus on their limitations against common attacks. These study shows that these standard hash function algorithms suffer collision attacks and time inefficiency. Other types of hash functions are also highlighted in comparison with the standard hash function algorithm in performing the resistance against common attacks. It shows that these algorithms are still weak to resist against collision attacks.**

***Keywords*—Hash function algorithms; MD5; PRIMEDS160; SHA-1; SHA-2; SHA-3**

I. Introduction

Among the most useful primitives that are crucial for data security is the cryptographic hash function, which offers message authentication, data integrity, and digital signature [1]-[3]. Additionally, it is employed as a core element of cryptographic protocols, secure transactions and crypto-currencies. Fig. 1 presents an output of a fixed length (termed as a message digest or hash code) that uses a one-way function (known as a hash function) with an input of arbitrary length (also termed as a "message" or "plain text") [4].

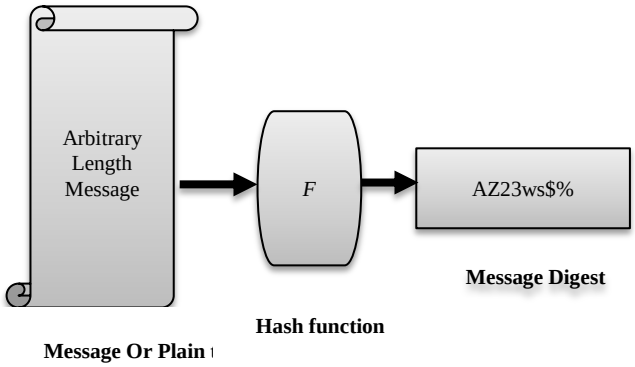

Fig. 1. Hash function.

The mathematical definition of a hash function (H) is defined as follows:

$$H: \{0, 1\}^* \rightarrow \{0, 1\}n \quad (1)$$

Where, {0, 1}* refers to the set of binary elements of any length including the empty string. Meanwhile, {0, 1}n is used to refer to a set of binary elements with length n. Thus, a set of fixed-length binary elements is mapped to arbitrary-length binary elements using the hash function.

The organization of the paper is as follows. In Sections II and III, the basic concepts such as security properties and applications of hash functions are discussed. A literature review on the most popular hash function algorithms is provided in Section IV. Then, the comparison of the standard hash algorithm based on the general properties and common attacks are discussed in Section V. Many researchers have also proposed their own algorithms as discussed in Section VI.

II. Properties of Hash Functions

Several properties of security must be satisfied for cryptographic hash functions [5], [6].

A. Resistance to Collision Attacks

It would be impossible for the attacker to find the same hash value or H(M) for two messages (M, M'). A collision attack happens when a pair of distinct messages having the same hash as shown in Fig. 2. The hash function must have the property of not producing same hash value for different messages.

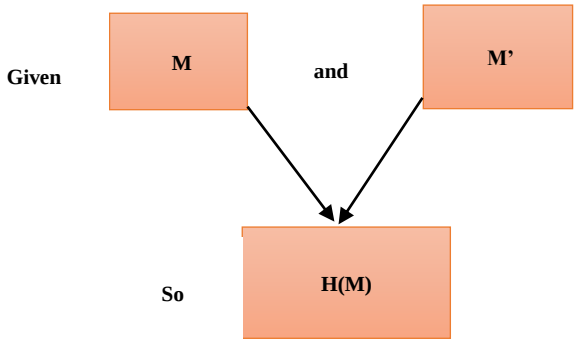

Fig. 2. Collision resistance.

B. Resistance to Pre-Image Attacks

A preimage is a message that hashes to a given value. In a preimage attack, it is usually assumed that at least one message that hashes to the given value exists as shown in

Fig. 3. Therefore, to be resistance to pre-image attacks, one often says that the adversary (also called the attacker) is given y = H(M) for some (randomly chosen) message M, which the attacker does not know. In other words, the attacker should find it is not possible to gain original data (or message (M)) from a given hash value H(M).

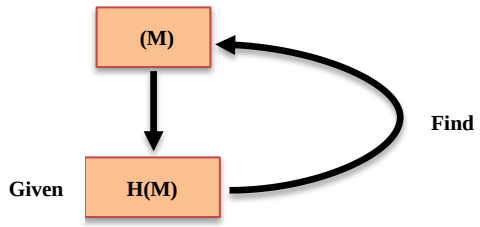

Fig. 3. Preimage resistance.

C. Resistance to Second Pre-Image Attacks

A second preimage is a message that hashes to the same value as a given (randomly chosen) message, called the first preimage. Obviously, the second preimage must be different from the first. Here, we assume that the attacker is also given the hash value of the first preimage. If not, then the attacker can compute it himself. In the latter case the cost of hashing the first preimage is placed on the attacker, which we do not assume here. A brute force preimage attack can also be used to find a second preimage.

One simply ignores the first preimage, except that one may take care not to try a message that is identical to the first preimage. By selecting messages at random, assuming that the domain of the hash function is much larger than the co-domain, the probability of the second preimage being equal to the first is negligible, and therefore we usually ignore this possibility. Due to the above attack, finding a second preimage seems to never be harder than finding a (first) preimage. However, there are artificial constructions that allow preimages to be found in constant time, but which are collisions and second preimage resistant.

Fig. 4 shows that the hash value H(M) could change with the slightest change in message (M). In summary, it should be impossible for an attacker—which has been given a message to obtain the original digest after manipulating it.

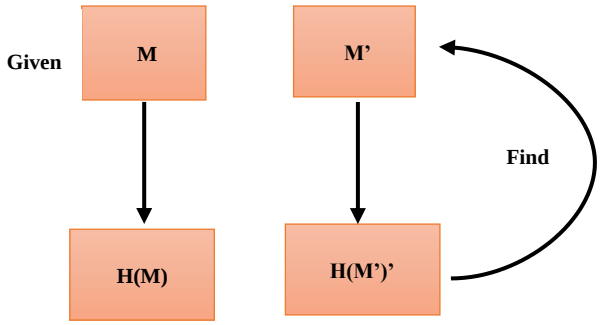

Fig. 4. Second preimage resistance.

Besides these properties, the hash function should also be able to work and calculate the digest for any input message of any size; the hash calculation process must be efficient.

III. Applications of Hash Function

Hash functions are used in many applications such as digital signature, message integrity, and authentication. This section discussed these applications.

A. Digital Signature

This is the first application of a secure hash function, it is a mathematical scheme used to validate the authenticity of the sender, message and signer of the document identity. In cases where it is crucial that an altered document or message is detected, or in any financial transaction, digital signature is commonly implemented. The signature for a document is produced via public and private keys utilized by the Digital Signature. This indicates that without authorization, it is difficult for another person to duplicate the document or message created by the person who had signed it first [7].

B. Message Integrity

Integrity checking is the foremost and fundamental objective of the hash function, which allows the detection of any changes being made to the data. The integrity of a message that is transmitted is checked via the sender, who hashes the message, whereby both hash value and message are sent. The message is generally sent from an insecure line, and only sometimes from a secure one. The received message is hashed from the side of the receiver, who checks the received hash value against the resulting hash value. The preservation of the message depends on whether or not the two hash values match; a match indicates preservation while a mismatch indicates non-preservation. There is a very low possibility that hash value and message are both altered (the hash value of the altered message is the altered hash value).

C. Message Authentication Code (MAC)

In constructions involving the Message Authentication Code (MAC), hash functions are popularly used as building blocks. Verification of identical sent and received messages can be done using the Message Authentication Code. However, only the sender or the recipient can compute the MAC. Therefore, identity verification of a sender to a third party cannot be executed using MAC. A keyed-hash function (which includes a keyed-in addition input to the message) is used to compute the MAC. The key must be kept secret or the operation will fail. Although third parties will not be privy to this key (kept secret from them), the same key must be used from the recipient and sender side. In the process of generating a MAC, any applicable constant string and hash function is used for the sender to input message and key. This is followed with the sending of the message and generated MAC to the receiver. The same hash function and algorithm is used on behalf of the receiver to generate a MAC of the message, so as to eliminate any chances of the message being manipulated. The message indeed would not have been manipulated if the MAC received from the sender matches the MAC generated by the receiver. This provides a simple way for verifying message integrity. To ensure MAC computation efficiency from both, the sender and receiver side, an efficient and high-speed hash function is required [6].

IV. Standard Hash Function Algorithms

The most standard hash functions used today are the dedicated hash, that is, hash functions that are especially designed for hashing purpose only. In this section, we will describe the more popular hash functions.

A. MD5(Message Digest 5)

MD5 is a popular hash function in the MD family, designed by Rives in 1991. This hash function uses the Merkle–Damgard construction. The MD5 algorithm outputs a 128-bit length from an input of an arbitrary length message. However, several attacks have been found on MD5. In 1992, Bore and Bosselaers found collision attacks usually targeting the compression function. In 1996, Dobbertin published the fact that collision attacks targeted MD5. Successful collision attacks were also reported against MD5 in [8]. The improvement of collision attacks on MD5 were also found in previous works [9], [10].

B. RIPEMD-160

It is a well-known hash function in the RIPEMD family, designed by Dobbertin, Bosselaers and Perneel in 1996. It is part of the international standard ISO/IEC10118-3:2004 of dedicated hash functions. It also uses the Merkle-Damgård construction. It produces a message digest length of 160 bits [11]. However, semifree-start collision, preimage and collision attacks on RIPEMD-160 were found in [10].

C. Secure Hash Algorithm (SHA)

Secure Hash Algorithm (SHA) is a group of hash functions published by the National Institute of Standards and Technology as a US Federal Information Processing Standard (FIPS). All of the current SHA algorithms were developed by the NSA:

- **SHA-1:** NIST (1995) developed the Secure Hash Algorithm 1 or SHA-1, which also uses the Merkle–Damgard construction as MD5, and generates a 160-bit message digest for an arbitrary length input message. However, collision attack was also founded against SHA-1 in previous studies [12]-[14]. Therefore, NIST announced the step-by-step elimination of SHA-1 [15].
- **SHA-2:** NIST (2002) added other algorithms to the SHA family with respective hash code lengths of 256, 348, and 512 bits i.e. SHA-256, SHA-384 and SHA-512, respectively. These follow the same structure as MD5 and SHA-1, but are more complex since a nonlinear function is added to the compression function. However, SHA-2 is not preferred to ensure integrity, as it is not as time efficient as SHA-1 [16]. On the other hand, Bitcoin, as the most popular crypto-currency, uses SHA256 for Hashcash which provides security over transactions made between peers in the Bitcoin network. However, SHA-256 has no multi-threading ability, and thus it is not fast enough for transactions [17]. The most recent attacks on SHA-2 have been shown in previous works [18].
- **SHA-3:** After several successful collision attacks which were progressively reduced in complexity (such as MD5, SHA-1 and SHA-2), NIST, in the Federal Register, announced a public competition to develop SHA-3, a completely new hashing algorithm. In 2007, the announcement for the initiative was published. Then, four years later, on October 2nd, 2012, the winner of the competition Keccak, was announced. In 2014, NIST considered SHA-3 as a standard hash function. However, this algorithm is susceptible to first collision-finding attacks [19], [20]. On the other hand, the algorithm shows relatively low software performance compared to other hash functions [21].

V. Comparison of Standard Hash Algorithm

The comparison of the standard hash algorithm based on the general properties, including block size, word size, output size, logical operation, and the number of rounds as shown in Table I. And also common attacks on these algorithms are summarized as illustrated in Table II.

TABLE I. Comparison between Standard Hash Function Algorithms based on Properties

Properties	Name of Algorithm				
	MD5	**RIPEMD -160**	**SHA-1**	**SHA-2 256/512**	**SHA-3 256/512**
Block Size	512 bits	512 bits	512 bits	512/1024 bits	1088/576 bits
Word Size	32 bits	32bits	32bits	32/64 bits	320/320bits
Output Size	128bits	160 bits	160 bits	256/512 bits	1600/1600bits
Rounds	18	80	80	64/80	24/24
Operations	ADD,XOR, AND,OR, NOT, SHIFT	ADD,, ROTATE, XOR,AND, OR,NOT	ADD, XOR AND, OR,NOT, ROTATE.	ADD, XOR, OR, AND SHIFT,, ROTATE	-
Construction	Merkle-Damgard	Merkle-Damgard	Merkle-Damgard	Merkle-Damgard	Sponge

TABLE II. COMMON ATTACKS ON STANDARD HASH FUNCTION ALGORITHMS

Algorithm		Type of attacks	Complexity	References
MD5		Collision	2^{39}	Ref [8]
		Fast Collision	2^{18}	Ref [9]
RIPEM-160		Collision	2^{67}	Ref [22]
		Preimage	$2^{158.91}$	Ref [23]
SHA-1		Collision	$< 2^{69}$	Ref [12]
		Collision	2^{61}	Ref [13]
		Freestart Collision	-	Ref [14]
SHA-2	**256**	Preimage	$2^{255.5}$	Ref [24]
	512	Preimage	$2^{511.2}$	Ref [24]
SHA-3	**256**	Practical Collision and near-Collision	-	Ref [19]
	512	Possibility first Collision	-	Ref [20]

From the above discussion, it is found that most of the popular hash functions from different families suffer from collision attacks and also are not time efficient. As a solution to this problem, researchers proposed other algorithms.

VI. DISCUSSIONS

Many researchers have proposed their own algorithm in order to overcome the above issue as shown in Table III. In this section, the authors have discussed some of the variations in hash function algorithms.

Belfedhal and Faraoun [25] used a variant of the Merkle-Damgard construction basing off on cellular automata to introduce a hash function algorithm producing a 256-bit hash value. Although the algorithm yielded good results for statistical test, it was not tested against collision and preimage attacks.

Li et al. [26] used a dynamic S-box to design a chaotic hash function that produces 128-bit hash values as the final hash code and thus compromising its practicability and flexibility lent via the S-box. One major drawback of this proposed algorithm is that the length of the hash code is not enough to guarantee security against collision or second pre-image attacks.

Abdulah et al. [27] developed a new hash function based on MD5, generating a 224-bit hash value. Perhaps the most serious disadvantage of this development is the time required to produce the message digest, which is as much as the MD5, meaning that the efficiency is very low.

Tur and Javurek [28] used neural network to develop hash function generation, which produced a 128-bit hash value. However, approaches of this kind are very difficult to execute besides having a short hash value.

Ahmad et al. [29] had integrated 2D and 1D chaotic maps in the development of a novel hash function scheme, where 128-bit hash value for an arbitrary length message was generated. Nevertheless, the length of the hash value is short and thus it is not resistant against collision attacks.

TABLE III. COMPARISON OF VARIATIONS OF HASH FUNCTION ALGORITHMS

Author	Year of publication	Advantages	Limitations
Belfedhal and Faraoun [25]	2015	It provides good cryptographic properties such as pseudo- random behavior and sensitivity to the input changes.	It was not tested against attacks that are cryptographic in nature e.g., meet-in-the-middle attacks, collision or birthday attacks.
Wang *et al.* [31]	2015	It provides variable output.	It was not tested against common attacks such as collision.
Li *et al.* [26]	2016	It has good statistical performance and collision resistance.	Any attacker could launch exhaustive collision attacks on the function because the final hash value is 128 bits
Tur and Javurek [28]	2016		Extra modules are still required to enable the proposed system to be used as a real application. It was not tested against attacks that are cryptographic in nature e.g., meet-in-the-middle attacks, collision or birthday
Li and Liu [30]	2016	The confusion and diffusion property of the proposed algorithm hash is good.	Any attacker could launch exhaustive collision attacks on the function because the final hash value is 128 bits.
Ahmad *et al.* [29]	2017	It has great statistical performance. It can generate hash value of length 160,256 or 512 bits.	
Zhang *et al.*[2]	2017	The proposed algorithm satisfies the requirement of statistical performance.	The algorithm is not time efficient to obtain the hash value.

Li and Liu [30] used chaotic mapping that is generalized and parameters that are variable to propose a hash function. In their work, an arbitrary length message was converted to corresponding ASCII values in the process of executing 6-unit iterations that has variable message values and parameters. Towards achieving the final hash value, iteration state values were used to cascade extracted bits. Based on a definition of the birthday attack, 128-bit hash value is not enough to guarantee a secure algorithm.

Wang et al. [31] proposed a hash algorithm, which generated a variable size digest. Their proposed algorithm was an improved version of MD-5 algorithm on the output length. Although the proposed algorithm has a variable size digest, which is good property to increase security, however it still uses the same construction as MD5.

Many researchers have proposed their own hash function algorithm as shown in Table III. Some of them were based on the chaotic design, with some being based on the complex chaotic system, the chaotic neural network, and the chaos tent map. Some of the current hash function designs produce a hash value size of 128 bits only which is not secure enough against collision attacks. Although these algorithms have offered satisfactory statistical performances, they are still weak in resisting collision attacks. Consequently, it is necessary to develop new approaches to hash function algorithm design that is able to prevent attacks effectively in comparison to existing algorithms as they are not sufficient to meet the requirement of latest technologies and security concern.

VII. Conclusion

There are various types of hash functions algorithms used to ensure the integrity and authentication of messages. Some have emerged as the standard, such as MD5, SHA-1, SHA-2 and SHA-3. This paper discusses these algorithms. It was found that most of them are either breakable, or are not time efficient. Also, this paper discusses other hash algorithms which were presented by researchers, but most of them were not tested against attacks that are cryptographic in nature such as collision attacks. Therefore, it can be concluded that a hash function that is efficient and safe, and fulfills application requirements such as data integrity and authenticity, must be designed and made into a priority.

Acknowledgment

We would like to express our gratitude to Ministry of Education (MOE Malaysia) for providing financial support (research grant Q.K130000.2538.19H12) in conducting our study. Our special thanks to Universiti Teknologi Malaysia (UTM) and specifically Advanced Informatics Department in Razak Faculty of Technology and Informatics for realizing and supporting this research work.

References

[1] S. Mohammed, “Secure Hash Design & Implementation Based On Md5 & Sha-1 Using Merkle – Damgard Construction,” Int. J. Adv. Res. Comput. Sci. Technol. (IJARCST 2016) Vol., vol. 4, no. 2, pp. 92–94, 2016.

[2] P. Zhang, X. Zhang, and J. Yu, “A Parallel Hash Function with Variable Initial Values,” Wirel. Pers. Commun., vol. 96, no. 2, pp. 2289–2303, 2017.

[3] R. Haddaji, “Comparison of Digital Signature Algorithm and Authentication Schemes for H . 264 Compressed Video,” Int. J. Adv. Comput. Sci. Appl., vol. 7, no. 9, pp. 357–363, 2016.

[4] R. P. Arya, U. Mishra, A. Bansa, and W. S. Email, “A Survey on Recent Cryptographic Hash Function Designs,” internatonal J. emrging trends Technol. Comput. Sci., vol. 2, no. 1, pp. 1–6, 2013.

[5] N. Garg and N. Wadhwa, “Design of New Hash Algorithm with Integration of Key Based on the Review of Standard Hash Algorithms,” Int. J. Comput. Appl. (0975 – 8887), vol. 100, no. 8, pp. 11–18, 2014.

[6] M. Ghebleh and A. Kanso, “A structure-based chaotic hashing scheme,” Nonlinear Dyn., pp. 27–40, 2015.

[7] A. A. Alkandari, I. Al-shaikhli, and A.Alahmad, “Cryptographic Hash Function : A High Level View,” Int. Conf. Informatics Creat. Multimed. Cryptogr., pp. 129–135, 2013.

[8] X. Wang and H. Yu, “How to Break MD5 and Other Hash Functions,” EUROCRYPT 2005, LNCS 3494, pp. 19–35, 2005.

[9] T. Xie, F. Liu, and D. Feng, “Fast Collision Attack on MD5,” pp. 1–12, 2013.

[10] V. Chiriaco, A. Franzen, and R. Thayil, “Finding Partial Hash Collisions by Brute Force Parallel Programming,” in 37th IEEE Sarnoff Symposium 2016, Newark, NJ, September 19-21, 2016, vol. 5, pp. 1–6.

[11] F. Mendel, T. Peyrin, and M. Schl, “Improved Crypanalysis of Reduced RIPEMD-160,” Int. Conf. Theory Appl. Cryptol. Inf. Secuerty., pp. 484–503, 2013.

[12] X. Wang, Y. L. Yin, and H. Yu, “Finding Collisions in the Full SHA-1,” Int. Assoc. Cryptologic Res. 2005, no. 90304009, pp. 17–36, 2005.

[13] M. Stevens, “New Collision Attacks on SHA-1 Based on Optimal Joint Local-Collision Analysis,” Int. Assoc. Cryptologic Res. 2013, pp. 245–261, 2013.

[14] M. Stevens, P. Karpman, and T. Peyrin, “Freestart collision for full SHA-1,” EUROCRYPT 2016., vol. 2012, pp. 1–21, 2016.

[15] K. Wu, Y. Li, L. Chen, and Z. Wang, “Research of Integrity and Authentication in OPC UA Communication Using Whirlpool Hash Function,” Appl. Sci. ISSN2076-3417, pp. 446–458, 2015.

[16] S. Verma and G. Prajapati, “Robustness and Security Enhancement of SHA with Modified Message Digest and Larger Bit Difference,” in 2016 Symposium on Colossal Data Analysis and Networking (CDAN), 2016, pp. 0–4.

[17] G. Meliolla, K. A. Nugroho, and F. I. Hariadi, “Implementation of Hash Function on Embedded- System Platform using Chaotic Tent Map Algorithm,” in Electronics and Smart Devices (ISESD), International Symposium on, 2016, pp. 179–183.

[18] N. Kishore and B. Kappor, “Attacks on and Advances in Secure Hash Attacks on and Advances in Secure Hash Algorithms,” IAENG Int. J. Comput. Sci., no. September, 2016.

[19] I. Dinur, O. Dunkelman, and A. Shamir, “New attacks on Keccak-224 and Keccak-256,” pp. 1–27, 2011.

[20] I. Dinur, O. Dunkelman, and A. Shamir, “Collision Attacks on Up to 5 Rounds of SHA-3 Using Generalized Internal Differentials,” no. 827, pp. 1–30, 2013.

[21] D. Kim, D. H. B, J. Lee, and W. Kim, “LSH : A New Fast Secure Hash Function Family,” Springer Int. Publ. Switz. 2015, pp. 286–313, 2015.

[22] F. Liu, F. Mendel, and G. Wang, “Collisions and Semi-Free-Start Collisions for Round-Reduced RIPEMD-160,” in 23rd Annual International Conferences on Theory and Application of Cryptology and Information Security,ASIACRYPT2017, 2017, pp. 1–32.

[23] Y. Shen and G. Wang, “Improved Preimage Attacks on RIPEMD-160 and HAS-160,” KSII Trans. Internet Inf. Syst., vol. 12, no. 2, pp. 727–746, 2018.

[24] D. Khovratovich, C. Rechberger, and A. Savelieva, “Bicliques for Preimages : Attacks on Skein-512 and the SHA-2 family,” pp. 244–263, 2012.

[25] A. E. Belfedhal and K. M. Faraoun, "Building Secure and Fast Cryptographic Hash Functions Using," J. Comput. Inf. Technol., pp. 317–328, 2015.

[26] Y. Li, G. Ge, and D. Xia, "Chaotic hash function based on the dynamic S-Box with variable parameters," Nonlinear Dyn., vol. 84, no. 4, pp. 2387–2402, 2016.

[27] H. S. Abdulah, M. A. H. Al-rawi, and D. N. Hammod, "Message Authentication Using New Hash Function," J. Al-Nahrain Univ., vol. 19, no. 3, pp. 148–153, 2016.

[28] M. Tur and M. Javurek, "Hash Function Generation by Neural Network," in 2016 New Trends in Signal Processing (NTSP), 2016.

[29] M. Ahmad, S. Khurana, S. Singh, and H. D. Alsharari, "A Simple Secure Hash Function Scheme Using Multiple Chaotic Maps," 3D Res., vol. 8, no. 2, pp. 1–15, 2017.

[30] Y. Li, X. Li, and X. Liu, "A fast and efficient hash function based on generalized chaotic mapping with variable parameters," Neural Comput. Appl., vol. 28, no. 6, pp. 1405–1415, 2016.

[31] M. Wang and Y. Zhen Li, "Hash Function with Variable Output Length," in 2015 International Conference on Network and Information Systems for Computers, 2015, pp. 3–6.