# Broad View of Cryptographic Hash Functions

Mohammad A. AlAhmad[1], Imad Fakhri Alshaikhli[2]

[1]Department of Computer Science, International Islamic University of Malaysia, 53100 Jalan Gombak Kuala Lumpur, Malaysia,

[2]Department of Computer Science, International Islamic University of Malaysia, 53100 Jalan Gombak Kuala Lumpur, Malaysia

***Abstract***

Cryptographic hash function is a function that takes an arbitrary length as an input and produces a fixed size of an output. The viability of using cryptographic hash function is to verify data integrity and sender identity or source of information. This paper provides a detailed overview of cryptographic hash functions. It includes the properties, classification, constructions, attacks, applications and an overview of a selected dedicated cryptographic hash functions.

***Keywords****-cryptographic hash function, construction, attack, classification, SHA-1, SHA-2, SHA-3.*

## 1. INTRODUCTION

A cryptographic hash function $H$ is an algorithm that takes an arbitrary length of message as an input $\{0, 1\}^*$ and produce a fixed length of an output called message digest $\{0, 1\}^n$ (sometimes called an *imprint, digital fingerprint, hash code, hash result, hash value, or simply hash*). Cryptographic hash functions play a fundamental role in modern cryptography practical applications, for example, digital signature [1,2], digital timestamp [3], message authentication code (or MAC) [4], public key encryption [5], tamper detection of files and many others. This versatility earned them the nickname "Swiss army knife of cryptography".

## 2. CLASSIFICATION, PROPORITIES, CONSTRUCTIONS AND ATTACKS OF HASH FUNCTIONS

### *A.* Classification of Hash Functions

Cryptographic hash functions can be classified as unkeyed hash functions and keyed hash functions as Figure 1 shown. Unkeyed hash functions that accepts a variable length message as a single input and produce a fixed hash digest, H: $\{0,1\}^*\rightarrow\{0,1\}^n$. It is also known as modification detection codes (MDCs). Where, keyed hash functions accept a variable length message and a fixed length secret key as two inputs to the hash function design to produce a fixed length hash digest, $H_K$: $\{0,1\}^k\times\{0,1\}^*\rightarrow\{0,1\}^n$. It is also known as message authentication codes (MACs). Unkeyed hash function is further classified into one-way hash function (OWHF), collision resistant hash function (CRHF), universal one way hash function (UOWHF) [2].

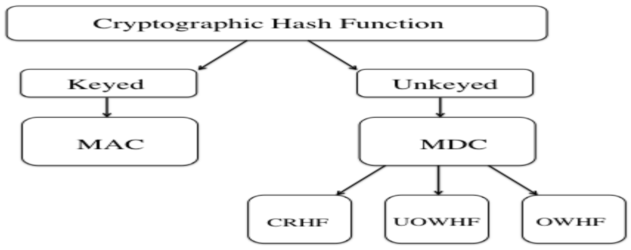

Figure 1. Classification of cryptographic hash function

CRHF usually deals with longer length hash values. An unkeyed hash function is a function for a fixed positive integer n which has, as a minimum, the following two properties:

1) Compression: h maps an input x of arbitrary finite bit length, to an output h(x) of fixed bit length n.
2) Ease of computation: given h and an input x, h(x) is easy to compute.

Modification Detection Codes are classified as follows:

1) One-way hash function is a hash function where finding an input which hashes to a pre-specified hash digest is difficult.
2) A collision resistant hash function is a hash function where finding any two inputs having the same hash digest is difficult.
3) In a universal one-way hash function, for randomly chosen input x, key k and the function $H_k$, it is hard to find y = x such that $H_k(x) = H_k(y)$ [2].

A keyed hash function is a function whose specific purpose is called message authentication code (MAC) algorithms as shown in Figure 1.Keyed hash functions should satisfy the following two properties:

1) Compression: $H_k$ maps an input x of arbitrary finite bit length, to an output $H_k(x)$ of fixed bit length n.
2) Ease of computation: for a known function $H_k$, given a value k and an input x, $H_k(x)$ is easy to compute. The result is called MAC value [2].

Apart from the classification of keyed and unkeyed hash functions, they can be classified into other ways such as hash function based on block ciphers, hash function based on modular arithmetic and dedicated hash functions.

### *B.* Proporities of Hash Functions

Hash functions play a major role in application security today. Hash functions provide different security properties depend on the security requirements of the application. The basic security properties of hash functions are preimage resistance, second preimage resistance and collision resistance. They are explained below:

1. Preimage resistance: for any given code h, it is computationally infeasible to find x such that H(x) = h.
2. Second preimage resistance: for any given input m, it is computationally infeasible to find y ≠m with H(y) = H(m).
3. Collision resistance: it is computationally infeasible to find any pair (m, y) such that H(y) = H(m) [2].

Figure 2 illustrates the definitions of hash function security proprieties.

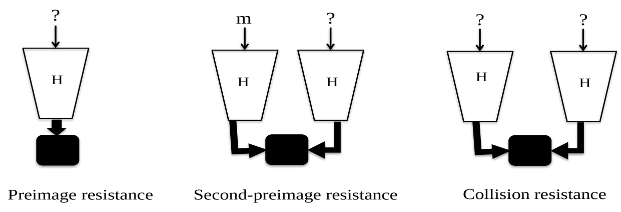

Figure 2. Hash function security proprieties

The preimage resistance property can be described as the inability to learn about the contents of the input data from its hash digest. The second preimage resistance property can be expressed as the inability to learn about the content of the second preimage from the given first preimage such that both of these preimages have same hash digest. The collision resistance property can be interpreted when two different and separate contents of inputs yield to the same hash digest.

## C. Classification of Hash Functions

Building hash functions can be achieved by using various constructions such as Merkle-Damgård or sponge constructions. Merkle-Damgård construction was introduced by R. Merkle's PhD in 1979. It represents a guidance of building dedicated hash functions from compression functions. In 2007, sponge construction introduced in SHA-3 competition by Guido Bertoni, Joan Daemen, MichealPeeter and Gilles Van Assche. It represents the compression function of the new SHA-3 standard (Keccak algorithm).

- The Merkle-Damgård Construction

Merkle-Damgård construction was described by R. Merklein his Ph.D. thesis [6] in 1979. As Figure 3 shows Merkle-Damgård (MD) construction that iterates sequentially a chaining transformation that takes as input message block and the previous chaining value. The input string x is divided into t equal-sized fixed-length blocks xi with bit-length r. Bit-length r corresponds to input length of desired compression function f. The algorithm steps of Merkle-Damgård construction as follows:

1) Break the input x into blocks $x_1$, $x_2$…..$x_t$.
2) Pad the last block $x_t$ with 0-bits if necessary to obtain the multiple length of r.
3) Create the length block $x_{t+1}$ with bit length r to hold the right justified binary representation of overall bit-length of x (MD strengthen).
4) Inputting $x_1$, $x_2$…..$x_t$ to the compression function (iterated processing) to produce an intermediate value of Hi.
5) Hi serves as feedback value to f and is processed with xi+1 in the next iteration. This implies the need of an initial value (IV) H0 for the first iteration that is often provided pre-defined with bit- length r.
6) After processing all the input blocks, then, function g transforms the preliminary result Ht+1 of bit-length r to the final hash-value with desired bit-length. Function g is often the identity mapping [6].

The most distinctive and special part of Merkle-Damgård construction is that the problem of designing a collision-resistant hash function reduced to designing a collision-resistant compression function. This means, if the compression function is collision resistant, then, the hash function is collision resistant. So, the properties of the compression function will be transformed to the hash function.

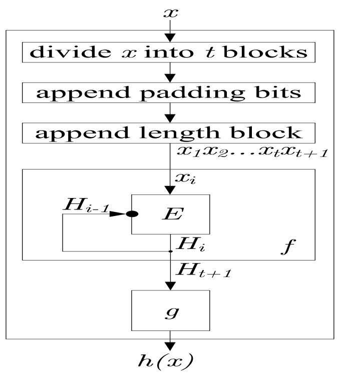

Figure 3. Detailed View of Merkle-Damgård Construction

The well-known Merkle-Damgård construction [6] has determined the basic structure of iterated hash functions. Merkle-Damgård iterates sequentially a chaining of input message blocks and the previous chaining value to produce the final hash digest h(x). Figure 4 shows the Merkle-Damgård strengthen overall design. Padding is an algorithm to extend the input length to become a multiple length of r. Padding is obtained by appending a single '1' bit and '0' bits as many as needed to reach length r. This approach called Merkle-Damgård strengthen or length padding which makes the construction secure.

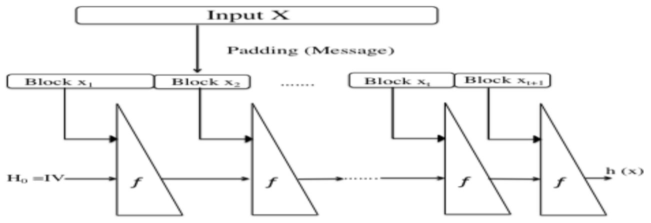

Figure 4. Merkle-Damgård strengthening

The Merkle-Damgård construction used in designing popular hash functions such as MD5, SHA-1 and SHA-2. Also, the Merkle-Damgård construction is quite well studied and several weaknesses (generic attacks) such as multi-collisions [7], long-message second preimage and differentiability [8] have been shown for this construction. Due to the structural weakness founded from these attacks, two intermediate Merkle-Damgård construction versions were developed; wide pipe hash construction and fast wide pipe construction.

- Wide Pipe Hash Construction

Stefan lucks [9] introduced the wide pipe hash construction as an intermediate version of Merkle-Damgård to improve the structural weaknesses of Merkle-Damgård design. Figure 5 shows the wide pipe hash construction. The process is similar to Merkle-Damgård algorithm steps except of having a larger internal state size, which means the final hash digest is smaller than the internal state size of bit length.

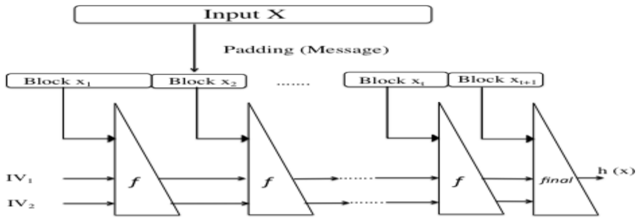

Figure 5. The wide pipe hash construction

Also, the final compression function compresses the internal state length (for ex, 2n- bit) to output a hash digest of n-bit. This simply can be achieved by discarding the last half of 2n-bit output.

- Fast Wide Pipe Construction

Mridul Nandi and Souradyauti Paul proposed the fast wide pipe construction. It is twice faster than the wide pipe construction. Figure 6 shows the fast wide pipe construction. As the Figure shows, the input (IVs) for each compression function is divided into halves.

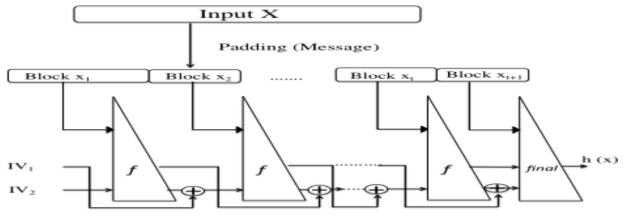

Figure 6. The fast wide pipe hash construction

The first half is inputted in the compression function and the other half is XORed with the output for the same compression function. The feed-forward process makes the overall design faster. Hence, faster process is obtained. The final output of the hash digest can be truncated to the desired digest length using the final compression function.

- The Sponge Construction

Sponge construction is an iterative construction designed by Guido Bertoni, Joan Daemen, MichealPeeter and Gilles Van Assche to replace Merkle-Damgård construction. It is a construction that maps a variable length input to a variable length output. Namely, by using a fixed-length transformation (or permutation) f that operates on a fixed number of b = r + c bits. Where r is called the bitrate and c is called the capacity as Figure 7 shown. First, the input is padded with padding algorithm and cut into blocks of r bits. Then, the b bits of the state are initialized to zero [9]. The sponge construction shown in Figure 7 operates in two phases:

i. **Absorbing phase:** The r-bit message blocks are XORed with the first r bits of the state of the function F. After processing all the message blocks, the squeezing phase starts.
ii. **Squeezing phase:** The first r bits of the state are returned as output blocks of the function F. lastly, the number of output blocks is chosen by the user [9].

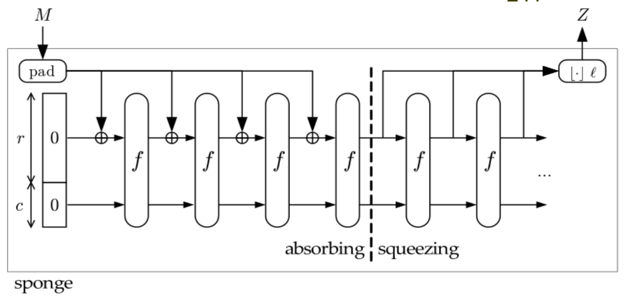

Figure 7. The sponge construction [9]

The sponge construction has been studied by many researchers to prove its security robustness. Bertoni et al. [10] proved that the success probability of any generic attack to a sponge function is upper bound by its success probability for a random oracle plus $N^2/2^{c-1}$ with $N$ the number of queries to $f$. Aumasson and Meier [11] showed the existence of zero-sum distinguishers for 16 rounds of the underlying permutation f of Keccak hash function. Boura, Canteaut and De Cannière [12] showed the existence of zero-sums on the full permutation (24 rounds).

## *D.* Attacks on Hash Functions

Attacks on hash functions are a technical strategy(s) an adversary may use to defeat the objectives of a hash function. These technical strategies may vary and in many cases attacks applied to the compression function of a hash function. A high-level classification of attacks on hash functions is shown in Figure 8.

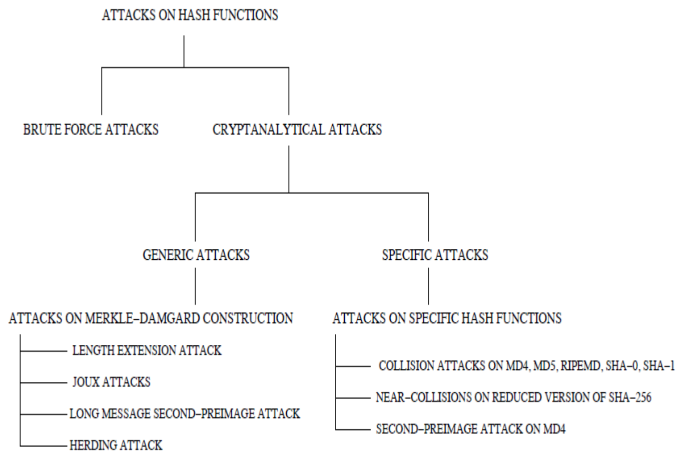
ATTACKS ON HASH FUNCTIONS
BRUTE FORCE ATTACKS
CRYPTANALYTICAL ATTACKS
GENERIC ATTACKS
SPECIFIC ATTACKS
ATTACKS ON MERKLE–DAMGARD CONSTRUCTION
LENGTH EXTENSION ATTACK
JOUX ATTACKS
LONG MESSAGE SECOND–PREIMAGE ATTACK
HERDING ATTACK
ATTACKS ON SPECIFIC HASH FUNCTIONS
COLLISION ATTACKS ON MD4, MD5, RIPEMD, SHA–0, SHA–1
NEAR–COLLISIONS ON REDUCED VERSION OF SHA–256
SECOND–PREIMAGE ATTACK ON MD4

Figure 8. Attacks on hash functions

Attacks on hash functions are mainly classified into two categories: brute force attacks and cryptanalytical attacks.

- Brute force Attacks

Brute force attacks are a particular strategy used to try randomly computed hashes to obtain a specific hash digest. Hence, these attacks do not depend on the structure of the hash function (i.e compression function). The security of any hash function lies on the output hash digests length. Which means, the longer hash digest the more secure hash function. The brute-force attack is a trial and error method to obtain a desired hash function. As an example of a brute-force attack is a dictionary attack which contain a list of dictionary words to try them all in a consecutive manner. These brute-force attacks can always be attempted, however they are not considered as a break unless the required number of evaluations of the hash function is significantly less than both the strength estimated by the designer of the hash function and that of hash functions of similar parameters with ideal strength [13].

- Cryptanalytical Attacks

A hash function cryptanalysis attempts to attack the properties of hash functions such as a preimage attack, second preimage attack and collision attack. Due to fixed size of the hash values compared to much larger size of the messages, collisions must exist in hash functions. However, for the security of the hash function they must be computationally infeasible to find. Note that collisions in hash functions are much easier to find than preimages or second preimages. Informally, a hash function is said to be "broken" when a reduced number of evaluations of the hash function compared to the brute force attack complexities and the strengths estimated by the designer of the hash function are used to violate at least one of its properties immaterial of the computational feasibility of that effort. For example, assume that it requires 290 evaluations of the hash function to find a collision for a 256-bit hash function. Though it is impractical to generate this amount of computational power today, the hash function is said to be broken as this factor is less than the $2^{128}$ evaluations of the hash function required by the birthday attack. This theoretical break on the hash function is also termed an "academic break" on the hash function. It should be noted that hash functions are easier to attack practically than encryption schemes because the attacker does not need to assume any secrets and the maximum computational effort required to attack the hash function is only upper bounded by the attacker's resources not users gullibility. This is not the case with block ciphers where the maximum practical count of executions of the block algorithm is limited by how much computational effort the attacker can get the user to do [14]. As Figure 8 shows that cryptanalytical attacks on hash functions are classified into two categories: generic attacks and specific attacks.

- Generic Attacks (Attacks on Merkle-Damgård construction)

Generic attacks are technical studies used to attack general hash function constructions (i.e Merkle-Damgård construction). The word *"generic"* means that the attack is not designed for a specific hash function (i.e SHA-2). For example, if the hash function uses a certain block cipher, replacing this block cipher with another should not affect the complexity of a generic attack of that hash function. The generic attacks are classified into four types, discussed in the following sections.

1) Length Extension Attacks

An attacker can use the advantage of using the padding scheme for the messages in Merkle-Damgård construction by applying length extension attack (it is also called extension attack). Length extension attack can be used to break secret prefix MAC scheme where the attacker computes the authentication tags without the knowledge of the secret key.

2) JouxAttack

Joux attack or Joux multi-collision attack is an attack on Merkle-Damgård hash function, where Antoine Joux shown that finding multiple collisions (more than two messages hashing to the same digest) in a Merkle-Damgård hash function is not much harder than finding single collisions as Figure 9 shown. In his multi-collision attack, Joux assumed access to a machine C that given an initial state, returns two colliding messages [7].

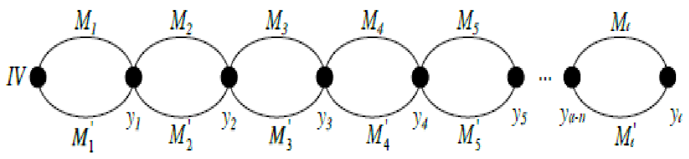

Figure 9. Joux Attack [7]

Also, Joux used his multi-collision in Merkle-Damgård hash function to produce a collision attack in a concatenation of two independent hash functions. Particularly, this attack deemed to be the first spark to look forward to start searching for a new paradigm of mode of operation of hash function other than Merkle-Damgård construction and hence announced SHA-3 competition.

3) Long Message second preimage Attacks

In the second preimage attacks, the attacker finds a second preimage S for a given message M, where $M \neq S$ and $H(M) = H(S)$ with an effort less than $2^t$ computation of H. In the long message second preimage attack, the attacker tries to find a second preimage for a long target message M of $2^q+1$ message blocks. The attacker does this by finding a linking message block $M_{link}$. Where, the digest of $f_{IV}$ of the linking message block $M_{link}$ matches one of the intermediate states Hi obtained in the hashing of M. The computation cost of this attack is about $2^{t-q}$ calls to the compression function *f*.

4) Herding Attack

This attack is due to Kelsey and Kohno [15] and is closely related to the multi-collision and second preimage attacks discussed above. A typical scenario where this attack can be used is when an adversary commits to a hash value D (which is not random) that he makes public and claims (falsely) that he possesses knowledge of unknown events (events in the future) and that D is the hash of that knowledge. Later, when the corresponding events occur, the adversary tries to herd the (now publicly known) knowledge of those events to hash to D as he previously claimed [15].

- Specific Attacks (Attacks on Specific Hash Functions)

Specific attacks on hash functions are based on the hash function itself. For example, attacks on MD5 [16], SHA-0 [17] and SHA-1 [18] are called multi-block collision attacks. Multi-Block Collision Attack (MCBA) technique on iterated hash function (i.e Merkle-Damgård construction) finds two colliding messages each at least two blocks on length. In such attack, collisions are found by processing more than one message block. In fact, multi-block collisions attack is applicable and valid on MD5, SHA-0 and SHA-1, since these hash functions use more than a single and collisions are distributed randomly.

As Figure 8 shows the sub-categories of "attacks on specific hash functions" which are (collision attack on MD4, MD5, RIPEMD, SHA-0 and SHA-1; near collisions on reduced version of SHA-256 and second preimage attack on MD4) are only examples of specific attacks on these hash functions. Meaning that, attacks can be customized and applied based on the hash function behavior and architecture.

## 3. Applications of Hash Functions

Hash functions play a major role in application's security such as certification, data integrity and authentication. The following sections illustrate these applications.

### *A. Digital Signature*

Digital Signature is a mathematical scheme used to validate the authenticity of the sender, message and signer of the document identity as Figure 10 shown. Digital signatures are commonly used in Web-commerce, financial transactions and other cases where it is crucial to detect alteration of a message or a document. It uses private and public keys along with the hash digest to create the signature for a document. Digital Signature provides signer authentication and authorization of a document. It indicates who signed a document, message or record and makes difficult for another person to produce the same without authorization [29].

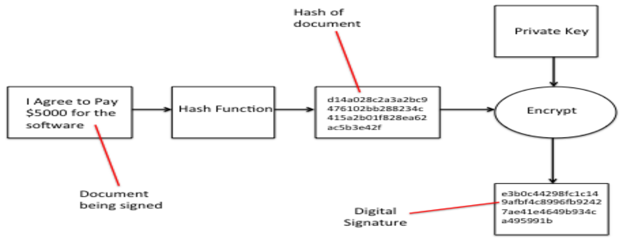
Hash of document
Private Key
I Agree to Pay $5000 for the software
Hash Function
d14a028c2a3a2bc9 476102bb288234c 415a2b01f828ea62 ac5b3e42f
Encrypt
Document being signed
Digital Signature
e3b0c44298fc1c14 9afbf4c8996fb9242 7ae41e4649b934c a495991b

Figure 10. Digitally signed document

### *B. MAC*

A Message Authentication Code (MAC) is similar in usage to a message digest. It is designed especially for applications to detect message tampering and forgery. MAC accepts a shared secret symmetric key (K) as input along with the arbitrary length and outputs MAC (sometimes called tag). Figure 11 shows the MAC algorithm process.

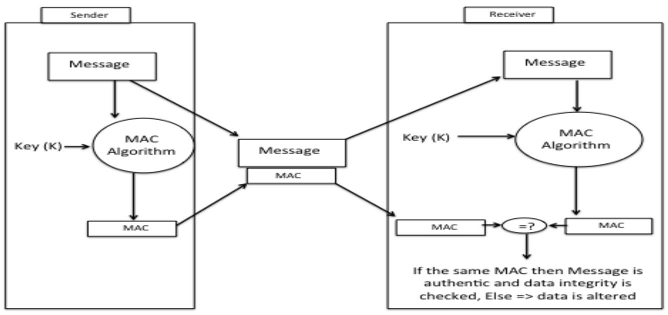
Sender
Message
Key (K)
MAC Algorithm
MAC
Message
MAC
Receiver
Message
Key (K)
MAC Algorithm
MAC
=?
MAC
If the same MAC then Message is authentic and data integrity is checked, Else => data is altered

Figure 11. MAC algorithm

As Figure 11 shows the process of the MAC algorithm, sender calculates the MAC by first calculating message digest of the message or document and then applying secret key K to the message digest. Then, the message with the calculated MAC sends to the receiver [29]. Independently, the receiver calculates a new MAC value by using the symmetric secret key (K) and generate new hash digest. If the attached MAC with original message matches the new calculated MAC performed by the receiver then the message is authenticated and integrity verified.

MACs differ from digital signature as MAC uses a symmetric secret key and digital signature uses asymmetric key (public and private keys).

### *C. HMAC*

A popular and specific implementation of message authentication codes is the HMAC (Hash Message Authentication Code). It is a specific construction for calculating a message authentication code which involves a secret key and cryptographic hash function to ensure secure data transfer over unsecure channels. As computers becoming more powerful, the need arises for complex hash functions. As a result, HMAC is a preferable to use with hash functions other than MAC due to its higher security. Cryptographic hash functions such as SHA-1 or MD5 may be used to calculate HMAC. In this case, the resulting hash function called HMAC-SHA-1 and HMAC-MD5 respectively.

### *D. Kerberos*

Kerberos is network authentication protocol. It is designed to provide strong authentication and improved security for users and between client/server applications. Kerberos was developed at the Massachusetts Institute of Technology (MIT) in 1998. Using Kerberos, a user can request an encrypted "ticket" from an authentication process so it can be used to request a specific service from a server.

### *E. Key Derivation*

A key derivation function (KDF) is an algorithm to derive a key of a given size from a secret value or other known information. That is used to derive keys from a secret value such as a value obtained by Diffie-Hellman key establishment. Keyed cryptographic hash function can be used for key derivation.

### *F. One Time Password*

Cryptographic hash functions are used to compute one time password (OTP). OTP is a password that is valid for a single login or transactions. By using cryptographic hash functions, hashed passwords are saved instead of keeping the password itself. So that, if the file of passwords are revealed then the passwords still protected if the hash function is preimage resistance.

### *G. Pseudorandom generator*

A cryptographic hash function can be used to generate pseudorandom generator (PRG). PRGs are used to generate pseudorandom bits from a short random seed, which can then be used in place of truly random bits that most cryptographic schemes rely on. On the foundational side, PRGs can be used as a building block for more complex cryptographic objects like pseudorandom function (PRF), bit commitment, etc [19].

### *H. Pretty Good Privacy*

Pretty Good Privacy or PGP is a popular program that is used to encrypt/decrypt and authenticate e-mails over the internet. PGP uses a hash function to ensure the integrity of e-mail message.

### *I. Secure Socket Layer/Transport Layer Security*

Secure Socket Layer (SSL) and Transport Layer Security (TLS) protocols are used to authenticate servers and clients over an untrusted network. SSL/TLS can help to secure data transferred using encryption. Also, SSL/TLS can authenticate servers as well as clients through secure communication.

## 4. An Overview of A Selected Hash Functions

### *A. MD4 and MD5*

The cryptographic hash function MD4 (Message Digest 4) was introduced by Ronald Rivestin 1990. MD4 was a novel design, which compresses an arbitrary input length and produce 128-bits as a hash digest. Later, other hash algorithms such as MD5, SHA-0, SHA-1 and HAVAL were derived and influenced by MD4. In 1991, hash function Message Digest 5 or MD5 was designed by Ronald Rivest as a strengthen version of MD4. MD5 is widely used algorithm in a variety of security applications.

Working of MD5 is almost similar to MD4 but some changes have been made to MD4. One extra round is added in MD5. MD5 also compresses arbitrary bit-length input into a 128-bit hash value [29].

### B. *RIPLEMD*

RIPEMD is a cryptographic hash function developed by Hans Dobbertin, Antoon Bosselaers and Bart Preneel, and first published in 1996. Its design is based on MD4. Which is consists of two parallel versions of the MD4 compression function. RIPEMD produce 160-bits as a hash digest. Dobbertin found a collision attack on two rounds of RIPEMD. Strengthen versions of RIPEMD were developed due the weakness founded in RIPEMD-160 bits [29]. These versions are RIPEMD-128, RIPEMD-256 and RIPEMD-320. RIPEMD produce 128-bits of hash digest. The extended version of RIPEMD-128 is RIPEMD-256, which produce 256-bits as a hash digest. Also, the extended version of RIPEMD-160 is RIPEMD-320, which produce 320 bits as a hash digest.

### C. *SHA-x Family*

- Secure Hash Algorithm-0

National Institute of Standard and Technology (NIST) along with National Security Agency (NSA) published the *Secure Hash Algorithm (SHA)* in 1993. At present, SHA is commonly referred to SHA-0. SHA-0 is an algorithm that produces a 160-bits hash digest. SHA-0 was developed to replace MD4 but it was withdrawn shortly after publication due to security issues.

- Secure Hash Algorithm-1

SHA-0 was replaced by SHA-1 in 1995. *Secure Hash Algorithm-1* or SHA-1 is a message digests algorithm, which is regarded the world's most popular hash function, which takes input a message of arbitrary length and produce output a 160 bits "fingerprint" of the input. However, the security level of this standard is limited to a level comparable to an 80-bit block cipher [20]. It is based on the design principle of MD4, and applies the Merkle-Damgård model of compression function.

- Secure Hash Algorithm-2

In August, 2002, NIST has published three additional hash functions, SHA-256, SHA-384 and SHA-512. These new hash functions family known as *Secure Hash Algorithm-2* or simply SHA-2. SHA-2 was introduced due to the need of a larger key of a hash function to match the new *Advanced Encryption Standard (AES)* which introduced in 2001. In February 2004, another hash function SHA-224 was added to the SHA-2 family. SHA-224 and SHA-384 are the truncated versions of SHA-256 and SHA-512 respectively. The proposed system architecture of SHA-2 hash family can support efficiently the security needs of modern communication applications such as WLANs, VPNs and firewall [29].

- Secure Hash Algorithm-3

In October 2012, NIST announced the winner of SHA-3 competition which started in 2008. Keccak was the winner of NIST competition and become the new SHA-3 standard. Keccak is a cryptographic hash function designed by Guido Bertoni, joan Daemen, Michael Peeters, and Gilles Van Assche. Keccak has completely different construction than SHA-0, SHA-1 and SHA-2 families. It supports at least four different output lengths *n* {224, 256, 384, and 512}in a high security levels [21]. According to [22] and [23] the construction of Keccak sponge design is building the compression function from different permutation *f* operates components in the following:

1. Signify the length of message bitstring by $|M|$, as a sequence of blocks in fixed length $x$, when calculated the ranges from 0 to $|M|_x$ -1.
2. Pad the message $M$ in a sequence of x-bit blocks to signify by $M||$ pad $[x]$ ($|M|$). Thus, padding rules have append a bitstring to determined the bit length of $M$ and the block length $x$.
3. It is a sponge hash functions to construct a function of [$f$, pad, $r$] where the permutation $f$ has different length in input and fixed length of output, a padding rule "pad" and a *bitrate* $r$. The permutation $f$ operates has seven set of bits, which denoted as Keccak-$f$ [$b$] wherethe $b$= 25$w$. Keccak-$f$ [$b$] is a permutation over, when the bits are figured from 0 to $b$-1. Thus, the different versions of Keccak-$f$ permutation have limited output values in {25, 50, 100, 200, 400, 800, 1600} to represent the hypercube of sponge construction of three dimensional array, and $c = b - r$ is the *capacity* of absorbing phase of the compressing state (Berton*et al.* 2011). Figure 12 from depicts the sponge construction of Keccak-$f$ [$r + c$].

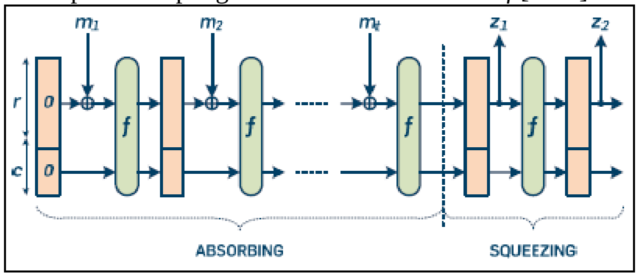

Figure 12. The sponge construction of Keccak [22]

4. The permutation is a sequence of operations on the three-dimensional array of elements of GF(2), specifically $a[5][5][w]$, with $w = 2^{\ell}$, where $w$ {1,2,4,8,16,32,64}. The expression $a[x][y][z]$ with $x$, $y \in \mathbf{Z}_5$ and $z \in \mathbf{Z}w$, signifies the bit in position ($x$, $y$, $z$), follows by indexing starts from zero. The mapping between the bits of $s$ and those of $a$ is $s[w(5y + x) + z] = a[x][y][z]$. That terms in the $x$ and $y$ coordinates should be taken modulo 5 and expressions in the $z$ coordinate modulo $w$. The source state has a fixed value and should never consider as an input [24].

## 5. Conclusion

This paper presented an extensive study of cryptographic hash functions. The presented study surveys cryptographic hash functions from various aspects. It included the properties, classification, constructions, attacks, applications and an overview of a selected dedicated cryptographic hash functions. Practically, MD4, MD5 and SHA-0 considered broken hash functions. Theoretically, SHA-1 considered a broken hash function. But SHA-2 considered secure one. SHA-3 was presented due to the need for a long term security hash function which has a new promising sponge construction.

## 6. References

[1] S. Goldwasser, S. Micali and R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks, SIAM Journal of Computing, vol 17, No. 2, pp. 281-308, April 1998.

[2] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications, Proceedings of the

Twenty First Annual ACM Symposium on Theory of Computing, ACM Press, pp 33-43, 1989.

[3] S. Haber and W. S. Stornetta. How to timestampting a digital document. Journal of Cryptology 3(2), pp. 99-111, 1991.

[4] H. Krawczyk, M. Bellare and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. Internet RFC 2104, February 1997.

[5] V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal of Computing 33:167-226, 2003.

[6] I. Damgård. A design principle for hash functions. In G. Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 416– 427. Springer, 1990.

[7] Antoine Joux. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In Matt Franklin, editor, Advances in Cryptology- CRYPTO 2004, volume 3152 of Lecture Notes in Computer Science, pages 306–316. Springer, August 15–19 2004.

[8] U. Maurer, R. Renner, and C. Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In M. Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.

[9] Lucks, S. (2004). Design principles for iterated hash functions, Cryptology ePrint Archive, Report 2004/253, 2004, http://eprint. iacr. org.

[10] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. On the indifferentiability of the sponge construction. *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, 4965:181–197, 2008.

[11] J.-P. Aumasson and W. Meier. Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi. *NIST mailing list*, 2009.

[12] C. Boura, A. Canteaut, and C. D. Canniére. Higher-order differential properties of Keccak and Luffa. Cryptology ePrint Archive, Report 2010/589, 2010. http://eprint.iacr.org/2010/589.pdf.

[13] Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of SHA-0 and Reduced SHA-1. In Ronald Cramer, editor, Advances in Cryptology - EUROCRYPT 2005, vol- ume 3494 of Lecture Notes in Computer Science, pages 36–57. Springer, 2005.

[14] [24] Yuliang Zheng, Josef Pieprzyk, and Jennifer Seberry, 1993." HAVAL – A One-Way Hashing Algorithm with Variable Length of Output", Lecture Notes in Computer Science, Volume 718, Advances in Cryptology – Auscrypt '92, pp. 83–104.

[15] J. Kelsey and T. Kohno. Herding hash functions and the nostradamus attack. In S. Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006,Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 183–200. Springer, 2006.

[16] Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, Advances in Cryptology - EUROCRYPT 2005, volume 3494 of Lecture Notes in Computer Science, pages 19–35. Springer, 2005.

[17] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Efficient collision search attacks on SHA-0. In Victor Shoup, editor, Advances in Cryptology— CRYPTO '05, volume 3621 of Lecture Notes in Computer Science, pages 1–16. Springer, 2005, 14–18 August 2005.

[18] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, Advances in Cryptology— CRYPTO '05, volume 3621 of Lecture Notes in Computer Science, pages 17–36. Springer, 2005, 14–18 August 2005.

[19] Blum, M. and S. Micali (1984). "How to generate cryptographically strong sequences of pseudorandom bits." SIAM journal on Computing **13**(4): 850-864.

[20] Vincent Rijmen and Elisabeth Oswald, 2005." Update on SHA-1". In Alfred Menezes, editor, Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, volume 3376 of LNCS, pp. 58–71.

[21] E. Andreeva, B. Mennink, B. Preneel & M. Skrobot (2012), Security Analysis and Comparison of the SHA-3 Finalists BLAKE, Grostl, JH, Keccak, and Skein. from Katholieke Universiteit Leuven, Belgium.

[22] G. Bertoni, J. Daemen, M. Peeters, & G. V. Assche (2012), Keccak An update. Retrieved March 22-23, 2012, from Third SHA-3 candidate conference, Washington DC.

[23] E. B. Kavun & T. Yalcin (2012), On the Suitability of SHA-3 Finalists for Lightweight Applications. from Horst Görtz Institute, Ruhr University, Chair of Embedded Security, Germany. Website: http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/March2012/ documents /papers /KAVUN_paper.pdf

[24] Imad Fakhri Al-shaikhli, Mohammad A. Alahmad and Khansaa Munther. The "Comparison and analysis study of sha-3 finallists." International Conference on Advanced Computer Science Applications and Technologies(26-28 Nov 2012): 7.

[25] Mohammad A. Ahmad, I. F. A. S., Hanady Mohammad Ahmad (2012). "Protection of the Texts Using Base64 and MD5." JACSTR Vol 2, No 1 (2012)(1): 12.

[26] Imad F. Alshaikhli, M. A. Ahmad. (2011). "Security Threats of Finger Print Biometric in Network System Environment." Advanced Computer Science and Technology Research **1**(1): 15.

[27] Al-Kuwari, S. and Davenport, J.H. and Bradford, R.J. Cryptographic hash functions: recent design trends and security notions. Science Press of China,2010.

[28] Sobti, R. and G. Geetha (2012). "Cryptographic Hash Functions: A Review." IJCSI International Journal of Computer Science Issues **9**(2): 461-479.

[29] Tiwari, H. and A. Krishna (2010). "Cryptographic hash function: an elevated view." European Journal of Scientific Research **43**: 452-465.