# Constructions and Attacks on Hash Functions

Zeyad A. Al-Odat, and Samee U. Khan
*North Dakota State University*
Fargo, ND, USA
Emails: zeyad.alodat@ndsu.edu, samee.khan@ndsu.edu

***Abstract*—This paper presents a study about the construction models that are used to build hash functions. The study covers Merkle-Damgåd, Sponge, HAIFA, Wide-Pipe, and Parallel-Tree constructions. Moreover, attacks on hash functions are also discussed in this paper. The major attacks on hash functions are presented with counter-attack methodology from the literature. This work helps the researchers in the area of cryptography hash functions to avoid any possibility of security threats in future designs.**

***Keywords*—Attack; Hash; Cryptography; Security.**

## I. Introduction

Secure Hash Algorithms are the most known cryptography primitives that are used for integrity and authenticity. A secure hash algorithm is used to convert data (from different domains) into a fixed-length bit string [1]. This length depends on the type of utilized hash algorithm. For instance, the length of the output that the MD5 hash function generates is equal to 128-bit.

The most known secure hash algorithms follow two types of constructions, the Merkle-Damgård and sponge constructions. The Merkle-Damgård was utilized by the majority of hash functions including (MD4, MD5, SHA-1, and SHA-2). The MD4 was published in 1989 by Ronald Rivest [2]. Then, for security reasons, the MD4 was replaced by the MD5 [3]. In 1995, the National Institute of Standards and Technology (NIST) announced the SHA-1 hash standard [4], which was more secure than the MD5. After successful attacks on the MD4, MD5, and SHA-1, the NIST announced the SHA-2 following the same construction ( Merkle-Damgård ). The SHA-2 supports six different fixed-length outputs SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256 [5].

To prepare a ready replacement for the SHA-2 standard, the NIST held a competition to chose the next hash standard. Out of 61 submissions, 51 candidates were selected for the first round. Then they were reduced to 14 in the second round. The last round comprised five candidates (Blake [6], JH [7], Grøstl [8], Skien [9], and Keccak [10]). The competition occluded by announcing the Keccak as the new hash standard, which takes the name of the SHA-3 standard. Unlike the other standards, Keccak follows the sponge construction. However, Keccak supports the same fixed-length outputs that the SHA-2 supports. Moreover, Keccak presents SHAKE128 and SHAKE256 for variable-length hash [11].

The mean of a strong hash function is the one that achieves the security requirements. In the case of fixed-length hash functions, five security requirements are considered: pre-image resistance, $2^{nd}$-pre-image resistance, and collision resistance. The other two properties, MAC and PRF, are considered when the Hashed Message Authentication Code (HMAC) is present, where the hash function is employed with a secret key.

However, the security requirements were exposed to attacks that made some of the hash standards weak. For instance, the SHA-1 hash function was exposed to a collision attack by Wang *et al.* [12]. In essence, the mean of a successful attack is the one that breaches the security requirements of any hash function.

In this paper, a study about the construction of hash functions and their corresponding attacks is presented. In this work, we show the various construction models that are used to build hash functions. Moreover, different types of attacks are discussed and the possible ways to overcome them.

The rest of paper is organized as follows: Section II presents the various construction models to build a hash function. Section III discusses the different types of attacks that threaten the hash functions. The researchers' efforts to counter the hash functions' attacks are presented in Section IV. Conclusions in Section V.

## II. Construction Models

### *A. The Merkle-Damgård Construction*

The need for a structured model for use in the digital signature formulation was needed. The Merkle-Damgård construction (which takes the name of two authors of two different works [13], [14], in 1989) describes constructing a hash primitive based on a compression function $F$. Figure 1 shows the general structure of the Merkle-Damgård construction, where a message ($M$) is divided into blocks and processed sequentially using a compression function ($F$). This structure requires initial values (IV) that are used to process the first block. Then, the output of each block is used to process the next block. The final output is produced after processing the last block of the message.

MD4 and MD5 formed the base for the following hash functions that come after, which have the same Merkle-Damgård construction. We list examples of these functions as follows:

- **SHA-0**: Which was designed by the National Security Agency (NSA), in 1993 [15].
- **SHA-1**: Which was placed as a replacement to the SHA-0 after undisclosed security concerns, in 1995 [16].

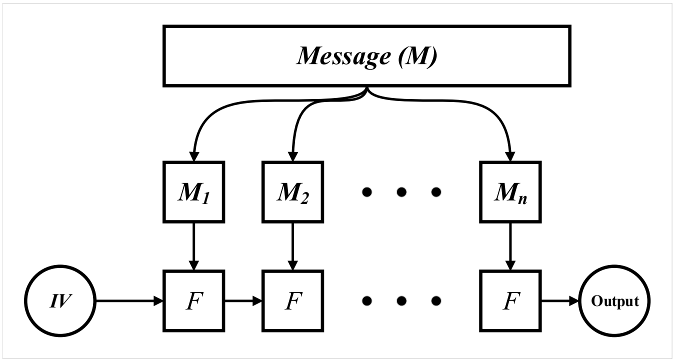

Fig. 1. The general structure of the Merkle-Damgård construction

- **SHA-2**: Which was designed starting from 2002 with four different flavors SHA-224, SHA-256, SHA-384, and SHA-512 [17]–[19].
- **RIPEMD**: Which stands for RACE Integrity Primitives Evaluation Message Digest, which was published in 1995 [20].
- **RIPEMD-160**: RIPEMD-160 is a strengthened version of RIPEMD with a 160-bit hash result. Which was published in 1996 [21].

## *B. The Sponge Construction*

Unlike the previous standards (SHA-1 and SHA-2), the SHA-3 relies mainly on the Sponge structure [22], [23], where the data are absorbed in and squeezed out to generate the hash output. The Sponge construction comprises two phases, absorb and squeeze. After a message is preprocessed, it is divided into equal size of blocks ($p_i$). The values bit-rate ($r$) and capacity ($c$) reflect the permutation level that the function $f$ will perform. Given an input length $N$ and an output length $d$, the output $Z = sponge[f, r](N, d)$ is generated with $d$ bit length. The permutation function $f$ is applied to the output $Z$ until the required number of bits for the $d$ output is produced.

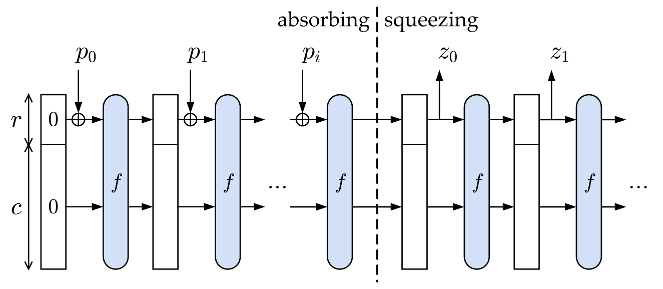

Fig. 2. The general structure of Sponge construction

## *C. HAIFA*

The HAsh Iterative FrAmework (HAIFA) construction was developed in 2006 by Biham *et al.* [24]. The construction of HAIFA is depicted in Figure 3. An optional `Salt` is added to the iterative computations of the classical Merkle-Damgård construction. Moreover, HIAFA keeps track of the number of hashed bits after each block computation. Researchers proved the strength of HAIFA construction against $2^{nd}$-preimage and collision attacks if the compression function performs optimally, as shown in [25].

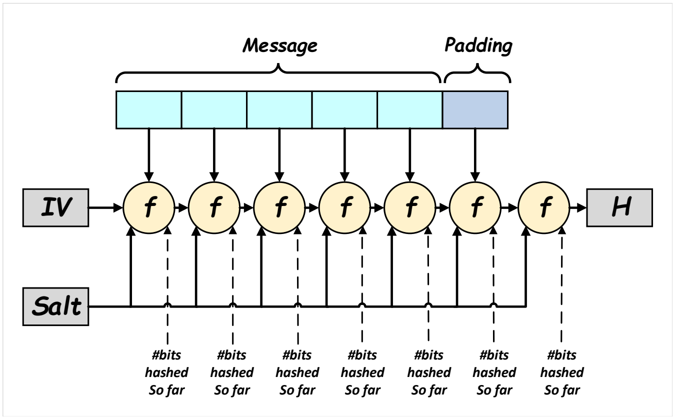

Fig. 3. HAIFA construction

## *D. Wide-Pipe Construction*

The Wide-Pipe construction behaves as the HAIFA construction without the `Salt` variables. The Wide-Pipe construction maintains two levels of compression function computations, as shown in Figure 4. The message compression function ($f_1$) that processes the message blocks. The second level performs the output compression function ($f_2$) to produce the output hash. The resulting hash ($H$) is produced after the transformation of $f_1$ compression function using $f_2$ compression function.

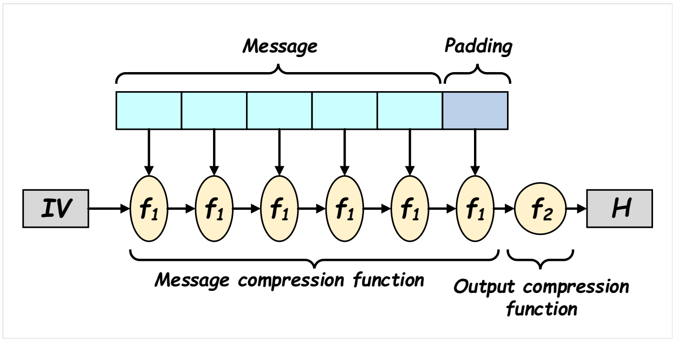

Fig. 4. Wide-Pipe construction

## *E. Parallel-Tree construction*

The Parallel-Tree construction works by processing message blocks in parallel and performs a reduction algorithm to the generated output. A message with a long length takes time to be processed using the iterative structure. In this model, the message is processed in parallel with reduced time compared to other structures. However, vulnerabilities of this model have not been studied, which makes this model away from attentions [26].

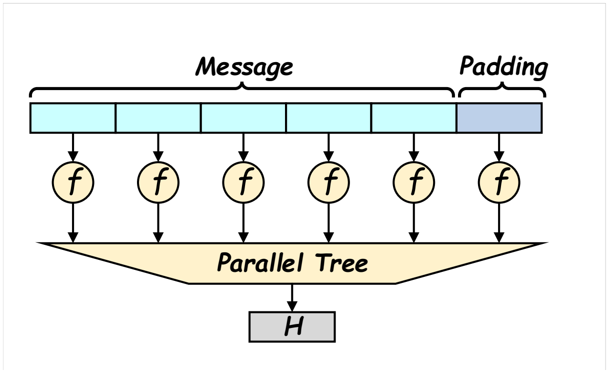

Fig. 5. Parallel-Tree construction

## F. Security Requirements

A hash function is considered as secure if it supports the security requirements of Pre-image, $2^{nd}$-Pre-image, Collision resistance, MAC, and PRF [27]–[29].

1) **Pre-image**: Which is based on the problem of finding a pre-image $M$ for a given hash $h$. Pre, which stands for *pre-image resistance*, requires that given a randomly chosen hash function $f$ and a uniformly randomly chosen hash $h$ it is hard to find a pre-image $M \in f^{-1}(h)$.

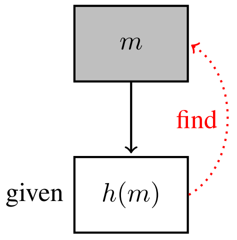

Fig. 6. Pre-image

2) **Sec-Pre-image**: Which is based on the problem of finding another message $M$' that hash the same hash $h$ as a given message $M$. *Sec-Pre-image* requires that the function $f$ and message $M$ to be uniformly and randomly chosen.

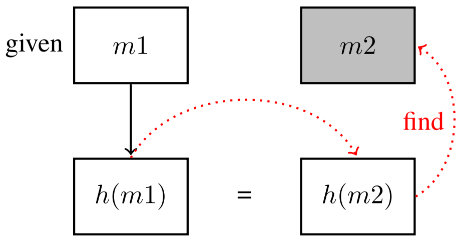

Fig. 7. Second Pre-image

3) **Collision**: Which is the property that finding two messages $M$ and $M$' that have the same hash $h$ for the same chosen $f$. The hash function $f$ must be strong enough to resist any possibility of a collision. Unfortunately, this property was the first to be broken.

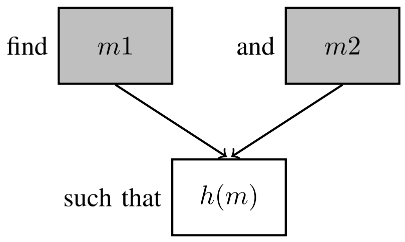

Fig. 8. Collision

4) **MAC**: Which stands for Message Authentication Code, is the property that canvasses the hash function $f$ as a keyed hash function ($f_k$) where $k$ is a randomly chosen secret key that is associated with $f$. The MAC property requires that the dilemma of finding a message $M$ and the hash $h$ with the presence of secret key $k$, is difficult to achieve.
5) **PRF**: Which stands for Pseudo-Random Function, is the property that the problem of determining the actual function $f_k$ for a randomly chosen $k$ with a randomly chosen hash $h$ must be difficult.

## III. Attacks on Secure Hash Algorithms

A successful attack on the secure hash algorithms is the attack that can break one of the security requirements. There exists a general attack that breaks a security requirement of the secure hash functions. A hash function is considered as *broken* when there is an explicit attack that is faster than the general attack for one of the security requirements of a hash function. With the increase of the computational power, all secure hash functions are vulnerable to the general attack due to the small bit length that is generated by the hash functions.

### A. Brute force attack

The most known general attack to break the security requirements (Pre and $2^{nd}$-Pre) is a brute force attack. The brute force attack is applied to Pre and $2^{nd}$-Pre by computing $f(M')$ for a

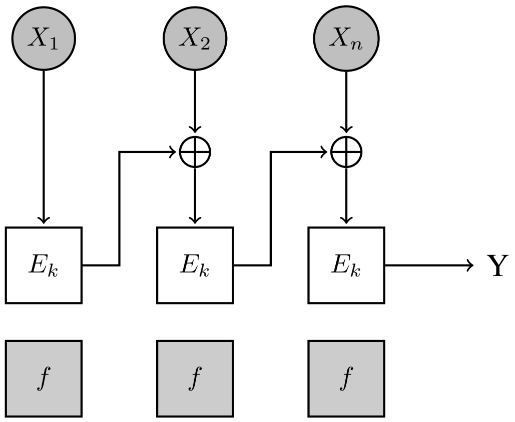

Fig. 9. Pseudo Random Function.

randomly chosen message $M'$ until $M'$ is found and $M' \neq M$. In this case, this attack is considered as infeasible in reality because it requires massive hash computations ($\approx 2^{100}$ hash computations).

### B. Birthday paradox

A general attack based on the birthday paradox is better than the brute force attack. The birthday paradox is based on a counter-intuitive basis, where for groups of as less as 23 people there exists a chance of one half to find two people that were born on the same day [30]. Assuming that all birthdays are equally likely and neglecting leap years, for 23 independent possibilities the success probability is $\frac{23}{365} \approx 0.06$, where there are $\frac{23 \times 22}{2} = 263$ pairs of people. Then the overall success probability is 0.5 ( noting that the pairs are not independently distributed, therefore 0.5 doesn't equal $\frac{253}{365} \approx 0.7$). For a hash function of hash size $N$, the birthday paradox attack succeeds after $\sqrt{\pi/2} \times 2^{N/2}$ computations of the hash function [**?**]. For instance, for a hash size of $N = 128$ bits, which is the hash size of the MD5 hash function, a collision can be found with approximately $2^{64} \approx 22 \times 10^{18}$ computations, which is in reach of a high-performance computer.

### C. Differential attack

The most successful attack on the Merkle-Damgård hash functions is the differential attack. The differential attack works by looking at two different evaluations of a hash function. These evaluations are examined at the same time by computing the difference between them and analyzes the relationship between the difference and the next evaluations' differences. This attack is used to generate a successful collision attack. The differential attack was first used against Data Encryption Standard (DES) [31], [32] based on the XOR difference. Then it was generalized to be used with the modular difference to break many hash standards. Wang *et al.* were able to break several hash standards (MD4, MD5, SHA-0, HAVAL-128, and RIPEMD) using the modular differential attack [12], [33], [34]. Because of the collision attack against Merkle-Damgård hash functions, the National Institute of Standards and Technology (NIST) held a competition to find a new hash function SHA-3 that will be used in the future. This competition concluded by announcing Keccak as the new SHA-3 standard [35].

*1) Same-Prefix Collision Attack:* The collision attack targets the Merkle-Damgård hash functions. Particularly, the hash standards before the SHA-2. Due to the iterative nature of the Merkle-Damgård construction and the fixed size output hash, the collision attack can be structured according to predefined inputs. Figure 10 shows the structure of the same-prefix collision attack. Two different messages (Message1, Message2) have the same prefix ($P$), suffix ($S$) and initialized with the same Initial Hash Value (IHV). Then a pair of two different blocks ($C$, $C^*$) can be computed such that $SHA(P\|C\|S) = SHA(P\|C^*\|S)$, where $IHV_n$ is the Intermediate Hash Value after processing the messages.

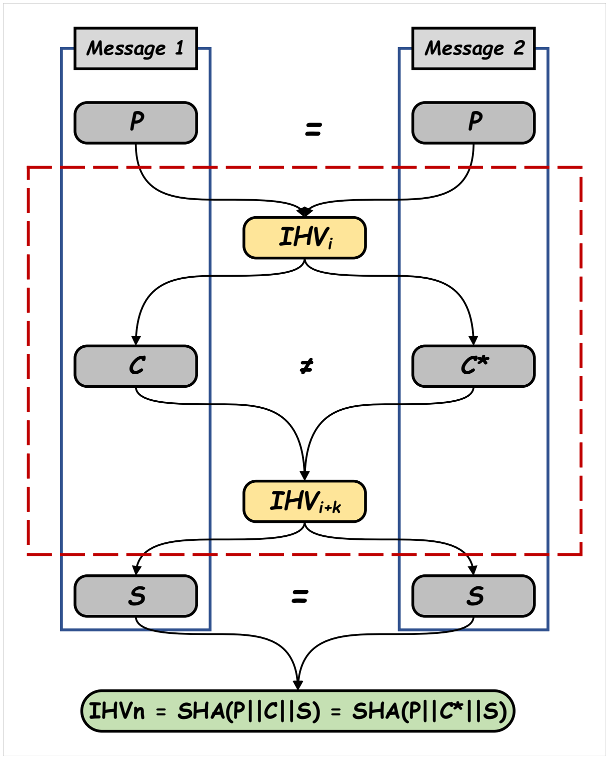

Fig. 10. Same prefix collision attack

*2) Chosen-Prefix Collision Attack:* Stevens *et al.* improved the same-prefix collision attack into a chosen-prefix by formulating two different prefixes with two different messages. However, this approach requires more internal blocks to produce successful collision [36].

Figure 11 shows the structure of the Chosen-Prefix collision attack. This attack succeeds because of the iterative nature of the Merkle-Damgård construction. Two different messages (Message1, Message2) started with two different prefixes ($P$, $P^*$). These prefixes may contain different length strings, then both messages are padded using blocks $A$ and $A^*$ to make them equal in size. According to predefined conditions, a birthday search is applied to find two blocks $B$ and $B^*$ such that $P\|A\|B$ and $P^*\|A^*\|B^*$ have the same size and produce two different intermediate hash values ($IHV_i$ and $IHV_i^*$) that have predefined structures. Finding the proper predefined structures leads to generate two near-collision blocks ($NC$ and $NC^*$) that produce the same $IHV_{i+k}$. This attack requires that both messages have the same suffix ($S$). Finally, the successful chosen-prefix collision attack maintains $SHA(P\|A\|B\|S) = SHA(P^*\|A^*\|B^*\|S)$.

### D. length extension attack

Length extension attack targets the Merkle-Damgård hash functions. Unlike the collision attack, the length extension succeeded to expose the SHA-2 hash function. This attack works with the aligning to Figure 12. A message $M$ is divided into $n$ equal size blocks to compute $H(M)$. according to the Merkle-Damgård construction, the hash value is computed by

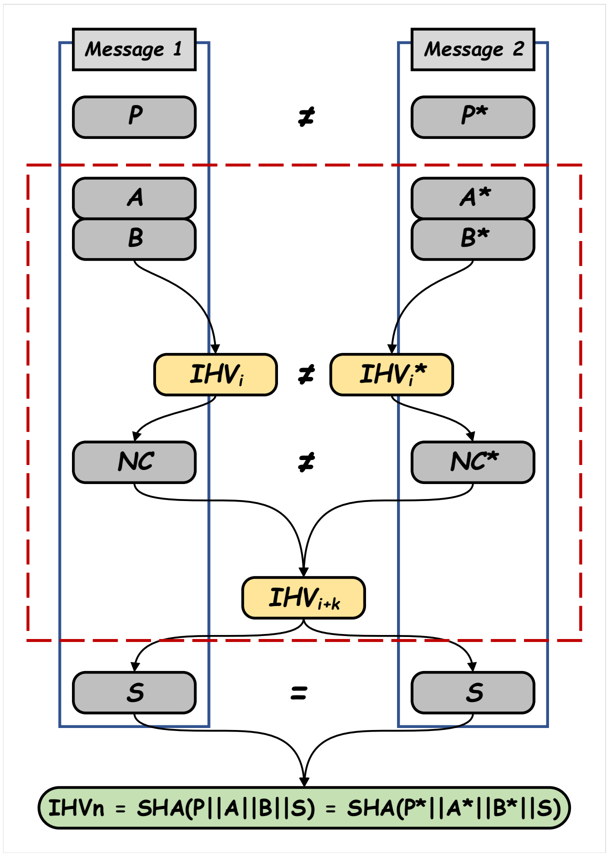

Fig. 11. Chosen-Prefix collision attack

processing the blocks sequentially. Therefore, the hash output ($H(M)$) is produced after processing block $m_n$. An attacker adds extra block ($m_{m+1}$) after the last block and performs the hash computations to produce $H(M')$.

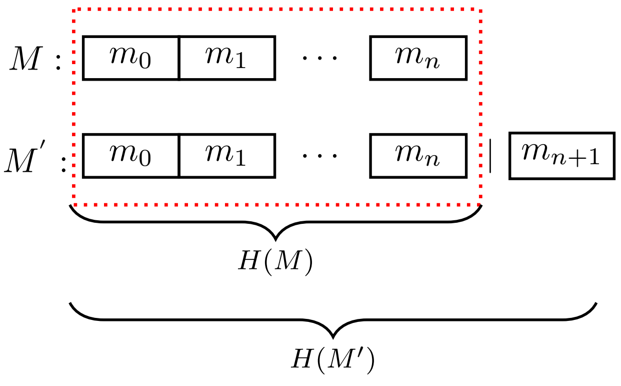

Fig. 12. length extension attack

## IV. Counter-Attack

The mean of a strong hash function is reflected by the strength of the function against breaching the security requirements. As stated by Kaminsky *et al.*, powerful cryptography primitive is the one that produces a random output during the internal computations [37]. However, the high computational nature of the hash functions makes the randomness analyses difficult because it requires a large scale environment for testing.

The counter-attack efforts to overcome the security breaches of the Merkle-Damgård functions were focusing on the collision and length extension attacks because they are the only attacks that give feasible examples.

To overcome the collision attack, Stevens *et al.* presented a model to detect the collision before it takes place [38]. The work supports the MD5 and SHA-1 hash functions. The proposed design works by generating a sibling message from a given one, then compares the Intermediate Hash Value (IHV) of both messages to check for a collision possibility.

Instead of generating a sibling message, a collision detection approach based on a two-block collision was presented in [39]. The proposed work employs the backward expansion equation to generate the Initial Hash Value ($IHV_0^{'}$) of a sibling message, not the message itself. Then compare the $IHV_0$ values of the original message and the generated $IHV_0^{'}$. If both values are equal, then a collision has been detected.

As the collision and length extension attacks target the Merkle-Damgård hash standards, an improved hash function that uses a different construction model was presented in [40]. The work employs the sponge structure to construct a multi-mode mode hash function that supports the MD5 and SHA-1 standards. The design uses a padding method different from the Merkle-Damgård padding method. The new padding method helps to strengthen the hash function against the length extension attack. On the other hand, the sponge structure is strong against the collision attack.

Generally, to design a hash function that is strong enough against any possible attack, the length and the function manipulators of the hash function need to be generated with a random pattern [41], such that:

$$\binom{t}{2}\left(\frac{1}{2^n} - \frac{1}{2^{n+1}} \approx O\binom{t^2}{2^n}\right),$$

where $t$ represents the sufficient number of random messages such that $t \approx \sqrt{2^n}$ for a hash function of output length $n$.

## V. Conclusions

The general construction models are covered in this paper. The work presents five constructions (Merkle-Damgård Sponge, HAIFA, Wide-Pipe, and Parallel-Tree) that were used to build hash functions. The study presents the main attacks on hash functions and the efforts to overcome these attacks. The mean of a successful hash function is represented by the strength of it against any possibility of an attack.

In the future, the study will be extended to cover all cryptography protocols and their corresponding attack and counter-attack paradigms.

## Acknowledgments

The work of Samee U. Khan was supported by (while serving at) the National Science Foundation. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## References

[1] Zeyad Al-Odat, Eman Al-Qtiemat, and Samee Khan. A big data storage scheme based on distributed storage locations and multiple authorizations. In *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pages 13–18. IEEE, 2019.

[2] Ronald L Rivest. Md4 message digest algorithm. 1990.

[3] Ronald Rivest. The md5 message-digest algorithm. 1992.

[4] Patrick Gallagher and Acting Director. Secure hash standard (shs). *FIPS PUB*, 180:183, 1995.

[5] Advance Encryption Standard. Federal information processing standard (fips) publication 197. *National Bureau of Standards, US Department of Commerce, Washington, DC*, 2001.

[6] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C-W Phan. Sha-3 proposal blake. *Submission to NIST*, 92, 2008.

[7] Hongjun Wu. The hash function jh. *Submission to NIST (round 3)*, 6, 2011.

[8] Praveen Gauravaram, Lars R Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S Thomsen. Grøstl-a sha-3 candidate. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009.

[9] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. The skein hash function family. *Submission to NIST (round 3)*, 7(7.5):3, 2010.

[10] B Guido, D Joan, P Michaël, and VA Gilles. The k sha-3 submission. 2011.

[11] Penny Pritzker and Patrick D Gallagher. Sha-3 standard: permutation-based hash and extendable-output functions. *Information Tech Laboratory National Institute of Standards and Technology*, pages 1–35, 2014.

[12] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full sha-1. In *Crypto*, volume 3621, pages 17–36. Springer, 2005.

[13] Ralph C Merkle. One way hash functions and des. In *Conference on the Theory and Application of Cryptology*, pages 428–446. Springer, 1989.

[14] Ivan Bjerre Damgård. A design principle for hash functions. In *Conference on the Theory and Application of Cryptology*, pages 416–427. Springer, 1989.

[15] Elaine B Barker. Secure hash standard (shs). Technical report, 1993.

[16] PUB FIPS. 180-1. secure hash standard. *National Institute of Standards and Technology*, 17:45, 1995.

[17] FIPS PUB NIST and PUB FIPS. 180-2. *Secure Hash Standard*, 2002.

[18] NIST FIPS Pub. 180-3: Federal information processing standards publication, secure hash standard (shs). Technical report, Technical report, National Institute of Standards and Technology, 2008.

[19] PUB FIPS. 180-4. *Secure hash standard (SHS)," March*, 2012.

[20] Antoon Bosselaers and Bart Preneel. *Integrity Primitives for Secure Information Systems: Final Ripe Report of Race Integrity Primitives Evaluation*. Number 1007. Springer Science & Business Media, 1995.

[21] Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. Ripemd-160: A strengthened version of ripemd. In *International Workshop on Fast Software Encryption*, pages 71–82. Springer, 1996.

[22] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The keccak sha-3 submission. *Submission to NIST (Round 3)*, 6(7):16, 2011.

[23] Bertoni Guido, Daemen Joan, P Michaël, and VA Gilles. Cryptographic sponge functions, 2011.

[24] Eli Biham and Orr Dunkelman. A framework for iterative hash functions—haifa. Technical report, Computer Science Department, Technion, 2007.

[25] Elena Andreeva, Bart Mennink, and Bart Preneel. Security reductions of the second round sha-3 candidates. In *International Conference on Information Security*, pages 39–53. Springer, 2010.

[26] Masumeh Damrudi and Norafida Ithnin. Parallel rsa encryption based on tree architecture. *Journal of the Chinese Institute of Engineers*, 36(5):658–666, 2013.

[27] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *International workshop on fast software encryption*, pages 371–388. Springer, 2004.

[28] Mihir Bellare and Thomas Ristenpart. Multi-property-preserving hash domain extension and the emd transform. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 299–314. Springer, 2006.

[29] Mihir Bellare and Thomas Ristenpart. Hash functions in the dedicated-key setting: Design choices and mpp transforms. In *International Colloquium on Automata, Languages, and Programming*, pages 399–410. Springer, 2007.

[30] Mario Cortina Borja and John Haigh. The birthday problem. *Significance*, 4(3):124–127, 2007.

[31] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. *Journal of CRYPTOLOGY*, 4(1):3–72, 1991.

[32] Eli Biham and Adi Shamir. Differential cryptanalysis of the full 16-round des. In *Annual International Cryptology Conference*, pages 487–496. Springer, 1992.

[33] Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. In *Eurocrypt*, volume 3494, pages 19–35. Springer, 2005.

[34] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient collision search attacks on sha-0. In *Annual International Cryptology Conference*, pages 1–16. Springer, 2005.

[35] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 313–314. Springer, 2013.

[36] Marc Stevens, Arjen K Lenstra, and Benne De Weger. Chosen-prefix collisions for md5 and applications. *International Journal of Applied Cryptography*, 2(ARTICLE):322–359, 2012.

[37] Alan Kaminsky. Testing the randomness of cryptographic function mappings. *IACR Cryptology ePrint Archive*, 2019:78, 2019.

[38] Marc Stevens. Counter-cryptanalysis. In *Annual Cryptology Conference*, pages 129–146. Springer, 2013.

[39] Zeyad Al-Odat, Mazhar Ali, and Samee U Khan. Mitigation and improving sha-1 standard using collision detection approach. In *2018 International Conference on Frontiers of Information Technology (FIT)*, pages 333–338. IEEE, 2018.

[40] Zeyad Al-Odat and Samee Khan. The sponge structure modulation application to overcome the security breaches for the md5 and sha-1 hash functions. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 811–816. IEEE, 2019.

[41] Goldreich Oded. Foundations of cryptography: Volume 2, basic applications. 2009.