Functional Analysis of Keyless Digest Functions: A Security Perspective

P.KARTHIK, Dr. P.SHANTHI BALA

Abstract: The term data security revolves around two radical things namely data protection and data integrity. The advent of cloud solutions has completely transformed the data storage and access mechanisms. Today, technology permits the user to store and access data through the internet without much access restriction. Therefore, conserving the integrity of data becomes the most grueling task than it was thought formerly. The digest functions come in aid to provide a comprehensive solution for the integrity violations of remote data. But, the cryptographic attacks on the digest functions like MD4, MD5, RIPEMD, and SHA-160 algorithms made the research community to reconsider the design principles of the digest functions for the cryptographic use. This work attempts to perform a functional analysis of the standard keyless-digest functions like MD-5, SHA-160, SHA-2 Family, and SHA-3 family in the perspective of security.

Index Terms: Keyless Digest Functions, Functional Analysis of Standard keyless-digest functions, Cryptographic digest functions, Construction principles of modern digest functions, Security vulnerabilities of modern digests, Security analysis of digest functions, Keyless digests, Security metrics of MDC for integrity verification.

1. INTRODUCTION

The digest functions are designed to provide security to the remote data. The inability of the owner not to physically manage the remote data triggers many concerns in the domain of data security. The hash function provides a comprehensive solution for the aforesaid problem. These functions do not directly involve in providing data security, but they aid to identify the security infract of the remote data maintained in a remote server. A hash function takes an arbitrary length input string to produce fixed size output. The output string is typically represented in the form of hexadecimal numbers[1],[2]. The input size is usually more immense than the output size. If m represents the size of the input message and n represents the size of the output, then the hash function would produce digest such that m > n. Therefore, it is unavoidable the cryptographic hash functions would generate collisions, or it is believed to do so in accordance with the pigeon-hole[1] principle. But the success of the cryptographic functions relies on how these functions prevent hash collisions despite revealing the secrets of its design paradigm and functionalities. The paper is established as follows. Part 2 deals with an overview of the digest functions. Part 3 elaborates the properties of provably secure digest functions. Part 4 presents the design principles of the standard digest functions. Part 5 deals with the cryptographic hash attacks. Part 6 advocates the security analysis of digest functions. Part 7 deals with the discussion on vulnerabilities of the standard digest functions, and Part 8 ends with a conclusion.

2. CRYPTOGRAPHIC HASH FUNCTION-AN OVERVIEW

A hash function is called a cryptographic hash function if it satisfies some properties like collision resistance, pre-image resistance, and second pre-image resistance[3],[4]. The cryptographic hash functions are much similar to other hash functions that it takes arbitrary length input of the form $\{0,1\}^*$ and produces fixed-size output $\{0,1\}^n$. The value n here represents the size of the hash output. This could be represented as follows.

$$H : \{0,1\}^* \longrightarrow \{0,1\}^n$$

The cryptographic hash functions are classified into two significant categories namely keyed and keyless hash functions. The keyless hash functions do not use any external keys. Most of the keyless hash functions use Merkel-Damgard(MD) construction for the design of hash functions. Some of the standard algorithms that entail MD construction are MD4, MD5, SHA-0, SHA-1 and RIPEMD, and the family of SHA-2 digest functions. Keyless hash functions are typically applied for the integrity confirmation of remote data. Contrarily, the keyed hash functions use keys to produce the hash values. Depending upon the nature of the key used, the keyed hash functions are again classified into two types namely public-key cryptographic hash function and private-key cryptographic hash function. In the private-key hash function, both the sender and the receiver share the corresponding key. The sender uses a key to produce hash value for the message to be transmitted which is then used by the beneficiary later to endorse the message. The thorough process is presented in Figure 1. In the receiving end, the beneficiary collects the message and calculates the hash using the shared key K. If both hash values are matched, then the integrity is achieved. Therefore, the private-key cryptographic hash functions are typically applied to construct message authentication Code (MAC)[8]. Figure 2 demonstrates the authentication process at the receiving end.

- *P.KARTHIK is currently pursuing a doctorate degree at the Department of Computer Engineering, School of Engineering and Technology, Pondicherry University, Puducherry, India. E-mail: thooralonly@gmail.com.*
- *Dr. P.SHANTHI BALA is currently working as an assistant professor at the Department of Computer Engineering, School of Engineering and Technology, Pondicherry University, Puducherry, India. E-mail: shanthibala.cs@mail.com.*

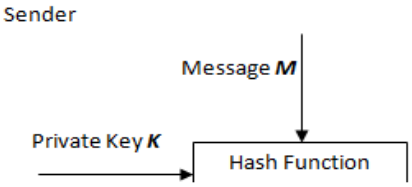

Figure 1: Working principle of Private-Key Digest Function -Sending end

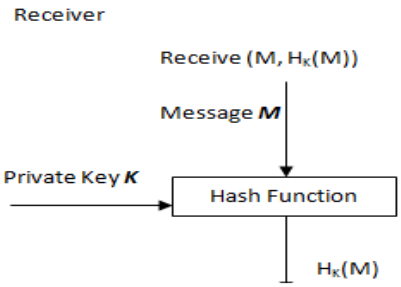

Figure 2: *Working principle of Private-Key Digest Function -Receiving end*

Contrarily, the public-key digest function applies two keys. The sender would sign the data using the private key. The signed data is later verified with the sender's public key at the receiving end[11],[9]. The public-key cryptographic digest functions are more secure than private-key MAC for the reason that, the sender never shares the private key in the process of message authentication. Damgard adopted RSA public cryptography for the first time in constructing a hash function. A digital signature implements this principle for message authentication.

3. PROPERTIES OF PROVABLY SECURE HASH FUNCTION

The digest function would satisfy some key properties, to be considered for cryptographic use. The properties are provided as follows.

1. Collision resistance

The collision resistance property advocates that, for any two messages m1 and m2 such that m1 ≠ m2 , their digest values H(m1) and H(m2) would differ from one another (i.e) H(m1) ≠ H(m2).

2. Pre-image resistance

Pre image resistance is also called as one-way property and it says, for any message m, it is straightforward to generate H(m). Nothing but, it is computationally infeasible to perceive m from H(m). Put differently, it is infeasible for a cryptanalyst to find a different message m' such that H(m)=H(m').

3. Second pre-image resistance

The second pre-image resistance entails that for any two different messages m1 and m2, their hash values H(m1) and H(m2) would differ from one another (i.e) H(m1) ≠ H(m2).

In addition to the aforesaid properties, Bartkewitz and Al-Kuwari et al., [1],[3], have mentioned some other properties for secure hash functions. They are presented as follows.

a. Compression - The digest function should map an arbitrary input message of size m into fixed-size output n, such that m>n.

b. Ease of computing - The digest function should not demand more CPU clocks to produce a message digest.

c. Non-correlation - This property enables the hash function to produce bizarre output for a given input. Every input bit possibly should affect every output bit[6]. To achieve this property, the digest function should meet the strict avalanche criterion. This property would affect more than 50% of the output bits for a single bit change in the input. The avalanche effect is used to prevent a differential attack[5],[7],[10].

d. Near collision resistance - For any two different messages m1 and m2, their hash values H(m1) and H(m2) should differ by a trivial number of bits.

e. Partial pre-image resistance - Let m be the message and H(m) be its hash value. The partial pre-image resistance insists an attacker should not gain the portion/subset of message m from H(m).

f. Semi free-start collision resistance - This sort of collision happens when two different messages m1 and m2 use the same initialization vector(IV) irrespective of the input message. However, the discussion is invalid if the algorithms allow the user to change IV upon the message.

g. Pseudo-collision resistance - Here the collision is produced by changing the IV for contradictory messages. As like the above, the discussion is invalid when the hash algorithms, do not offer an opportunity for the users to fix the IV.

h. Pseudorandom Oracle -Randomness is ordinarily considered as the measure of security and most desirable property of any cryptographic hash function. Therefore, the research community wanted to introduce erratic behavior to the hash functions through Random Oracle(RO). This would prevent an attacker to launch a differential attack and to set mathematical conditions for finding collisions. The RO represents a mathematical model which produces arbitrary output whenever queried. This model would produce bizarre outputs for different queries and would produce identical outputs for similar queries. Conceptually, this model would try for one to one mapping from variable-sized input to fixed-size output. The idea of replacing the hash function with RO would surely avoid hash collisions. Mihir Bellare et al had introduced the Random Oracle model to conceptualize provably secure cryptographic hash function[12],[13]. They had provided solution for protocol problem π using RO paradigm as follows.

- Find the formal definition of π in RO model and made this accessible to public.
- Construct a protocol P for π.
- Prove the P satisfies the properties of π.
- Replace RO access with P.

Bellare et al had claimed that any cryptographic hash function follows RO paradigm can be proven to be secure. However, the claim was proven wrong by Coron et al [16]. In his work, he disputed the claim of the nonexistence of structural flaws in the design of the iterative hash model. He appealed that the claim of structural flaws would not exist is applicable only to monolithic functions and not to iterative functions. He believed most of the modern cryptographic hash functions that manage Merkle-Damgard construction of the iterative model would contain some structural flaws. Therefore, any system that claims itself as secure by possessing one-way, collision resistance and pseudo-randomness not necessarily to make it a secure system. Cannetti R et al [14],[15] in his work proved the system makes sure of in the random-access model, on its execution proved to be insecure.

4. HASH FUNCTION : DESIGN PRINCIPLES

Hash function compresses the arbitrary length input into a fixed-size output. Though many hash algorithms were proposed for cryptographic use, it was Merkel and Damgard independently demonstrated the provably secure properties for the first time for cryptographic use[22],[23]. Merkel proved collision resistance property in a keyless approach, while Damgard demonstrated with a keyed design employing RSA principles. Their design became the standard for the cryptographic hash functions. The modern cryptographic functions MD5 and SHA-160 implement this design for producing hash values. The construction of Merkel-Damgard is provided as follows.

4.1 Markel-Damgard Construction

Let M represents the message to be hashed. The algorithm divides the message into fixed-size blocks as follows.

M={M1,M2,M3,...,Ml)

The block size m is ordinarily in multiples of 512 bits or 64 bytes. If the message size is not in multiples of the block size, then padding would be applied. This is defined as length padding. The first bit of the padding would be set to one, where the remaining bits would be carried out with zeros. This is to ensure that the padded bit values would be in multiples of two. It was the first time Xucjia Lai and James Massey[24], who coined the term Merkel-Damgard strengthening for length padding. The padding is typified by the letter d, and the number of bytes to be padded is calculated adopting the formula provided at the end.

d=M+1+64 MOD m

The diagram presented in Figure 3 demonstrates the thorough process of Merkel-Damgard construction.

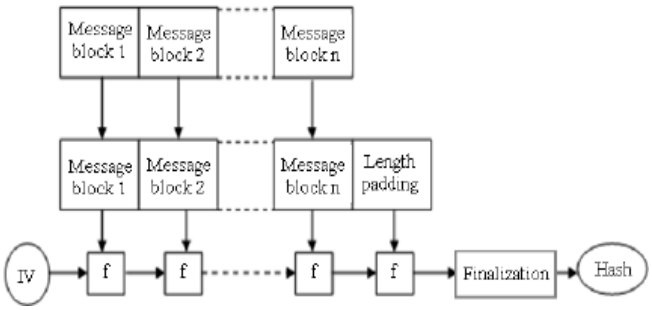

Figure 3: *Merkel- Damgard Strengthening*

4.2 The MD5 digest function

The MD5 is a 128-bits algorithm and it works on the input block of 512 bits[25]. It divides the input message M into 512 bits or 64 bytes of blocks. If the message size is not exactly in multiples of block size then the padding would be applied. It reserves 64 bits of data at the end message block to preserve the value of the message length. Therefore, this algorithm could be efficient to handle the messages of size less than 2^{64} bits. It uses 4 internal state vectors namely A, B, C, and D each of the size 32-bits. MD5 works on 64 rounds and the individual round is indicated by K_i. The value of K_i is calculated employing the following formula.

K_i=abs(sin(i + 1)*2^{32})

The algorithm uses 4 auxiliary functions changed for every 16 rounds.

a) F(X,Y,Z) = (X & Y) | (X & Z) => R1-R16

b) G(X,Y,Z) = (X & Z) | (Y & $\overline{Z}$) => R17-R32

c) H(X,Y,Z) = X ⊕ Y ⊕ Z => R33- R48

d) I(X,Y,Z) = Y ⊕ (X | Z) => R49-R64

The algorithm additionally employs left-shift and addition modulo operations as follows.

$<<<_n$ - Left-shift n bits.

⊞ - Addition Modulo 2^{32}

The diagram given in Figure 4 demonstrates the operating principle of the MD5 algorithm.

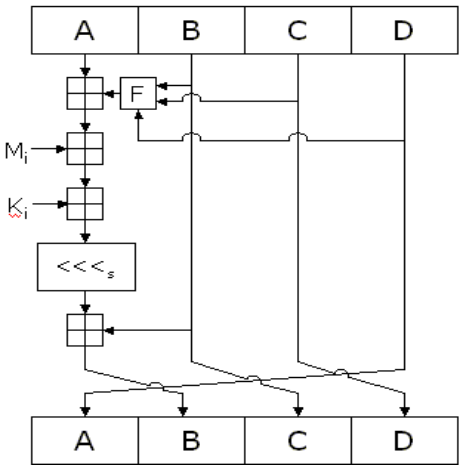

Figure 4: *MD5 Working Principle*

4.3 The SHA-160 digest function

The SHA-160 stands for Secure Hash Algorithm and it is a 160-bits hash algorithm. Similar to MD5 it works on the input block of 512-bits and the padding is done using Merkel-Damgard strengthening[26]. It reserves 64 bits of data at the end block to store the length of the message. It uses five 32-bit constants(A,B,C,D,E and F) as internal state vectors that perform the role of IV. The SHA-160 works on 80 rounds and for every 20 rounds it alters the nonlinear function F and the key K_t as follows. Here the value t lies between 1 to 80.

- F(t;B,C,D) = (B & C) | (B & D) =>t0- t19
- F(t;B,C,D) = B ⊕ C ⊕ D =>t20- t39
- F(t;B,C,D) = (B & C)|(B & D)|(C & D) =>t40-t59
- F(t;B,C,D) = B ⊕ C ⊕ D =>t60 -t79.

Similarly,

K(t) = 5A827999 (0 <= t <= 19)
K(t) = 6ED9EBA1 (20 <= t <= 39)
K(t) = 8F1BBCDC (40 <= t <= 59)
K(t) = CA62C1D6 (60 <= t <= 79).

The algorithm generates expanded word for each round that is indicated by Wt as given at the end.

- Let M represents the message such that M={M1,M2,...,Mn}
- Each message block Mi is of size 512-bits or 64 bytes. Divide each block into 16 words of 32-bits each. The distinct 32-bits word is initially assigned to W0 - W16.
- Then, For every round t = 16 to 79, it calculates Wt as follows,

W(t) = S^1(W(t-3) ⊕ W(t-8) ⊕ W(t-14) ⊕ W(t-16)).

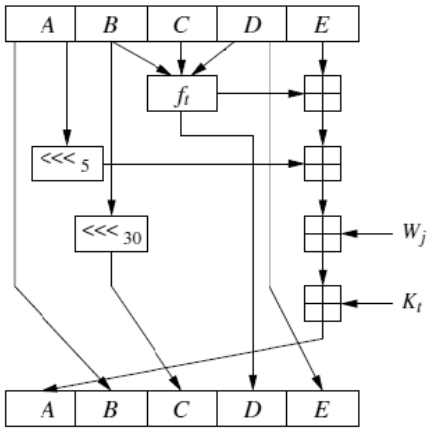

Figure 5: *SHA-160 Working Principle*

4.4 The SHA-256 digest function

SHA-2 stands for Secure Hash Function 2. The construction principle of SHA-2 is completely different from its predecessor SHA-1. There are six variants of SHA-2 available. They are, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256. Among the variants, SHA-256 and SHA-512 represent the primitive versions. The other variants are the truncated forms of either SHA-256 or SHA-512. The SHA-256 operates on 32-bit words, whereas the SHA-512 operates on 64-bit words. The construction and other functional aspects of both algorithms are the same. The SHA-2 process the input message as similar as to its predecessor SHA-1. Padding would be applied when the message size is not exactly in multiples of the block size. The block size continues 512-bits. This algorithm uses six auxiliary functions listed below.

a) $Ch(x, y, z)=(x \wedge y) \oplus (x \wedge z)$
b) $Maj(x, y, z)=(x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$
c) $\sum_0(x) = S^2(x) \oplus S^{13}(x) \oplus S^{22}(x)$
d) $\sum_1(x) = S^6(x) \oplus S^{11}(x) \oplus S^{25}(x)$
e) $\sigma_0(x) = S^7(x) \oplus S^{18}(x) \oplus R^3(x)$
f) $\sigma_1(x) = S^{17}(x) \oplus S^{19}(x) \oplus R^{10}(x)$

At this place, S^n and R^n represent n bits circular right shift and right shift operations respectively. This algorithm uses 8 initial hash values H_1^0 to H_8^0. The hash values are initialized with 32 bits of the square roots of first 8 prime numbers. The following operations will be carried out for every message block Mi ∈ M.

- Initialize the registers a, b, c, d, e, f, g, and h with intermediate hash values. The intermediate hash values will be the initial hash values during the start of the iteration.
- Apply the register values in the compression function and perform compression in 64 rounds. In each of this round performs the following.
- Compute Ch(e, f, g), Maj(a, b, c), $\sum_0(a)$, $\sum_1(e)$, and W_i by employing the auxiliary functions.
- Compute W_i as;
  $W_i = \sigma_1 W(i-2) \boxplus W(i-7) \boxplus \sigma_0 W(i-15) \boxplus W(i-16))$.
- Calculate T_1 and T_2 and the other register values as;
  $T_1 = h \boxplus \sum_1(e) \boxplus Ch(e, f, g) \boxplus K_i \boxplus W_i$
  $T_2 = \sum_0(a) \boxplus Maj(a, b, c)$
  $h=g$; $g=f$; $f=e$; $e= d \boxplus T1$; $d=c$; $c=b$; $b=a$;
  $a= T_1 \boxplus T_2$.
- Calculate Intermediate Hash values as follows.
  $H_1^i = a \boxplus H_1^{(i-1)}$
  $H_2^i = b \boxplus H_2^{(i-1)}$
  . . .
  $H_8^i = h \boxplus H_8^{(i-1)}$

$H^{(N)} = \{ H_1^{(N)}, H_2^{(N)}, H_3^{(N)}, ..., H_8^{(N)} \}$ will the final hash value for the given message M. The diagram given at the end outlines the operating principle of SHA-2 compression function[37].

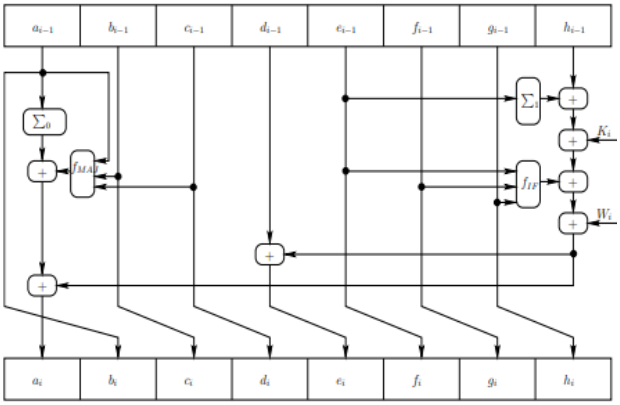

Figure 6: *SHA2-256 digest function*

4.5 The SHA3 digest function

The SHA-3 digest function was an outcome of the direct competition conducted by the NIST, with the objective to introduce a new standard SHA-3, in cryptographic hash function[28]. It was during October 2012 NIST announced Keccak as the winner of the competition[29]. Keechak replaced RO with sponge construction. The current model would produce variable-length output to make it suitable for various cryptographic applications like mask generation, stream cipher, and MAC computing. The contemporary standard additionally offers duplex construction which performs well together with sponge construction in the in-differentiability framework. Keeccak proved permutation-based framework could be the best alternative for the block-ciphers[27],[28],[30].

4.5.1 The Sponge Construction

The sponge function takes a variable-length input string and it produces an arbitrary-length output. The sponge function first pads a sequence of an x-bit block to the message M. The length of the padding would be decided upon the length of the message and the block length. The padding would be considered as sponge compliant if it satisfies the following condition.

$\forall n \geq 0, \forall M,M' \in Z_2^* : M \neq M' \Rightarrow M||pad[r](|M|) \neq M'||pad[r](|M'|)||0^{nr}$

The permutation function f works on the fixed-state bits of width d. The state bits are initialized to zero. The padded input message is cut into blocks size of r-bits. The function f is then interleaved and it is sued to the message at two phases called absorbing phase and squeezing phase. In the absorbing phase the r bits of the state bits are XORed with r bits of message. The output of the function f would serve as the state bit for the next interleaved function. This would be extended until all the r-bit blocks in the input message M are processed. Figure 7 illustrates the operating principle of the SHA-3 digest function. In the Squeezing phase, the function f returns first r bits of the state bits. The final hash value would continue to be

the chain of all the outputs of the interleaved function f.

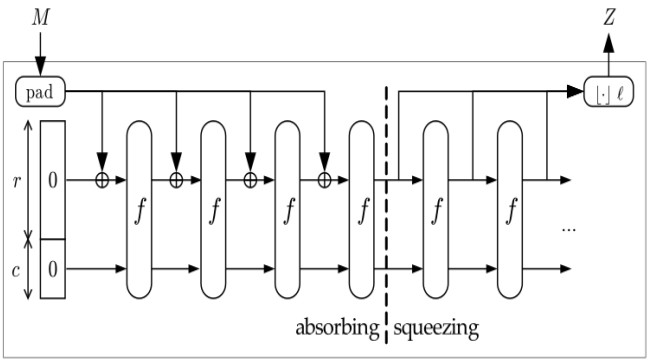

Figure 7: *SHA-3 Operating Principle*

5. CRYPTOGRAPHIC ATTACKS ON DIGEST FUNCTION

The security of the digest function depends on how the algorithm withstands against the attacks. This section presents the various attacks against cryptographic algorithms. Preneel[17],[18] presented various attacks on hash functions under five categories. Goldwasser et al [19] advocated chosen ciphertext attack under two categories. This work focuses on generic attacks and chosen-ciphertext attacks.

5.1 Generic attacks

Generic attacks are launched against all digest functions. This section focuses only on significant generic attacks like Random attack, Exhaustive key-search, and Birthday attack.

5.1.1 Random Attack - In this attack, the length of the hash output establishes the security of the algorithm. If n represents the length of the hash output, then the probability for an attacker to make a successful attack would be $1/2^n$. This implies that more the number of output bits, less the possibility for an attacker to establish a successful attempt. The number of output bits chosen for a hash function depends on the type of the target system in which the hash function is deployed. For example, in the MAC attack, the attacker has to perform his/her trial online that is often subjected to wrong trails threshold. In such systems, it is sufficient enough to gain 32 bits to achieve the required security. Contrarily the attack on Merkel-Damgard construction could be performed offline and there would not be any threshold limit for wrongful trials. Therefore, the MD construction demands at least 64 bits to ensure security. Bart Preneel suggested MD hash functions should at least gain 70 bits, to ensure security[17],[18].

5.1.2 Exhaustive Key Search - This method could be applied to symmetric key hash functions. At this place, the attacker has many plaintexts, MAC code pairs i.e. {M, H}. For every $m \in M$, $H(m) \in H$ pair, the attacker applies brute force to eliminate unwanted keys from the key-space K.

5.1.3 Birthday Attack - This attack applies the probability theory of birthday problem and was coined by Yuval for the first time during 1979[20]. He proved the attack could be applied on any hash system susceptible to possible collisions. The birthday paradigm could impair the hash system with half the trails than the Brute force attack. Therefore, this attack enables the cryptanalyst to drastically reduce the unsuccessful trials to detect collision on any cryptographic hash functions. Bellare et al[21] proved that, if q queries are applied on a cryptographic function of output size n, then the probability for making a collision is $q^2/2^n$. Therefore to prevent a birthday attack, the output size of the hash function should be substantially increased.

5.2. Chosen Ciphertext Attack

The Chosen ciphertext attacks target specific hash functions after identifying their weakness. Goldwasser et al classified this attack in an increasing order based on its severity. The chosen ciphertext attack is classified into two significant categories. They are entered as follows.

5.2.1 Key Only Attack - At this point, the attacker is given only the signer public key. He would be allowed to perform trails to detect the private key.

5.2.2 Message Attack - In this attack, the cryptanalyst could be able to probe the keys for some of the known as well as chosen messages. The following message attacks could be performed under this category.

5.2.3 Known Message Attack- In this attack, the attacker cognizes the signature of the known messages. However, he is denied to choose a message.

5.2.4 Generic Chosen Message Attack- At this point, the attacker is allowed to choose the messages and also he is allowed to collect the signatures for the chosen messages.

5.2.5 Directed Chosen Message Attack- At this point, the cryptanalyst chooses the signature first, before accessing the messages.

5.2.6 Adaptive Chosen Message Attack- In this attack, the attacker considers the signer as an RO. He would make queries to the signer for the message signatures correspond to the signer's public key. The choice of the query is frequently based on the previously obtained message signatures. This attack is more vicious for the reason that, it involves both the signer and the attacker cooperatively work for breaking the target hash function.

6. SECURITY ANALYSIS OF DIGEST FUNCTIONS

The digest functions are designed to prevent hash collisions. Any function that does not satisfy this property would be excluded from cryptographic use. Wang et al[31] proved collisions on MD4, MD5, HAVAL, and RIPEMD. Klima[32] applied tunnels to produce collision on MD5 hash algorithms. Tao Xie et al demonstrated MD5 collisions using a single block[33]. Wang et al demonstrated collisions on SHA-0 and SHA-160 algorithms[34,35]. SHA-2 had been unreported fully broken, but Khovratovich et al [36] proved they had violated 80% of the security of the SHA-256 algorithm by penetrating up to 52 rounds. The SHA3 had been reported partially broken up to 5 rounds[38]. The functional analysis of the standard digest functions is presented in Table 1. The analysis of Table 1 entries reveal the following facts.

1. The security of the digest function relies upon the size of the digest.
2. The attacks/partial attacks reveal the fact that the attacks are independent of the type of construction used.
3. The table entries prove the attacks are owing to bitwise operators like AND, OR, ROT, NOT, MOD, CEIL, TRUNK(x), and XOR operators.

S.NO	NAME OF THE DIGEST FUNCTION	INPUT BLOCK SIZE bits	HASH O/P SIZE bits	NO OF STATE VECTOR S USED	SIZE OF EACH STATE VECTOR Bits	NO. OF ROUND S PERFOR MED	BINARY OPERATIONS INVOLVED	HASH COLLISIO N REMARK S (F/N)	RO/SPONG E CONSTRU CT
1	MD5	512	128	4	32	64	^, | , Rot, $\oplus$, <<<, $\boxplus^{32}$	Found	RO
2.	SHA-0	512	160	5	32	80	^, | , Rot, $\oplus$, <<<, $\boxplus^{32}$	Found	RO
3.	SHA-1	512	160	5	32	80	^, | , Rot, $\oplus$, <<<, $\boxplus^{32}$	Found	RO
4.	SHA-2 224	512	224	8	32	64	^, | , Not, Rot, $\oplus$, <<<, $\boxplus^{32}$	Partial	RO
5.	SHA-2 256	512	256	8	32	64	^, | , Not, Rot, $\oplus$, <<<, $\boxplus^{32}$	Partial	RO
6.	SHA-2 384	1024	384	8	64	80	^, | , Not, Rot, $\oplus$, <<<, $\boxplus^{64}$	Partial	RO
7.	SHA-2 512	1024	512	8	64	80	^, | , Not, Rot, $\oplus$, <<<, $\boxplus^{64}$	Partial	RO
8.	SHA-2 512/224	1024	224	8	64	80	^, | , Not, Rot, $\oplus$, <<<, $\boxplus^{64}$	Partial	RO
9.	SHA-2 512/256	1024	256	8	64	80	^, | , Not, Rot, $\oplus$, <<<, $\boxplus^{64}$	Partial	RO
10.	SHA-3 512	576	512	25	64	24	^, Not , Rot, $\oplus$, <<<	Partial	Sponge

Table 1: Comparative study on the functional analysis of standard digest functions

7. DISCUSSION

1. The Cryptographic hash function takes arbitrary length input string m and produces fixed-length output n. In most of the cases the value of m> n. Therefore, this design paradigm itself would cause a collision in accordance with the Pigeon-hole principle. To design a digest function a model is needed that maps the arbitrary length input string into the fixed-size output string. The mapping would be expected to be one to one. But designing such model mathematically on a compression function would remain an impossible task. This is because the mapping could be done only when input and output boundaries are known. However, the cryptographic functions try including an erratic behavior in the hash output to prevent a cryptanalyst neither to perform differential analysis nor to set mathematical conditions. The erratic behavior would ultimately prevent the hash collisions.
2. A digest function takes an arbitrary length input string and produces fixed-length output. But, if we consider the cryptographic hash functions like MD4, MD5, SHA-160, and SHA2 family, they would reserve the last 64/128 bits for storing the length attribute during the MD-Strengthening. Therefore, these algorithms could handle any input string of size < $2^{64}/2^{128}$ bit. The aforesaid algorithms could not produce digests when the size of an input string is > $2^{64}/2^{128}$ bit. Therefore, they fail to handle arbitrary size input messages in reality.
3. The other issue in the design of the hash function is proof of confirmation. The solitary way to prove the collision-free system is to engage Brute-force. But there is no concrete system present to confirm the claim made by the cryptographic functions using Brute-force. Most of the proofs are theoretical and could not be considered as concrete proof. For example, Merkle and Damgard proved any algorithm that uses the RO model, would not produce a collision. The claim was falsified by Coron et al in his work. At present, the MD5 collisions could be absolutely produced with a notebook PC within a minute. The SHA-3 algorithm manages Sponge construction as a replacement for RO. It was reported to be broken up to 5 rounds. Maybe on some other day, the claim would be falsified by embodying the hash collisions on SHA-3 digest function.

8. CONCLUSION

The Cryptographic digest functions provide a comprehensive solution for the integrity issues of the remote data and to be specific, knowing the integrity violations of the remote data. If the integrity violations left unnoticed then, it would possibly produce catastrophic effects in the life of a person or in the business involved. The hash or digest functions be functional to get rid of the previous situation. But choosing a suitable digest function for an application remains a complex task. This work attempts providing a comprehensive functional analysis of the standard digest functions in the perspective of security. It attempts reporting the significant parts of the design of a digest function like security attributes, Crypto attacks, operating principles, construction design, and the unsolved security issues of the keyless hash functions. We believe this work would help the research community in identifying the significant attributes of the digest function which are essentially to be addressed in the perspective of security.

REFERENCES

[1] Bartkewitz, Timo. "Building hash functions from block ciphers, their security and implementation properties." Ruhr-University Bochum (2009).

[2] Preneel, Bart, René Govaerts, and Joos Vandewalle. "Hash functions based on block ciphers: A synthetic approach." Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 1993.

[3] Al-Kuwari, Saif, James H. Davenport, and Russell J. Bradford. "Cryptographic hash functions: recent design trends and security notions." (2010): 133-150.

[4] Menezes, Alfred J. "van Oorschot, Paul C. Vanstone, Scott A." Handbook of applied cryptography (1996).

[5] Webster, A. F., and Stafford E. Tavares. "On the design of S-boxes." Conference on the theory and application of cryptographic techniques. Springer, Berlin, Heidelberg, 1985.

[6] Kam, John B., and George I. Davida. "Structured design of substitution-permutation

encryption networks." IEEE Transactions on Computers 10 (1979): 747-753.
[7] Matsui, Mitsuru. "Linear cryptanalysis method for DES cipher." Workshop on the Theory and Application of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1993.
[8] FIPS, NIST. "198: The keyed-hash message authentication code (HMAC)." National Institute of Standards and Technology, Federal Information Processing Standards(2002): 29.
[9] Damgård, Ivan Bjerre. "Collision free hash functions and public key signature schemes." Workshop on the Theory and Application of of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1987.
[10] Dewangan, Chandra Prakash, et al. "Study of avalanche effect in AES using binary codes." IEEE Conference on Advanced Communication Control and Computing Technologies (ICACCCT). 2012.
[11] Bellare, Mihir, and Phillip Rogaway. "The exact security of digital signatures-How to sign with RSA and Rabin." International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1996.
[12] Bellare, Mihir, and Phillip Rogaway. "Random oracles are practical: A paradigm for designing efficient protocols." Proceedings of the 1st ACM conference on Computer and communications security. ACM, 1993.
[13] Bellare, Mihir, and Phillip Rogaway. "Optimal asymmetric encryption." Workshop on the Theory and Application of of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1994.
[14] \Canetti, Ran, Oded Goldreich, and Shai Halevi. "On the random-oracle methodology as applied to length-restricted signature schemes." Theory of Cryptography Conference. Springer, Berlin, Heidelberg, 2004.
[15] Cannetti, R., O. Goldreich, and Shai Halevi. "The random oracle methodology, revisited (preliminary version)." Proc. 30th Annual ACM Symp. On Theory of Computing, Perugia, Italy, ACM Press. 1998.
[16] Coron, Jean-Sébastien, et al. "Merkle-Damgård revisited: How to construct a hash function." Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 2005.
[17] Preneel, Bart. Analysis and design of cryptographic hash functions. Diss. Katholieke Universiteit te Leuven, 1993.
[18] Preneel, Bart. "The state of cryptographic hash functions." School organized by the European Educational Forum. Springer, Berlin, Heidelberg, 1998.
[19] Goldwasser, Shafi, Silvio Micali, and Ronald L. Rivest. "A digital signature scheme secure against adaptive chosen-message attacks." SIAM Journal on Computing 17.2 (1988): 281-308.
[20] Yuval, Gideon. "How to swindle Rabin." Cryptologia 3.3 (1979): 187-191.
[21] Bellare, Mihir, and Tadayoshi Kohno. "Hash function balance and its impact on birthday attacks." International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2004.
[22] Damgård, Ivan bjerre. "a design principle for hash functions." conference on the theory and application of cryptology. Springer, new york, ny, 1989.
[23] Merkle, Ralph c. "one way hash functions and des." conference on the theory and application of cryptology. Springer, new york, ny, 1989.
[24] Lai, Xucjia, and James L. Massey. "hash functions based on block ciphers." workshop on the theory and application of of cryptographic techniques. Springer, berlin, heidelberg, 1992.
[25] Rivest, ronald. The md5 message-digest algorithm. No. Rfc 1321. 1992.
[26] Eastlake 3rd, d., and Paul jones. Us secure hash algorithm 1 (sha1). No. Rfc 3174. 2001.
[27] Bertoni, Guido, et al. "Keccak sponge function family main document." Submission to NIST (Round 2) 3.30 (2009).
[28] Bertoni, Guido, et al. "Keccak." Annual international conference on the theory and applications of cryptographic techniques. Springer, Berlin, Heidelberg, 2013.
[29] NIST, Third-round report of the SHA-3 cryptographic hash algorithm competition (November 2012), http://dx.doi.org/10.6028/NIST.IR.7896
[30] Bertoni, Guido, et al. "Duplexing the sponge: single-pass authenticated encryption and other applications." International Workshop on Selected Areas in Cryptography. Springer, Berlin, Heidelberg, 2011.
[31] Wang, Xiaoyun, et al. "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD." IACR Cryptology ePrint Archive 2004 (2004): 199.
[32] Klima, Vlastimil. "Tunnels in Hash Functions: MD5 Collisions Within a Minute." IACR Cryptology ePrint Archive 2006 (2006): 105
[33] Xie, Tao, and Dengguo Feng. "Construct MD5 Collisions Using Just A Single Block Of Message." IACR Cryptology ePrint Archive 2010 (2010): 643.
[34] Wang, Xiaoyun, Yiqun Lisa Yin, and Hongbo Yu. "Collision search attacks on SHA1." (2005).
[35] Wang, Xiaoyun, Yiqun Lisa Yin, and Hongbo Yu. "Finding collisions in the full SHA-1." Annual international cryptology conference. Springer, Berlin, Heidelberg, 2005.
[36] Khovratovich, Dmitry, Christian Rechberger,

and Alexandra Savelieva. "Bicliques for preimages: attacks on Skein-512 and the SHA-2 family." Fast Software Encryption. Springer, Berlin, Heidelberg, 2012.

[37] https://es.cs.uni-kl.de/research/applications/sha2/

[38] Chang, Shu-jen, et al. "Third-round report of the SHA-3 cryptographic hash algorithm competition." NIST Interagency Report 7896 (2012): 121.