Cryptographic Hash Functions: A Review

Article *in* International Journal of Computer Science Issues · March 2012

CITATIONS
97

READS
26,211

2 authors:

Rajeev Sobti
Lovely Professional University
27 PUBLICATIONS **205** CITATIONS
SEE PROFILE

Geetha Ganesan
Lovely Professional University
65 PUBLICATIONS **308** CITATIONS
SEE PROFILE

Cryptographic Hash Functions: A Review

Rajeev Sobti[1], G.Geetha[2]

**[1]School of Computer Science, Lovely Professional University
Phagwara, Punjab 144806, India**

**[2]School of Computer Applications, Lovely Professional University
Phagwara, Punjab 144806, India**

Abstract

Cryptographic Hash functions are used to achieve a number of security objectives. In this paper, we bring out the importance of hash functions, its various structures, design techniques, attacks and the progressive recent development in this field.

Keywords: *Cryptography, Hash function, compression function*

1.Introduction

Cryptographic techniques mainly encryption & decryptions have been used for centuries to protect military and political secrets and D.Kahn in [1] has given comprehensive study of this history. Throughout this history of cryptology, confidentiality has taken the primary seat and it was believed that if the secrecy is maintained (using symmetric encryption and secret key) then the authentication will automatically be achieved. The logic was if decryption of an encrypted text results in a meaningful message it must have been constructed by someone who knows the secret key. During all this period the field of cryptology was kingdom of selected few i.e. it was studied and practiced by few. The trend changerswereDiffie and Hellman, who are credited for advent of public key cryptography in mid 70s. Their seminal paper "New Directions in Cryptography" [2] introduced a number of relevant concepts like Digital Signatures and differentiated Confidentiality from Authentication and to quite an extent initiated the development of cryptographic schemes for the protection of authenticity. These schemes use a very important cryptographic primitive named 'Cryptographic Hash Functions'. However cryptographic hash functions have received much less attention from the cryptologic community than encryption schemes in the past. Bert Rompay in his thesis [3] quoted the example of NESSIE (New European Scheme for Signature Integrity and Encryption) project to illustrate how cryptographic hash functions have been ignored in the past. In NESSIE project,seventeen block ciphers and six stream cipherswere submitted as candidates (both are categories of encryption schemes), but only one un-keyed hash function and two keyed hash functions (also known as MAC – Message Authentication Code) were submitted,. Rompay [3] also gave example of opencompetition used by the National Institute of Standards and Technology (NIST)in the United States to decide on the block cipher to be used as Advanced Encryption Standard. This competition had fifteen candidates out of which theRijndael [7] block cipher finally chosen. On the other hand, for its hash function standard [6] NIST simply chose the SHA hash functions, designed bythe NSA without disclosure of their design strategy or any supporting cryptanalytic results. However the trend has changed in recent years because of the wide range of applications areas of cryptographic hash functions. Cryptographic Hash Functions are used to achieve a number of Security Goals like Message Authentication, Message Integrity, and are also used to implement Digital Signatures (Non-repudiation), Entity Authentication and Digital Steganography. Considerable research has been undergoing in the field of Cryptographic Hash Functions. Hash Functions are being generated from existing primitives like Block ciphers (e.g. Whirlpool [84], Skein [66]) as well as being explicitly and specially constructed from scratch like MDx family [9, 10] and SHA family [4,5,6,8] of hash functions.

Organization of the paper: This paper will present the detailed study of Cryptographic Hash Functions. Organisation of the paper is as follows. In Section 2 and 3 the basic concepts like definitions, properties and applications of Hash functions are detailed.Section 4 discusses the basic as well as currently used iterative structures of Hash functions. In Section 5 and 6 security properties and possible attacks are detailed. In Section 7 various design techniques of underlying compression functions have been explained. Section 8 throws light on the current scenario in Hash functions.

2. Cryptographic Hash Functions

The term hash function has been used in computer science from quite some time and it refers to a function that compresses a string of arbitrary input to a string of fixed length. However if it satisfies some additional requirements (as detailed further), then it can be used for cryptographic applications and then known as Cryptographic Hash functions.

Cryptographic Hash functions are one of the most important tool in the field of cryptography and are used to achieve a number of security goals like authenticity, digital signatures, pseudo number generation, digital steganography, digital time stamping etc. Gauravram [16] in his thesis has suggested that the usage of cryptographic hash functions in several information processing applications to achieve various security goals is much more widespread than application of block ciphers and stream ciphers.

Rompay [3] has given the following formal definition of hash functions

Definition: *A* ***hash function*** *is a function* ***h****:* $D \rightarrow R$,

where the domain $\mathbf{D} = \{0,1\}^*$ *and* $\mathbf{R} = \{0,1\}^n$ *for some* $n >= 1$ (1)

Cryptographic Hash Functions are broadly of two types i.e. Keyed Hash functions; the one which uses a secret key, and Un-keyed Hash Functions; the other one which does not uses a secret key. The keyed Hash functions are referred to as Message Authentication code. Generallythe term hash functions refer to un-keyed hash functions and **in this paper we will concentrate on Un-keyed Hash functions only.** Un-keyed or simply Hash functions(some time also known as MDC – Manipulation Detection Code)can further classified into OWHF (One Way Hash Functions), CRHF (Collision Resistant Hash Functions) and UOWHF (Universal One way Hash Functions) depending on the additional properties it satisfies.

2.1 One Way Hash Functions (OWHF)

OWHF as defined by Merkle [11] is a hash function H that satisfies the following requirements:

I. H can be applied to block of data of any length. (In practice, 'any length' may be actually be bounded by some huge constant, larger than any message we ever would want to hash.)
II. H produces a fixed-length output.
III. Given H and x (any given input), it is easy to computer message digest H(x).
IV. Given H and H(x), it is computationally infeasible to find x.
V. Given H and H(x), it is computationally infeasible to find x and x' such that H(x) = H(x')

The first three requirements are must for practical applications of a hash function to message authentication and digital signatures. The fourth requirement also known as **pre-image resistance or one way property**, states that it is easy to generate a message code given a message but hard (virtually impossible) to generate a message given a code. The fifth requirement also known as **Second pre-image resistance**property guarantees that an alternative message hashing to the same code as a given message cannot be found.

2.2 Collision Resistant Hash Functions (CRHF)

One of the early definitions of Collision Resistant Hash functions was given by Merkle [12]. Based on the same, CRHF may be defined as a Hash function H, that satisfies all the requirements of OWHF (I to V as listed in 2.1) and in addition satisfy the following **collision resistance** property:

Given H, it is computationally infeasible to find a pair (x, y) such that H(x) = H(y)

2.3Universal One Way Hash Functions (UOWHF)

Mani Naorand Moti Yung [13] presented the idea of Universal One Way Hash functions and using the same, presented a digital signature scheme that was not based on trapdoor functions. Rather Mani Naorand Moti Yung [13], used 1-1 One way functions to construct UOWHF and in turn implement Digital Signature scheme.The Security property of UOHWF as described in [13] is reframed as follows:

Let U contains a finite number of hash functions with each having the same probability of being used. Let a probabilistic polynomial time algorithm A (A is collision adversary) operates in two phases. Initially, A receives input k and outputs a value x known as initial value, then a hash function H is chosen from the family U. A then receives H and must output ***y*** *such that H(x) = H(y). In other words, after getting a hash function it tries to find a collision with the initial value. Now U will be called as a family of Universal One Way Hash Functions if for all polynomial-time A the probability that A succeeds is negligible.*

"How to construct UOWHF of higher orders efficiently?" is still as unsolved problem in cryptography.

3. Security Services of Cryptographic Hash Functions

3.1 Achieving Integrity & Authentication

Verifying the integrity and authenticity of information is a prime necessity in computer systems and networks. In particular, two parties communicating over an insecure channel require a method by which information sent by one party can be validated as authentic (or unmodified) by the other. [17]

Message Integrity & Authentication may be implemented in multiple ways. Symmetric Encryption based mechanisms may be used but they have their own drawbacks. Drawbacks like speed, cost factor, optimization for data sizes etc. have been highlighted by Tsudik [18]. Such methods combine the Confidentiality and Authentication functions. However there are scenarios where encrypting full message (confidentiality) is not required. For such applications keeping message secret is not the concern but authenticating it is important. For example in SNMP (Simple Network Management Protocol), it is usually important for a managed system to authenticate incoming SNMP commands (like changing the parameters at the managed system), but concealing the SNMP traffic is not required.

In order to implement message authentication and integrity, the alternative techniques (other than the methods mentioned in last paragraph) are *MAC* or *hash functions*. MACs may be constructed out of block ciphers like DES. More recently, however, there has been a surge of interest in the idea of constructing MACs from cryptographic Hash Functions [17]. In addition to using Hash Functions for implementing MAC, Hash functions can be used to achieve message authentication and integrity goals without the use of symmetric encryption. Tsudiac [18] has detailed a protocol based on the same idea.Rompay [3] has also detailed the ways of ensuring authentication using hash functions alone as well as using hash functions with encryption. The usage of Hash Functions for Message Authentications and ensuring message integrity has surged because majority of hash functions are faster than block ciphers in software implementation and these software implementations are readily and freely available [17].

3.2 Implementing Efficient Digital Signatures

Digital signature is a security goal of a cryptosystem which intends to achieve the goal of authenticity and a security service or property of non-repudiation [16]. MAC and Hash Functions alone do not implement the Security goal of Digital Signatures. It was Diffie and Hellman [2] who first realised the need for a message dependent electronic signature (fingerprint) to avoid disputes between sender and receiver. RSA [19] was the first public key crypto systems with digital signature capabilities. However there has been an interesting part of this invention. James Ellis, Clifford Cocks and Malcolm Willaimson from GCHQ (Government Communication Head Quarters), Cheltenham, Britain perhaps invented the idea of Public key in 1972. The three Britons had to sit back and watch as their discoveries were rediscovered by Diffie, Hellman, Merkle, Rivest, Shamir and Adleman over the next three years because of the polices of GCHQ that all work is top secret and cannot be shared with anyone [20].

Hash functions are used to optimize the digital signature schemes. Without the use of Hash, the signature will be of same size as message. The fundamental concept here is instead of generating the signature for the whole message which is to be authenticated; the sender of the message only signs the digest of the message using a signature generation algorithm. The sender then transmits the message and the signature to the intended receiver. The receiver verifies the signature of the sender by computing the digest of the message using the same hash function as the sender and comparing it with the output of the signature verification algorithm. It is obvious that this approach saves a lot of computational overhead involved in signing and verifying the messages in the absence of hash functions [16].

3.3 Authenticate Users of Computer Systems

Hash functions may be used to authenticate the users at the time of login. The passwords are stored in the form of message digest to avoid access of the same even to Database Administrators (because of Pre-Image resistance ofHash digest). Whenever user tries to login and enter the password, the message digest of the entered password is computed and compared with the digest stored in the database. If it matches, then login is successful, otherwise user is not authenticated.

3.4 Digital Time Stamping

Majority of text, audio and video documents are available in digital format and a number of simple techniques and tools are available to change digital documents. So some sort of mechanism is required to certify when such a document was created or last modified. Digital timestamp solve the purpose and provide a temporal authentication Rompay [3] in his thesis work has suggested the multiple ways like simple scheme based on trusted third party, scheme that links timestamps into temporal chain and the otherone that make use of Merkle Tree. Rompay [3] highlighted that Digital time stamp helps in protecting intellectual property rights, ensuring strong auditing procedures and implementing true non-repudiation services. Before [3], Haber and Stornetta [21] has also detailed how One way hash functions and digital signatures can be used to implement the digital time stamping.

3.5 Hash functions as PRNG

Hash functions as one way functions can be used to implement PRNG (Pseudo random number generator). A very simple technique can be to start from an initial value (s) known as seed and computer H(s) and then H(s+1), H(s+2) and so on. [22, 23] has given some other ways of constructing Pseudo random strings from Hash functions.

3.6 Session Key Derivations

Hash functions as one way functions can be used to generate sequence of session keys that are used for the protection of successive communication sessions. Starting from a master key K0, the first session key can be *K1 = H(K0)* and second session key can be *K2 = H(K1)* and so on. Matyas*et.al.*[24] described the key management scheme based on control vectors which makes use of hash functions and Encryption functions for generating session keys.

3.7 Constructions of Block Ciphers

Block ciphers can be used to construct a cryptographic hash function however the inverse is also true and there has been block ciphers designed using Hash functions. In [25] Handschuh and Naccache proposed to use the compression function of cryptographic hash function SHA-1 [5] in encryption mode. The name of the cipher was SHACAL. SHACAL-1 (originally named SHACAL) and SHACAL-2 are block ciphers based on SHA-1 [5] and SHA-256 [6] respectively. SHACAL-1 (originally named SHACAL) is 160-bit clock cipher and SHACAL-2 is 256 bit block cipher. Both were selected for the second phase of NESSIE project. In 2003 SHACAL-1 was not recommended for NESSIE portfolio because of concerns about its key schedule, while SHACAL-2 was finally selected as one of the 17 NESSIE finalists.SHACAL-1 used the compression function of SHA-1 and turned it into a block cipher by using the state input as the data block and using the data input as the key input. In other words SHACAL-1contemplated the SHA-1 compression function as an 80-round, 160-bit block cipher with a 512-bit key. Keys shorter than 512 bits are supported by padding them with zero up to 512. SHACAL-1 was not intended to be used with keys shorter than 128-bit.

3.8 Other Applications

Hash Functions can also be used to index data in hash tables, for fingerprinting, to detect duplicate data or uniquely identify files, and as checksums to detect accidental data corruption and for generating random numbers also.

Looking at this wide range of applications, it is not correct to say thatHash Functions belong to one particular cryptographic sub branch. These cryptographictools deserve a separate status for themselves. They are used in almost all placesin cryptology where efficient information processing is required.

4. Iterative Structure of Hash Functions

4.1 MerkleDamgard Iterated Hash Design

At Crypto '89, Ivan Damgard [26] and Ralph Merkle [12] independently proposed the iterative structure to construct a collision resistant hash function using fixed length input collision resistant compression function. Both independently provided proofs in their papers [12 and 26] that if there exists a fixed length collision resistant compression function: f: $\{0,1\}^a$ X $\{0,1\}^b \rightarrow \{0,1\}^c$ then one can design a variable length input collision resistant hash function H: $\{0,1\}^* \rightarrow \{0,1\}^n$, by iterating that compression function. Originally named "Merkle's Meta Method", this scheme is now mostly calledthe Merkle-Damgard construction.Lai and Massey [27] named such a structure as *Iterated Hash Structure*.

Rompay [3] has given the following formal definition of Compression function, Output transformation and Iterated Hash functions.

Definition: *A* ***compression function*** *is a function f : D* $\rightarrow$ *R where D =* $\{0,1\}^a$ *X* $\{0,1\}^b$ *and R =* $\{0,1\}^c$ *for some a,b,c>=1 , and a + b >= c.* (2)

Definition: *An* ***output transformation*** *is a function g : D* $\rightarrow$ *R where D =* $\{0,1\}^a$ *and R =* $\{0,1\}^n$ *for some a, n >=1 and a>=n .* (3)

Definition:*Suppose that a compression function f :* $\{0,1\}^c$ *X* $\{0,1\}^b \rightarrow \{0,1\}^c$ *and an output transformation* $\{0,1\}^c \rightarrow \{0,1\}^n$ *are given. Then an* ***iterated hash function*** *is the hash function h :* $(\{0,1\}^b)^* \rightarrow \{0,1\}^n$ *defined by* $h(X_0, X_1, .. X_{t-1}) = g(H_t)$ *where* $H_{i+1} = f(H_i, X_i)$ *for 0<=i<t. The input block* X_i *(0<=i<t) =* $\{0,1\}^b$ *and Initial chaining value* $H_0 = IV \in \{0,1\}^c$ (4)

As per the definition the block length is *b* bits and chaining variable length is *c* bits long. In case the input string is not an exact multiple of *b* bits then some sort of padding is used. The padding technique has varied from one algorithm to another. However the general convention is to pad the input strings with bit 0 followed by sequence of bit 1 and at the end append the length of message such that after all the padding (bit 0, sequence of 1s and the message length), the total length of the padded message is exact multiple of *b* bits (block length). The length of message is padded to avoid a particular type of attack named as fixed point attack. The output transformation is required when the message digest size required is less than the size of chaining variable i.e. n < c. In case n = c, then output transformation can be ignored. Wherever output transformation is required, it can be implemented by just selecting c bits out of n or using some folding techniques.

Merkle [12] and Damgard [26] suggested that if IV is not fixed then finding second pre-image or collision is trivial and also if length is not padded then attacks based on fixed points can be used to break iterated hash structure. Both independently provided proof that

if IV is fixed as well as length padding is used then hash function will be collision resistant if compression function is collision resistant. The process of fixing IV and adding length padding is known as MD-strengthening.

Majority of Hash Functions launched in recent years and being used these days follow the iterated hash function. MD4 [9], MD5 [10], SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 [4, 5, 6,8] all are influenced by the Merkle and Damgard's iterated hash design as explained above.

MerkleDamgard construction as explained above has some drawbacks like it suffer from some generic attacks (to be discussed in Section 5 and 6) JouxMulticollision [37], Herding attacks [38], Length Extension attacks [39] etc. Because of these structural weaknesses, some other constructions have been suggested in literature. Few of these are:

4.2 Wide Pipe Iterated Hash Design

Mainly because of length extensions &JouxMulticollisions[37], Stefan Lucks [36] proposed an improvement over MerkleDamgard(MD) structure named 'Wide Pipe Iterated Hash Design'.Wide pipe design is quite similar to MD design, but it has larger internal state size. Lucks [36] suggested that Joux [37] and length extension are mainly based on Internal collisions and internal collisions can be avoided if we widen the internal pipe from n bits to $w >= n$ bits. If a hash of n bits is desired, then two compression functions $f_1$andf_2 will be required:
-- f_1: $\{0,1\}^w$ X $\{0,1\}^m \rightarrow \{0,1\}^w$
-- f_2: $\{0,1\}^w \rightarrow \{0,1\}^n$
Then wide pipe iterated hash is constructed like follow:
-- for i = 1, …., L : Computer $H_i = f_1 (H_{i-1}, M_i)$
-- Finally Set $H(M) = f_2 (H_L)$
Compression function $f_1$takes w bits (generally w = 2n) of chaining value and m bits of message (M) and compressed this to an output of w bits and in the last another compression function f_2 compresses the last internal hash value (w bits) to the final hash value (n bits). SHA-224 and SHA-384 are based on the same design and are derived from SHA -256 and SHA-512 respectively. In addition to wide pipe, Lucks [36] has also proposed double-pipe hash (twined pipe) design.

4.3 Hash Iterated Framework (HAIFA)

Biham and Dunklermann [41] in 2006 proposed the HAIFA structure to overcome many of the pitfalls observed in MerkleDamgard Construction.The main ideas behind HAIFA are the introduction of number of bits that were hashed so far and a salt value intothe compression functions. Formally, instead of using a compression functionof the formf_{MD} : $\{0,1\}^{m'}$ X $\{0,1\}^n \rightarrow \{0,1\}^{m'}$, Biham and Dunklemann [41] proposed to use f_{MD} : $\{0,1\}^{m'}$ X $\{0,1\}^n$ X $\{0,1\}^b$ X $\{0,1\}^s \rightarrow \{0,1\}^{m'}$, i.e. in HAIFA chaining value H_i is computed as

$$H_i = f (H_{i-1}, M_i, \#bits, salt)$$

where*#bits* is number of bits hashed so far and *salt* is a salt value.For comparison of HAIFA structure with Wide pipe design or other designs refer [41].

4.4 Fast Wide Pipe (FWP) Design

A further improvement of wide pipe design was suggested by Mridul Nandi and Souradyutipaul [40] in 2010. They proposed that FWP was nearly twice as fast as the Wide-pipe for a reasonable selection of the input and output size of the compression function. The idea was that internal state i.e. widepipe chaining value should be divided in two halves. One half is inputted to the succeeding compression function but the other half is combined (XOR) with the output of that succeeding compression function i.e. we feed-forward half of the previous chaining value to XOR it to the output of the compression function.

4.5 Sponge Construction

G. Bertoni*et. al.*[42, 43, 44] proposed sponge construction to design hash functions that closely map the random oracle. In the context of cryptographic hash functions, *sponge functions* provide a particular way to generalize hash functions to more general functions whose output length is arbitrary. G. Bertoni*et. al.* in [42] explained that sponge functions are only distinguishable from random oracles by the detection of innercollisions and the probability of inner collisions can be made arbitrarily small by increasinga security parameter, called the capacity.

As per G. Bertoniet. al. [44] the *sponge construction* is a simple iterated construction for building a function **F** with variable-length input and arbitrary output length based on a fixed-length transformation (or permutation) **f** operating on a fixed number **b** of bits. Here **b** is called the *width*.

The sponge construction operates on a state of b=r+c bits, r is called bitrate and c as capacity. Initially all the b bits of state are set to zero and I/P message is padded and divided into block of r bits each. Then sponge construction proceeds in two phases: Absorbing phase and Squeezing Phase

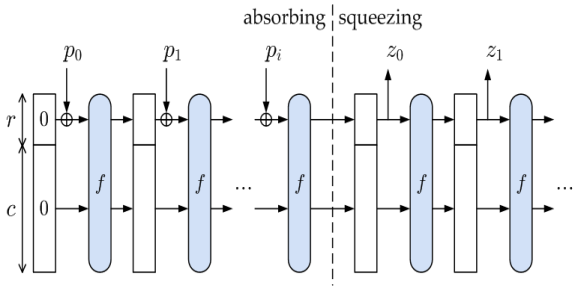

Fig. 1 The sponge construction for hash functions. p_i are input, z_i are hashed output [44]

In first phase input is "absorbed" into the hash state at a given rate, then an output hash is "squeezed" from it at the same rate. To absorb r bits of data, the data is XORed into the leading bits of the state, and the block permutation is applied. To squeeze, the first r bits of the state are produced as output, and the block permutation is applied if additional output is desired.
Central to the Sponge construction is capacity c of hash function and it can be adjusted based on security requirements. SHA-3 [45] final round candidate algorithm Keccak[46] is a hash function based on Sponge construction only and it sets a conservative c=2n, where n is the size of the output hash.

4.6 Other Constructions

In addition to the above listed Iterative Hash constructions, few more like **Enveloped MerkleDamgard**, **RMC construction** and **ROX construction** have been suggested in literature. To know more about these structures refer [41, 52, 53, 54].**Cascaded Constructions** have also been discussedin the literature to build large hash values by concatenating concatenate several smaller hashes. For example, given two hash functions H1 and H2, the concatenation H1(M) || H2(M) can be used to generate large hash value for message M. In this construction, H1 and H2 can either be two completely different hash functions or two slightly different instances of the same hash function. But Joux [37] using multicollisions proved that If H1 and H2 are good iterated hash functions with no attack better than the generic birthday paradox attack, then the large hash function H1|| H2 obtained by concatenating H1 and H2 is not really more secure that H1 or H2 by itself.

5.Security Properties of Hash Functions

5.1 Basic Security Properties

Basic notion of security of Hash functions revolves around ***preimage resistance, second-preimage resistance****and* ***collision resistance***as defined in Section 2.In literature Collision resistance property is referred to as collision freeness or strong collision resistance, second pre-image resistance is called as weak collision resistance and preimage resistance is referred to as one-wayness [16]. It is easy to see that collision resistance implies second-preimage resistance i.e. if a hash function ***h***is collision resistant then ***h***is also second pre-image resistant. However second-preimage resistance and one-wayness are incomparable (the properties do not follow/imply one another), although hash functions which are one-way but not second-preimage resistant are quite contrived. In practice, collision resistance is the strongest property of all three, hardest to satisfy and easiest to breach, and breaking it is the goal of most attacks on hash functions [27].

Rogaway and Shrimpton [14] extended the notion of hash function security and defined seven different security notions, three on pre-image resistance, three on second pre-image resistance and one on collision resistance. The work of Rogaway and Shrimpton [14] is based on generic concept of a hash function family that is a finite set of hash functions with common domain and range. The security of hash function and probability of success of an adversary depends on the manner in which one chooses a particular hash function from the hash function family for example the hash function can be chosen on random or may be fixed element. Based on these variations, seven different security notations and relation between them are given in [14].

5.2 Avalanche Criterion and Completeness

From a good hash function it is desired that for two different inputs, the output of hash function should be completely different, regardless of difference in inputs. The same can be formalised with two properties of hash functions i.e. Completeness and Avalanche effect. **Strong Avalanche effect** represents a property when small change in input result in a significant change in message digests. **Completeness**represents a property when eachinput bit affects all output bits.**Strict Avalanche Criterion** combines both the avalanche effect and thecompleteness and represent a property when a change in one bit of input results in changing every bit of the output (message digest) with a probability of ½. If these **criterions** are not satisfied then the probability of successful attack on the hash functionsincreases considerably.

5.3 Certificational Properties and weaknesses

In addition to basic properties some certificational properties have been defined in literature from time to time. For exampleIlyaMironov [28] and Gauravram [16] suggested *near collision resistance, partial pre-image resistance, free start collision resistance, pseudo collision resistance, semi Free start collision* as certificational properties for hash functions and / or underlying compression functions. Lack of resistance of these properties is termed as certificational weaknesses. Certificational properties for hash functions and compression functions intuitively appear desirable but cannot be shown as necessary properties of hash functions. Certificational weaknesses does not result in breaking a hash function directly but is enough to cast doubt on its design principles and may lead to full collision under certain circumstances.

CertificationalProperties or weaknesses may be defined w.r.t. hash function as a whole or for underlying compression function only. These certificational properties, weaknesses and possible attacks on these properties are briefly touched upon in this section:

5.3.1 Certificational Properties of Hash functions

Near Collision Resistance: A hash function is said to be Near Collision resistant if it is hard to find two messages x and x' such that the hamming distance between h(x) and h(x') is small (typically a few bits).Near collision may also be termed as almost collision and can be defined for underlying compression function also. With respect to underlying compression function, almost / near collision means that two message blocks are found for which the difference between the outputs has a low Hamming weight.Gauravram [16] quoted the example of how near collisions in case of hash functions with truncated outputs can lead to full collision. If we have a truncated hash function that makes use of leftmost 224 bit of output after chopping rightmost 32 bits then if near collision is found such that message digests only in the rightmost 32 bits then such a near collisions are practically full collisions only.

Partial Pre-image resistance: A hash function is said to be partial pre-image resistant if difficult in finding a partial pre-image is same as finding pre-image from a given digest. Also it is hard to find the input if part of the input is known along with digest.

5.3.2 Certificational Properties on the Compression Function

Certificational properties or weaknesses on the compression functions used in the MerkleDamgard structure or similar other iterative structures are classified based on the IV / H_0 (Initial value) used. These classifications and nomenclature has varied from author to author. For example *Pseudo collision resistance*as defined in [47] is termed as *Special pseudo (type-3) collision resistance*in [16]. Similarlyfor an attack, Rompay in [3]has used the nomenclature as *Random IV collision* and for the same attack Gauravram in [16] has used the nomenclature as *Semi free start collision*. Furthermore Mironovin [28] defined *Pseudo Collision resistance* and *Free Start collision resistance* as two separate properties on the other side Gauravrama [16] and Knudsen [48] termed *pseudo collisionresistance* and *free start collisionresistance* as one and the same thing. In this sub section we use the terminology and classification done by Gauravram in [16] as it has been found most exhaustive and clear but at the same time we also list the alternative nomenclature used by different authors.

Type -1 Collision: Type-I collision resistance is not a certificational property but it is discussed here as it related to other certificational properties based on initial value. Type-I collision refers the collision in a compression function using an IV (initial value) specified in the specification of the hash functions for two distinct messaged. Corresponding property may be defined as: it is hard to find two messages X and X' for compression function f: $\{0,1\}^n$ X $\{0,1\}^m \rightarrow \{0,1\}^n$ such that f(H,X) = f (H, X') , where H represents the initial value (IV) specified in the specification of hash function. Type-1 collision is also referred to as strong collision.

Type – 2 Collision: Type – 2 collision resistanceis also termed as Random IV Collision resistance [3] or Semi Free Start collision resistance [16]. Type-2 collisions are the collisions using the same random (or arbitrary) initial values for two distinct message inputs. Corresponding property may be defined as: it is hard to find two messaged X and X' for the compression function f: $\{0,1\}^n$ X $\{0,1\}^m \rightarrow \{0,1\}^n$ such that f(H,X) = f (H, X') , where computation starts from an arbitrary (random) value H for the input chaining variable.

Type – 3 Collision: Type - 3 collision resistance is also termed as Pseudo collision resistance [16] or Free start collision resistance [48]. Type-3 collisions are the collisions of compression function using two different initial values for two distinct message inputs. Corresponding property may be defined as : it is hard to find two pairs (H, X) and (H', X') for compression function f: $\{0,1\}^n$ X $\{0,1\}^m \rightarrow \{0,1\}^n$ such than f(H,X) = f (H',X') such that (H , X) $\neq$ (H', X'). Here H/H' represent initial / intermediate chaining value and X/X' represent message block.

Special Type – 3 Collision: Special Type – 3 collision are the collisions of the compression function using two different initial values on the same message block. Corresponding property may be defined as: it is hard to find two pairs (H, X) and (H, X') for compression function f: $\{0,1\}^n$ X $\{0,1\}^m \rightarrow \{0,1\}^n$ such than f(H,X) = f (H,X') such that X $\neq$ X'. Here H represent initial / intermediate chaining value and X/X' represent message block. Note that [3] and [47] uses pseudo collision resistance to represent the same property. However Gauravram [16] categorised it as a special category of Pseudo collision resistance and named it as Special pseudo collision resistance.

Inner (almost) Collisions: As defined by Rompay [3], these are collisions or almost-collisions for the temporary values of the chaining variable (for two distinct message blocks), at some stage of the compression function (for example after s1 step operations where s1 < s). This may be helpful for an attacker who tries to generate a collision in the output of the compression function.

The collision attacks on compression functions as described above are also applicable on their hash function iterative modes. Type-1 collision attacks are practical one and can be used to attack applications that in turn make use of Type-1 susceptible hash functions. Paper [49] represents such an example. Type-2 or Type -3 attacks are not practical but create doubts on the hash functions. Attacks in paper [47] and [50] are examples of Type-2 or Type-3 attacks. In [47] B. den Boer and A. Bosselaers gave an early, although limited, result of finding a "pseudo-collision" (Type- 3) of the MD5 compression function; that is, two different initialization vectors which produce an identical digest. In [50] H. Dobbertin published an attack (Type-2), without details, that found a collision in MD5 with an IV (Initial value) chosen by him that was different from the one actually used in MD5 . While this was not an attack on the full MD5 hash function, it was close enough for cryptographers to recommend switching to a replacement, such as SHA-1. However attacks in [31] and [32] are Type-1 attacks.

6. Methods of attack on Hash Functions

Attacking a hash function means breaking one of the security properties (basic, extended or certificational property) of hash functions. For example breaking pre-image resistance means adversary is able to break the pre-image property i.e. an adversary is able to create a message that hashes to a specific hash. Breaking certificational properties may not yield a practical attack but are an important warning to reflect weakness in the hash / compression function. Gauravram [16] recommended switching to a strong hash function when an attack on certificational properties is observed. In an iterated hash function, if a pre-image or collision (Type-1 collision only) can be found for compression function (f), the same can be extended and an attack on hash function can be derived. So attacks may focus on structure of hash function or on algorithm of compression function.In this sub section we will review different types of attacks on hash functions.Attacks on Hash functions can be classified into two broad categories - Brute Force Attacks and CryptanalyticalAttack.

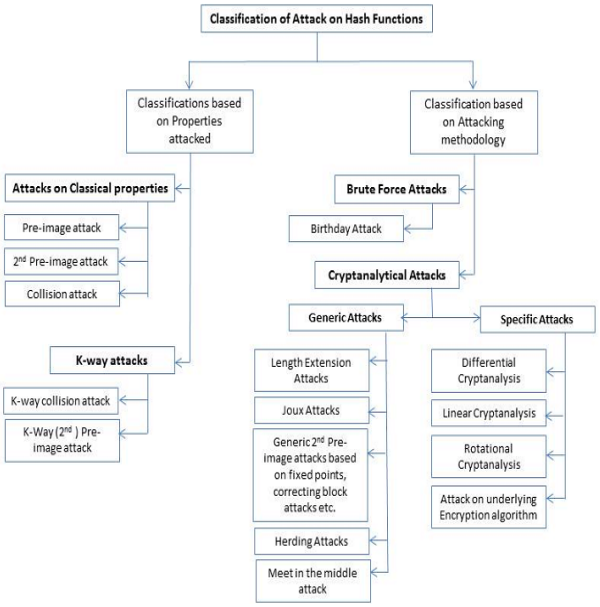

Fig. 2 Classification of attacks on Hash Functions

6.1 Brute Force Attack

Brute force attacks work on all hash functions independent of their structure andany other working details. They are similar to exhaustive search or brute-forcekey recovery attacks on the encryption schemes to extract the secret key of the encryption scheme. The security of any hash function lies in its output bit size. For a hash code of length n, the level of effort required to resist different brute force classical attacks on hash functions is as follow:

Pre-image attack: Effort required for brute force attack = 2^n. In this attack, for a given n-bit digest h ofthe hash function H( ), the attacker evaluates H( ) with every possible inputmessage M until the attacker obtains the value h.

2nd Pre-image attack: Effort required for brute force attack = 2^n.In this attack, for a given message M andthe hash function H( ), the attacker tries H( ) with every possible input messageM' ≠ M until the attacker obtains the value H(M).

Collision attack: Effort required for brute force attack = $2^{n/2}$. In this attack, for a given hash function H,the attacker tries to find two messages M and M' such that M ≠ M' andH(M) = H(M'). On average the opponent would have to try 2^n / 2 (= 2^{n-1}) messages to find one that matches the hash code of the intercepted message However a chosen plain text attack (based on Birthday Paradox) is possible and in that case the effort required for collision in a Hash function is $2^{n/2}$ in place of 2^{n-1} [29]. It is also referred as Birthday attack.

In addition to the above discussed classical attacks, the following natural extensions have also been studied by different authors.

K-Way Collision attack for K >=2: FindK different messages M^isuch that$H(M^1) = \ldots = H(M^K)$. [36]

K-Way (2^{nd}) pre-image attack for K>=1: GivenY (or M with H(M) = Y), find K different messages M^i, with $H(M^i) = Y$ and $M^i \neq M$.[36]

6.2 CryptanalyticalAttack

Cryptanalysis of Hash functions focuses on the underlying structure of hash function and/or on the algorithm of Compression Function. Due to fixed size of the hash values compared to much larger size of themessages, collisions must exist in hash functions. However, for the security of thehash function, they must be computationally infeasible to find. Collisionsin hash functions are much easier to find than pre-images or 2^{nd} pre-images.

Informally, a hash function is said to be "broken" when a reduced number ofevaluations of the hash function compared to the brute force attack complexitiesand the strengths estimated by the designer of the hash function are used toviolate at least one of its properties immaterial of the computational feasibilityof that effort. For example, assume that it requires 2^{90} evaluations of the hashfunction to find a collision for a 256-bit hash function. Though it is impracticalto generate this amount of computational power today, the hash function is saidto be broken as this factor is less than the 2^{128} evaluations of the hash functionrequired by the Birthday attack. It should be noted that hash functions are easier to attack practically thanencryption schemes because the attacker does not need to assume any secrets andthe maximum computational effort required to attack the hash function is onlyupper bounded by the attacker's resources not users gullibility. This is not thecase with block ciphers where the maximum practical count of executions of theblock algorithm is limited by how much computational effort the attacker can getthe user to do [16].

Collision finding algorithm and attacks may be classified as single block attacks or multi block attacks depending on whether that attack uses single block (i.e. one compression function) or more than one block (i.e more than one iteration of compression function) for finding collision or pre-images.

Gauravaram [16] in his Ph.D. thesis has further classified Cryptanalyticalattacks on hash function in two categories i.e. Generic and Specific attacks.

6.2.1 Generic Attacks

Theattacks that work on a general hash function construction are called genericattacks. For example, attacks on the Merkle-Damgard construction that workon all hash functions designed using MerkleDamgard construction are the generic attacks.Generic attacks are applicable even if we replace the underlying compression function by some abstract oracle. Length extension attacks, Jouxmulticollisionattacks [37], Generic 2^{nd}preimage attacks like the one based on Fixed points, correcting block attack, Herding Attacks and Meet in the Middle attacks are example of Generic cryptanalysis attacks.

a) Length Extension Attacks: Length extension also known as 'message extension' or 'padding' attack is well known weakness of MerkleDamgard construction. Given h = H(M), it is straightforward to compute M' and h', such thath' = H(M||M') (even for unknown M (but for known length |M|). The attack is based on using H(M) as an internal hash for computing H(M||M').Gauravram [16] classified it further in two types i.e. Type – A extension attack and Type- B extension attack. The categorization is based on whether the original message contains the length padding or not.Using the length extension attack it is possible, from only hash of a message and its length, to compute hash of longer messages that start with the initial message and include the padding required for the initial message to reach multiple of block size [56]. Length extension attack has been studied way back in 1992 by Tsudic [18] and even these days certain vulnerabilities based on this simple attack are being observed. Thai Duong and Juliano Rizzo [55] in 2009 showed a vulnerability in the Flickr (one of the best online photo management and sharing application in the world) signing process for making use of Flickr authentication API and this vulnerability allows an attacker to generate valid signatures without knowing the shared secret. By exploiting this vulnerability, an attacker can send valid arbitrary requests on behalf of any application using Flickr's API. When combined with other vulnerabilities and attacks, an attacker can gain access to accounts of users who have authorized any third party application.

b) JouxMulticollisionAttacks: Joux in [37] studied the generic multicollision attack on iterated hash functions. Joux showed that finding multicollisions, i.e. r-tuples of messages that all hash to the same value, is not much harder than finding ordinary collisions, i.e. pairs of messages, even for extremely large values of r. More precisely, the ratio of the complexities of the attacks is approximately equal to the logarithm of r i.e. constructing 2^d – collisions cost d times as much effort as building ordinary 2-collisions. In this attack, it is assumed that collision finding algorithm exists and the algorithm finds collision for the compression function $\boldsymbol{f}$with every call to it. To start with the attacker calls this collision finding algorithm to the compression function with the initial state H_0 and algorithm return two messages M1 and N1 such that $\boldsymbol{f}_{H0}(M1) \neq \boldsymbol{f}_{H0}(N1) = H_1$. Then the attacker calls this algorithm with state

H_1 and algorithm returns two message block M2 and N2 such that $f_{H1}(M2) \neq f_{H1}(N2) = H_2$. H_2 is then used as state and call to algorithm returns message blocks M3 and N3 such that $f_{H2}(M3) \neq f_{H2}(N3) = H_3$. Similarly successive calls to algorithm can be made. If only thee calls are made, then we have obtained $2^3 = 8$ different messages that maps to digest H_3. If we assume collision finding algorithm was based on brute force attack and every call takes time $2^{n/2}$ then it took O (3 x $2^{n/2}$) time to find 8-collisions. In general it can be demonstrated that this technique required O (d x $2^{n/2}$) time for finding 2^d-collisions instead ofa compression function f using a brute force collision finding algorithm. The brute force mechanism for finding 2^d-collisions would have required $\Omega^1(2^{n.k})$ where $k = (2^d - 1)/2^d$ and n is the message digest size.

c) Multi (2^{nd}) preimage Attacks based on Joux Technique: Thenotionmulti (2^{nd}) preimage represents multiple preimages as well as multiple 2^{nd}preimages. The technique presented by Joux [37] can be extended and multi (2^{nd}) pre-images can be found at a cost less than the brute force complexity of finding multiple (2^{nd}) preimages. Gauravram [16] exemplified this technique and presented that total cost of 2d – preimages or 2d – 2nd preimages for n-bit message digest is O (d x 2t/2 + 2t) instead of Ω (2d x 2n).

d) Generic 2nd preimage Attacks: In generic 2nd preimage attack on hash function of length n bits, the attacker tries to find a second pre-image X' for a target message X such that $X \neq X'$ and H (X) = H (X') with an effort less than 2^n . A number of techniques have been suggested to produce generic 2^{nd}pre-image attacks. Correcting Block attacks as defined in [3] can be used to generate generic 2^{nd} pre-image attacks. R D Dean [51] used Fixed Point attacks to generate generic 2^{nd}Preimages and Kelsey and Sheiner [57] made use of jouxmulticollisionsfor generating 2^{nd} pre-image attacks. In this subsection we provide brief overview of these attacks:

Correcting block attack: In thisopponent used a pre-existing (message, digest) pair and tries to change one or more message blocks such that the resulting digest remains same. To generatea second preimage X'for a target message X, the adversary chooses one of the input blocks X_i and replaces it with an alternative block X_i' so that $f(H_i, X_i') = f(H_i, X_i)$. If all other blocks of the alternative message X' are equal to the corresponding blocks of target message X, then the same hash result will be obtained and a second pre-image has been found. If the size of the internal state i.e. chaining variable is *c* bits and block size is *b* bits and $b > c$, then the number of block X_i' satisfying the property $f(H_i, X_i') = f(H_i, X_i)$ is approximately $2^b / 2^c$ i.e. 2^{b-c}. Challenge is such blocks are a small subset of all possible blocks, and for an ideal hash function about 2^c operations are needed to find one[3]. One round of MD5 has been detected for this attack. In MD5, the attacker takes a message block X (consisting of 16 words), fixes the 11 words of X, modifies one word and calculate the remaining 4 words to generate a message block X' which maps to the same digest. Correcting block attack is possible if the preimages for compression function can be obtained with the computation starting from pre-specified chaining values. Fixing the value of IV helps in thwarting the attack thus MD strengthening in case of MerkleDamgard construction avoids this attack from working on complete hash functions [16].

Fixed Point Attacks: In thisattack adversary looks for a fixed point in the compression function***f.*** A fixed point is chaining variable H_i such that $f(H_i, X_i) = H_i$. Few authors refer the pair (H_i, X_i) as fixed point. Whenever fixed point exists, the presence of message block X_i does not affect the message digest. To generate preimages of message X, one may insert arbitrary number of blocks with value X_i to the message X where chaining variable takes the value H_i. Fixed point attack can be avoided by inserting the message length at the end of message. As MD strengthening pad the message length at the end of original message MD strengthening thwarts fixed point attacks from affecting complete hash functions. However if fixed points are occur at more than one iteration of compression function, then attack may become practical. In such a case the attacker can insert message block X_i at stage *i* such that $f(H_i, X_i) = H_i$ andcan remove X_j from X at some later stage j, such that $f(H_j, X_j) = H_j$. Even in this case attack is only possible if the initial value is not fixed (the attacker chooses IV = H_i), or if fixed points can be found for a significant fraction of all chaining values.

R D Dean in [51] presents different techniques that make use of fixed points to produce attack on complete hash functions even in the presence of MerkleDamgard strengthening. One very simple technique proposed by R D Dean in [51] for MD4 and MD5 hash functions is to repeat the fixed point block 2^{55} times, which adds 2^{64} bits to the input. Since the message length in MD4 and MD5 is computed modulo 2^{64}, this effectively adds 0 to the length field, and the proper hash value comes out. Kelsey and Sheiner [57] have also improved the generic correcting block attack using the notion of expandable messages such that it bypasses the defense provided by MD strengthening. For details of expandable messages and various techniques to find generic 2^{nd}preimage attacks refer [51, 57].

e) **Herding Attacks:** Kesley and Kohno in [38] presented a new attack on hash functions based on

[1]Formally the symbol O is used for the expected running time and is asymptotically "at most" and Ω is used for the expected running time and is asymptotically "not less than"

MerkleDamgard structure, called the *Herding attack*. In Herding attack, an attacker who can find many collisions on the hash functions by brute force can first provide the hash of a message, and later "herd" any given starting point of a message to that hash value by the choice of an appropriate suffix. With this attack Kesley and Kohno identified an essential security property for hash functions called Chosen Target Forced Prefix (CFTP) preimage resistance. CFTP preimage resistance as defined by Kesley and Kohno in [38] is reproduced here:

In the first phase of the attack, adversary performs some pre-computation and then outputs an n-bit hash value H: H is his "Chosen Target". The challenger then selects some prefix ***P*** *(picks uniformly at random from large but finite set of strings) and supplies it to adversary;* ***P*** *is the "Forced Prefix." In the second phase of attack, adversary computes and outputs some String S. Adversary is said to compromise the CFTP preimage resistance if it takes less than* 2^n *evaluations of the hash function to find S such that* ***hash****(P||S) = H.*

Kesley and Kohno in [38] presented that for hash functions based on MerkleDamgard construction, CTFP preimage resistance can always be violated by repeated application of brute-force collision-finding attacks. An attack that violates this property effectively (less than 2^n computations) "herds" a given prefix to the desired hash value; and such an attack is called as Herding attack.As per Kesley and Kohno [38] the following steps are used for applying herding attack:

i. In the first phase of a herding attack, the attacker repeatedly applies a collision-finding against a hash function to build a diamond structure[2].
ii. In the second phase of the attack, attacker exhaustively searches for a string S' such that P || S' collides with one of the diamond structure's intermediate states.
iii. Having found such a string S', attacker can construct a sequence of message blocks Q from the diamond structure, and thus build a suffix S = S' || Q such that hash (P||S) = H.

Kesley and Kohno [38] also described the various contexts in which herding attack can be used. Nostradamus attack, Stealing credits for inventions, Tweaking a signed document and Random number fixing are examples of such contexts explained in [38]. At very general level, the methodology of these attacks as explained in [38] is as follow:

i. The attacker presents the victim with a hash H, along with a claim about the kind of information this represents. She promises to produce the message that yields the hash after the events predicted have occurred.
ii. The attacker waits for the events to unfold, just as the victim does.
iii. The attacker herds a description of the events as they did unfold into her hash output, and provides the resulting message to the victim, thus "proving" her prior knowledge.

f) Meet in the Middle Attack: This attack is a variation of birthday attack and is applicable to hash function that make use of compression function ***f*** invertible to the chaining variable H_i or the message block X_i .It allows theattacker to construct messages that corresponds to certain digest. To apply this attack adversary generates r1 samples for the first and r2 samples for the last part of the bogus message. Adversary then moves forward from initial value and goes backward from the hash value. The probability that two intermediate values are same is given by, $P \approx 1 - e^{-k}$, wherek = (r1*r2) / 2^n ; n = length of initial value or chaining value or message digest.If meeting point is found then then the concatenation of the message parts form a bogus message that results in the given hash value. [58]

6.2.2 Specific Attacks

The attacks that work on specific hash function or the algorithm of its compression function are called specificattacks. For example, collision attacks on the specific hash functions MD4 [30],MD5 [31,32], SHA-0 [33,34] and SHA-1 [33,35]. Attacks using differential cryptanalysis, linear cryptanalysis, rotational cryptanalysis &attack on the underlying encryption algorithms are type of specific cryptanalysis attacks. The most successful of these are the attacks based on differential cryptanalysis.

Differential Cryptanalysis: Differentialcryptanalysis was introduced by Biham and Shamir [59] and the technique was mainly devised to analyse block ciphers. In differential cryptanalysis the correlation between the difference in input and output is studied. If X and X' are two inputs then the difference between them is defined as $\Delta X = X \text{ op } X'$. If H and H' are two corresponding message digests then the difference between them is defined as $\Delta H = H \text{ op } H'$. The difference operation *op* canbe XOR operation or integer subtraction or any other operation. For differential cryptanalysis attack, the attacker searches for specific difference in inputs (ΔX)that result in specific difference in output (ΔH) with high probability. In case of hash function, the difference in output should be zero to result in collisions. Examples of specific attacks using differential cryptanalysis are [30, 31, 32,33, 34, 35, 60, 61].

Linear Cryptanalysis: Linear cryptanalysis was proposed by Matsui [62]. S. Bakhtiari*et. al.* in [58]

[2]Diamond structure is a data structure reminiscent to a binary tree. Diamond structure is a structure of messages constructed to produce large multicollisions. For details refer [38]

quoted that for Block ciphers like DES, better results have been obtained with Linear Cryptanalysis compared to Differential Cryptanalysis. Hash functions based on the Encryption algorithm can be susceptible to linear cryptanalysis, but till date not much successful attack on Hash functions using linear cryptanalysis has been reported.

Rotational Cryptanalysis: The termRotationalcryptanalysis was coined by in February 2010 by Dmitry Khovartovich and IvicaNikolic in [64]. The attack may also be classified as generic attack because as per [64] it may be applied on all the algorithms that are based on three operations modular addition, rotation and XOR (ARX for short). However we have placed it under the category of specific attacks as this attack has been demonstrated by Khovartovich and Nikolic against reduced round Threefish cipher – part of Skein hash function [66], a SHA3 competition [45] candidate only. Secondly as per our classification, the generic attacks are applicable to all the hash functions falling under a particular structure like MerkleDamgard, so it is better to consider rotational cryptanalysis as a specific attack. In October 2010, a followup attack that combines rotational cryptanalysis with the rebound attackwas presented by the same authors along with Christian Rechberger in [65].

Attacks on underlying Encryption Algorithm: Ifthe underlying compression function of hash function is implemented using the Encryption algorithm, then the weakness in encryption algorithm can be exploited to attack hash functions. Encryption function may have complementation property or weak keys or may have fixed points and the same may be used to attack complete hash function based on encryption algorithm. Miyaguchi*et. al.* in[63] analyzed the hash functions from the standpoint of the complementation property and weak keys of the block ciphers used in them and notified their weaknesses.

7.Type of Hash functions based on design of underlying Compression Function

From the discussion in section 4, it is evident that for processing arbitrary length of input the iterative structure of hash function (may be MerkleDamgard or any other) is desired and the crucial part of this iterative structure is Compression function and thus designer can view of all these approaches have been given in this section.

7.1 Hash Functions based on Block Cipher as Compression functions

One of the possible approaches that have been studied by the authors is to design a compression function from an existing cryptographic primitive like block ciphers. The advantage is that the existing implementations in hardware or software can be reused. Secondly some existing block ciphers like DES [67] or AES [7] have received a lot of scrutiny, and thus there is a lot of trust in their security properties [3]. At the same time a number of drawbacks of block cipher based hash functions have also been observed. One of the arguments is that the block ciphers do not possess the properties of randomizing functions. For example they are invertible. This lack of randomness may lead to weakness that may be exploited [85]. Secondly the differential cryptanalysis is easier against block operations in hash functions than against block operations used for encryption; because the key is known so several techniques can be applied. [68, 69] suggest the various techniques of using differential cryptanalysis for attacking hash functions based on clock ciphers. Thirdly it has been suggested that block cipher based on hash functions are significantly slower than hash functions based on compression function specially designed for hash functions. It is also felt that use of a block cipher for a purpose for which it was not designed may reveal some other weaknesses which may not be relevant in case of encryption. However with the adoption of AES, there has been renewed interest in developing a secure hash function based on strong bock cipher and exhibiting good performance [85]. Hash functions based on Block ciphers can be further classified as follows:

7.1.1 Single block length construction

These are the schemesin which size of hash code equals the block size of underlying block cipher. A number of proposals have been made and the basic concept to construct compression function ***f***fromblock cipher as described in [15] is as follow:

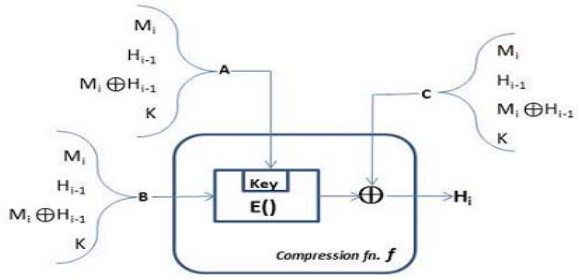

Fig. 3 Compression function based on block cipher

E is the clock cipher that takes two inputs A and B and produces an output that is XOR with variable C. Variable A, B and C can be either Mi, H_{i-1}, ($M_i \oplus H_{i-1}$) or a constant K (K may be assumed to be zero also). Message M is divided in to blocks and padding is done as illustrated in Section 4 and in each round one block M_i is processed in the compression function ***f***as per follow:

H_0 = Initial value

$H_i = E_A(B) \oplus C$

The three different variables A, B and C can take on one of four possible values, so there are 64 total schemes of this type. Preneel*et. al.*[72] studied them all and showed that 12 of them (as given in the *Table 1*) are secure.

Table 1: Secure Hash Functions as per [72] based on Block Cipher

Secure Schemesbased on Block cipher to generate Compression function	Other Common Name for the scheme as per the Literature
$H_i = E_{Hi-1}(M_i) \oplus M_i$	Matyas-Meyer-Oases Scheme [70]
$H_i = E_{Hi-1}(M_i \oplus H_{i-1}) \oplus M_i \oplus H_{i-1}$	--
$H_i = E_{Hi-1}(M_i) \oplus H_{i-1} \oplus M_i$	Miyaguchi – Preneel Scheme Independently proposed by Miyaguchi[71] and Preneel[73]
$H_i = E_{Hi-1}(M_i \oplus H_{i-1}) \oplus M_i$	--
$H_i = E_{Mi}(H_{i-1}) \oplus H_{i-1}$	Davies-Meyer Scheme [70, 74]
$H_i = E_{Mi}(M_i \oplus H_{i-1}) \oplus M_i \oplus H_{i-1}$	--
$H_i = E_{Mi}(H_{i-1}) \oplus M_i \oplus H_{i-1}$	--
$H_i = E_{Mi}(M_i \oplus H_{i-1}) \oplus H_{i-1}$	--
$H_i = E_{Mi \oplus Hi-1}(M_i) \oplus M_i$	--
$H_i = E_{Mi \oplus Hi-1}(H_{i-1}) \oplus H_{i-1}$	--
$H_i = E_{Mi \oplus Hi-1}(M_i) \oplus H_{i-1}$	--
$H_i = E_{Mi \oplus Hi-1}(H_{i-1}) \oplus M_i$	--

For formal proof of the security of these 12 schemes refer to [75] and for various other schemes proposed in literature that have been shown to be insecure refer [15, 72].

7.1.2 **Double block length construction**

A Hash function generating digest of 64 bits (or 128 bits) is insecure as brute force collision will require 2^{32}(or 2^{64}) operations only. Using the Single block length construction schemes as mentioned in previous sub-section, we will get a 64 bit digest with DES as underlying block or 128 bit digest with AES as underlying block cipher. To increase the digest size of hash function and to make it more secure double length block constructions is suggested. It is schemesin which size of hash code doubles the block size of underlying block cipher. This means, DES will result in a 128-bit hash function, and AES in a 256-bit hash function.The best known scheme in this class as suggested by Rompay [3] is MDC2 and MDC4 designed by B. Brachtl*et. al.* [76, 77].MDC-2 is sometime called as Meyer-Schilling scheme. The compression function of MDC2 makes uses of two parallel computations ofMatyas-Meyer-Oases scheme [70]. Explanation of MDC-2 as given in [3] is reproduced here using the terminology used in previous subsection.

Let C^L and C^R denote the left and right halves of b-bit block length of underlying block cipher. Then the compression function of MDC-2 can be described by $H_i \| H_i' = f(H_i \| H_i', M_i)$, which depends on the following computations:

$C_i = E_{Hi-1}(M_i) \oplus M_i$

$C_i' = E_{H',i-1}(M_i) \oplus M_i$

$H_i = C^L_i \| C'^R_i$

$H_i' = C'^L_i \| C^R_i$

The compression function of MDC-4 consists of two sequential executions of MDC-2 compression function. For the second MDC-2 compression, the keys are derived from the outputs (Chaining variables) of the first MDC-2 compression, and the plaintext inputs are the outputs (Chaining variables) from the opposite sides of the previous MDC-4 compression.

For details of few of the other double length construction schemes studied in literature like Quisquarter-Girault, LOKI Double Block, Parallel Davies Meyer, Tandem and Abreast Davies – Meyer schemes, refer [15, 78, 79, 80, 81]

Few of the famous hash functions based on block ciphers are listed below:

GOST Hash Function – This hash function comes from Russia, and is specified in the GOST R.34.11-94. It uses the GOST block encryption algorithm. For details refer [82]

AR Hash Function: AR Hash function was developed by Algorithmic Research, Ltd. and has been distributed by the ISO for information purposes only. Its basic structure is a variant of underlying block cipher (DES in the reference) in Cipher Block Chaining mode. For details refer [83]

Whirlpool Hash Function: Whirlpool is one of the only two hah functions endorsed by NESSIE (New European Scheme for Signatures, Integrity and Encryption). Unlike virtually all other proposals for a block-cipher based hash function, Whirlpool uses a block cipher that is specifically designed for use in the hash functions and that is unlikely ever to be used as a standalone encryption function. For details refer [84]

Skein Hash Function: Skein hash function is one out of five finalists in the NIST hash function competition [45] to design SHA-3 standard that will replace SHA-1 and SHA-2 [4, 5, 6, 8]. The algorithm is based on Threefishtweakable Block Cipher. For details refer [66]

Grøstl Hash Function: JustLikeSkein, Grøstl also is a SHA-3 final round candidate algorithm. Its compression functions is not exactly uses existing block cipher but Grøstl uses the same S-Boxes as AES. Its compression function f is based on a pair of permutation functions P and Q and these permutation functions are heavily based on AES [7] block cipher.

6.2 Hash functions based on Modular Arithmetic

Compression function can also be designed using modular arithmetic. This allows the reuse of existing implementations of modular arithmetic such as in asymmetric cryptosystems. The idea of cryptosystems based on modular arithmetic is to reduce the security of a system to the difficulty of solving the problems in number theory. Two important hard problems in number theory which can act as a base for generating cryptosystems are factorisation and Discrete logarithm. Rompay in [3] has referred to design of two variants of MASH hash functions based on modular arithmetic. The advantage of such hash functions is that the level of security can be easily enhanced by choosing Modulus M of appropriate length but hash functions based on modular arithmetic are very slow, even slower than block cipher based hash functions. Also many such constructions have been broken in the past.

6.3Dedicated Hash Functions

Dedicated hash functions are the one which are designed for the explicit purpose of hashing. Compression functions of dedicated Hash functions are not based on the existing cryptographic primitives like block ciphers and are not constrained to reuse existing components such as block ciphers or modular arithmetic. This means that they can be designed with optimised performance in mind. A number of such hash functions have been designed. Few of the famous dedicated hash functions and the status of attacks on these hash functions are as follows:

MDx Familyof hash functions: MD2, MD4 and MD5 are three hash functions from MDx family. Compared to other two, MD2 is slower and has not obtained much success. Dedicated hash functions which have received the most attention in practice are those based on the MD4 algorithm [3]. MD4 is a hash function proposed by R. Rivest in 1990 [9]. It was designed specifically towards software implementation on 32-bit platforms. Because of security concerns, Rivest in 1991 came up with a conservative version namedMD5 [10] to replace the earlier Hash MD4. MD5 became a milestone in the development of Hash. It was a widely-used well-known 128-bit iterated hash function, used in various applications including SSL/TLS, IPsec, and many other cryptographic protocols. It was also commonly-used in implementations of time stamping mechanisms, commitment schemes, and integrity-checking applications for online software and random-number generation. Type-2 (Semi free start collision) and Type-3 (Pseudo collision) attacks on MD5 were reported in [47, 50]. Strong collisions (Type-1 collisions) on MD4 and MD5 have been reported by Wang *et. al.* in [30, 31, 32] and these attacks make the further usage of these hash functions questionable.

SHA family of Hash Functions: Secure Hash Algorithm (SHA) developed by the National Institute of Standards and Technology (NIST) was also designed on the same principle as MD4 and was published as Federal Information Processing Standard (FIPS 180)in 1993 [4]. A revised version was issued as FIPS180-1 in 1995 and is generally referred to as SHA-1 [5]. When revised version of SHA-1 was published no details of the weaknesses found in SHA-0 (originally SHA) were provided [33]. SHA-1 produces a hash value of 160 bit. In 2002, NIST produced a revised version of the standard known as FIPS180-2 [6] and defined three new versions of SHA with digest lengths of 256, 384 and 512 and known as SHA-256, SHA-384, and SHA-512 respectively. So total SHA versions becomes four including SHA-1 (160 bit). In October 2008, FIPS 180-2 has been replaced by FIPS 180-3 [8] and in new standard SHA-224 has been added which is same as other SHA algorithm producing 224 bits of message difest. All these SHA versions are based on the same principle of MD4 and hash length has changed and certain other improvements have been carried from one version to next. Attacks on SHA-0 and SHA-1 have been reported in [33, 34, 35]. Till date no practical attack has been reported on SHA-2.

RIPEMD family of Hash Functions: RIPEMD family of hash functions consists of RIPE MD, RIPEMD-128, RIPEMD-160, RIPEMD-256, RIPEMD-320. RIPE MD, a 128 bit hash function, based on MD4 algorithm, was developed in the framework of the EU (European Union) project RIPE (RACE Integrity Primitives Evaluation) by Hans Dobbertin, AntoonBosselaers, Bart Preneel.. RIPEMD-160 [87] was an improved version of RIPE MD. The 128 bit version was intended only as a drop-in replacement for the original RIPEMD, which had been found to have questionable security. The 256 and 320 bit versions diminish chance of accidental collision, and don't have higher level of security compared to RIPEMD-160. A collision on RIPEMD was reported in [30] but that does not affect RIPEMD-160. Till date no practical attack has been observed on RIPEMD-160.

HAVAL Hash functions YuliangZeng, *et. al* invented HAVAL hash function in 1992 [86]. To certain extent it takes the motivation from MD4 hash function only. However HAVAL can produce hashes of different length i.e. 128, 160, 192,224 or 256 bits. In addition, HAVAL has a parameter that controls thenumber of passes a message block (of 1024 bits) is processed. A messageblock can be processed in 3, 4 or 5 passes. By combining output lengthwith pass, authors provided fifteen (15) choices for practical applicationswhere different levels of security are required. Algorithm was designed for 32-bit computers Experiments showed that HAVAL is 60%faster than MD5 when 3 passes are required, 15% faster than MD5 when4 passes are required, and as fast as MD5 when full 5 passes are required. Research has uncovered weaknesses which make further use of HAVAL (at least the variant with 128 bits and 3 passes) questionable. The strong collision attack on HAVAL was reported by Wang et. al. in [31].

All the above dedicated hash functions are somehow designed with motivation from MD4 algorithm only and thus are sometime collectively known as MDx type hash functions.

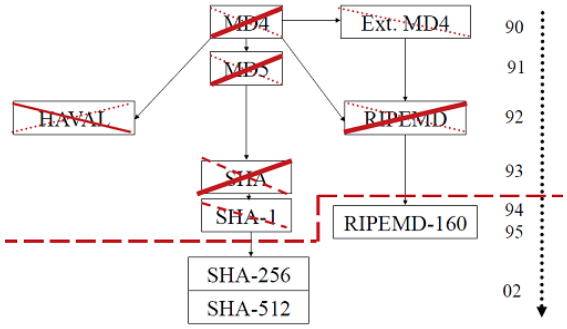

Fig. 4MDx-type hash function history [106]. Vertical line refer year when hash function was invented and functions Crossed with red lines have been attacked

Fewotherfamousdedicated hash functions reported in literature are SNEFRU [88], Tiger [89], JH [90], Keccak[46], Blake 91]. **Snefru**; designed by Ralph Merkle in 1990, like Khufu and Khafre block ciphers was an Egyptian Pharaoh. Snefru's initial design as well as modified design has been shown to to be insecure against differential cryptanalysis [93]. **Tiger** hash function was designed by Anderson and Biham in 1995mainly for 64-bit platforms. It is quite efficient on Software but because of its inherent use of large S-Boxes, implementation in hardware or small microcontrollers is difficult. Tiger hash function is frequently used in Merkle Hash tree form, where it is referred to Tiger Tree hash (TTH). TTH is used by many clients on Direct Connect and Gnutella file sharing networks. The last two in the list i.e. JH, Keccakand Blake are among the five finalists in the NIST hash function competition [45] to design SHA-3 standard. JH hash function makes use of S-boxes and is well suited for bit slicing. Keccak on the hand make use of sponge construction as detailed in Section 4. Blake does not fit exactly into the category of dedicated hash functions because it is based on ChaCha Stream Cipher.

6.4 Few Other approaches

There has been few hash functions that have not been based on existing cryptographic primitives like block ciphers or modular arithmetic but rather are based on some hard problems like knapsack problem, cellular automata or Discrete Fourier transformations.Hash function based on knapsack was proposed by Ivan Damgard in [26] but the same was shown to be broken in [94, 95]. Cellular automata based hash function was proposed in [96] by Wolram and in [97] by Daeman*et.al.*Claus Schorr[98, 99, 100] has proposed hash functions based on discrete Fourier transformations called FFT- hash. Three modifications of FFT-Hash have been proposed. First two modifications, FFT-Hash I and FFT – Hash II, was broken few weeks after the proposal [101, 102]. Third modification is quite slower. As a whole, all these approaches (based on knapsack or cellular automata or FFT) have not found much success and are not generally used these days.

7. Current Scenario: Progressing to SHA-3

The current Secure Hash Standard as developed by NIST (National institute of Standards and Technology) is FIPS 180-3 [8]. This standard suggests five hash functions SHA-1, SHA -224, SHA-256 SHA-384, and SHA-512. All these are dedicated hash functions as explained in Section 6 and to certain extent are based on MDx family. The practical attack on MDx family, followed by attack on SHA-0 and SHA-1 has been discussed in section 6.3. Majority of these attacks have been carried out in year 2004 and 2005 by a team of researchers from the Shandong University in Jinan China, led by Xiaoyun Wang. The same team also broke HAVAL-128 and RIPEMD. Looking at the variety of hash functions attacked by this team, it seemed likely that their approach may prove effective against all cryptographic hashes in the MD family, including all variants of SHA [103].

Burr from US National Institute of Standards and Technology [104] in his paper reviewed the scenarios of Cryptographic Hash Functions. Burr pointed out that with SHA-1 and SHA-2 in its cryptographic toolkit, NIST had hoped to be done with hash functions for a long time. Aside from a near break of MD5 by Dobbertin [26] in 1996, researchers made little progress in hash function analysis until mid-2004.

Since then, Wang, AntonineJoux, and Eli Biham have attacked nearly all the early hash functions, including SHA-1. Given that SHA-2 functions are in the same family as the earlier broken functions, these attacks shook cryptographers' long term confidence in nearly all hash functions designed to date. Cryptographers have learned much about hash functions and how to attack them in the past couple of years, and yet cryptanalysts generally agreed that practical attacks on the SHA-2 hash functions are unlikely in the next decade. However, attacks and research results could reduce their strength well below theoretical work levels (2^{112}, 2^{128}, 2^{192}, and 2^{256} operations for SHA-224, SHA-256, SHA-384, and SHA-512, respectively) [104].

Hoch and Shamir in year 2006 [105], studied the multi collisions on Iterated Concatenated Expanded (ICE) Hash Functions. Hoch and Shamir extended the idea presented by Joux [37]. Joux in 2004 [37]showed that in any iterated hash function it is relatively easy to find exponential sized multicollisions, and thus the concatenation of several hash functions does not increase their security. But Joux [31] Attack does not work on ICE i.e. when in addition to Iterated and Concatenated Hash Function technique message Expansion is also added i.e. each iterated function process message block more than once. Hoch *et al.*[105]considered the general case (ICE) and proved that even if we allow each iterated hash function to scan the input multiple times in an arbitrary expanded order, their concatenation is not stronger than a single function. Finally, authors extended their result to tree-based hash functions with arbitrary tree structures. Hoch *et al.* showed that a large class of natural hash functions (ICE and its generalization TCE) is vulnerable to a multicollision attack, and hoped that the techniques developed here will help in creating multicollision attacks against even more complicated types of hash functions. **Such a conclusion was perhaps hinting to probable attack on SHA 2 family of hash functions.**

Looking at the current scenarios, In **Nov 2007** NIST (National Institute of Standards and Technology) announced a **public competition** [45] to develop a new cryptographic hash algorithm to replace the older SHA-1 and SHA-2. The competition was NIST's response to advances in the cryptanalysis of hash algorithms. The winning algorithm will be named "SHA-3", and will augment the hash algorithms currently specified in the Federal Information Processing Standard (FIPS) 180-3, Secure Hash Standard [8]. As per NIST website "NIST is initiating an effort to develop one or more additional hash algorithms through a public competition, similar to the development process for the Advanced Encryption Standard (AES)." [45]

By October 31, 2008, NIST received **sixty-four** entries; and **selected fifty-one** candidate algorithms to advance to the **first round on December 10, 2008**, and fourteen advanced to the **second round on July 24, 2009**. A year was allocated for the public review of the fourteen second-round candidates. NIST received significant feedback from the cryptographic community. Based on the public feedback and internal reviews of the second-round candidates, NIST selected five SHA-3 finalists – **BLAKE [91], Grøstl [92], JH [90], Keccak [46],** and **Skein [66]** to advance to the third (and final) round of the competition **on December 9, 2010**, which ended the second round of the competition. A one-year public comment period is planned for the finalists. NIST also plans to host a final SHA-3 Candidate Conference in the spring of 2012 to discuss the public feedback on these candidates, and select the SHA-3 winner later in 2012 [45].

8. Conclusion

In this paper, we have shown how cryptographic hash functions slowly gained its importance in the field of cryptology. We have made all attempts to give a complete picture of cryptographic hashes, its design techniques and vulnerabilities. This paper would really help budding researchers who would take up research in this particular field.

References

[1]D. Kahn,TheCodebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet, Scribner, 1996.
[2] W. Diffie, and M. Hellman, "New Directions in Cryptography",IEEE Transactions on Information Theory, vol. 22, No. 6, 1976, pp. 644-654.
[3] B. V. Rompay, "Analysis and Design of Cryptographic Hash functions, MAC algorithms and Block Ciphers", Ph.D. thesis, Electrical Engineering Department, KatholiekeUniversiteit, Leuven, Belgium, 2004.
[4] FIPS 180, Secure Hash Standard (SHS), National Institute of Standardsand Technology, US Department of Commerce, WashingtonD. C., 1993.
[5] FIPS 180-1, Secure Hash Standard (SHS), National Institute of Standards and Technology, US Department of Commerce, WashingtonD. C.,1995.
[6] FIPS 180-2, Secure Hash Standard (SHS), National Institute of Standards and Technology, US Department of Commerce, WashingtonD. C.,2002.
[7] FIPS 197, Advanced Encryption Standard, National Institute of Standards and Technology, US Department of Commerce, WashingtonD. C.,2001.
[8] FIPS180-3, Secure Hash Standard (SHS), National Institute of Standards and Technology, US Department of Commerce, Washington D. C., 2008.
[9] R. Rivest, "The MD4 Message Digest Algorithm", IETF RFC 1320, 1992.
[10] R. Rivest, "The MD5 Message Digest Algorithm", IETF RFC 1321, 1992.
[11] R. C. Merkle, "Secrecy, Authentication and Public Key Systems", Ph.D. thesis, Department of Electrical Engineering, Stanford University, Stanford, USA, 1979.
[12] R.C. Merkle, "One Way Hash Functions and DES", in CRYPTO, 1989, pp.428-446.

[13] M. Naor, and M. Yung, "Universal One-Way Hash Functions and their Cryptographic Applications", in STOC, 1989, pp.33-43.
[14] P. Rogaway, and T. Shrimpton, “Cryptographic Hash-Function Basics: Definitions, implications and separations for preimage resistance, second preimage resistance, and collision resistance”, inFSE, 2004, pp.371-388.
[15] B. Schneier,Applied Cryptography, John Wiley & Sons, 1996.
[16] P. Gauravram, “Cryptographic Hash Functions: Cryptanalysis, design and Applications”, Ph.D. thesis, Faculty of Information Technology, Queensland University of Technology, Brisbane, Australia, 2003
[17] M. Bellare, R. Canetti, and H. Krawczyk, “Keying Hash Functions for Message Authentication”, in CRYPTO’96, 1996, pp.1-15.
[18] G. Tsudik, "Message Authentication with One-Way Hash Functions", inINFOCOM, 1992, pp. 2055-2059.
[19] R.L. Rivest, A. Shamir, and L.M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", inCommun. ACM, 1978, pp.120-126
[20] S. Singh, The Code Book: The Evolution of Secrecy fromMary, Queen of Scots to Quantum Cryptography, Doubleday Books, 1999.
[21] S. Haber, and W. Stornetta, “How to Time-stamp a Digital Document”, *Journal of Cryptology*, Vol. 3, No. 2, pp. 99-111, 1991.
[22] M. Bellare, R. Canetti, and H. Krawczyk, "Pseudorandom Functions Revisited: The Cascade Construction and Its Concrete Security", in FOCS, 1996, pp.514-523.
[23] I. Haitner, D. Harnik, and O. Reingold, "Efficient Pseudorandom Generators from Exponentially Hard One-Way Functions", in ICALP (2), 2006, pp.228-239.
[24] S.M. Matyas, A.V. Le, and D.G. Abraham, "A Key-Management Scheme Based on Control Vectors", IBM Systems Journal, No. 2, 1991, pp.175-191.
[25] H. Handschuh, and D. Naccache, “SHACAL (-Submissions to NESSIE -), in First Open NESSIE Workshop, 2000.
[26] I. Damgård, "A Design Principle for Hash Functions", inCRYPTO, 1989, pp.416-427.
[27] X. Lai and J. L. Massey, "Hash Function Based on Block Ciphers", in EUROCRYPT, 1992, pp.55-70.
[28]I. Mironov, “Hash Functions: Theory, Attacks, and Applications”, Microsoft Research, Silicon Valley Campus, 2005.
[29] M. Bellare, and T. Kohno, "Hash Function Balance and Its Impact on Birthday Attacks", in EUROCRYPT, 2004, pp.401-418.
[30] X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu, "Cryptanalysis of the Hash Functions MD4 and RIPEMD", inEUROCRYPT, 2005, pp.1-18.
[31] X.Wang, D. Feng, X. Lai, and H. Yu, “Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD", IACR Cryptology ePrint Archive, 2004, pp. 199.
[32] X. Wang, and H. Yu, "How to Break MD5 and Other Hash Functions", inEUROCRYPT, 2005, pp. 19-35.
[33] E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, and W. Jalby, "Collisions of SHA-0 and Reduced SHA-1", inEUROCRYPT, 2005, pp.36-57.
[34] X. Wang, H. Yu, and Y. L. Yin, "Efficient Collision Search Attacks on SHA-0", inCRYPTO, 2005, pp.1-16.
[35] X. Wang, Y. L. Yin, and H. Yu, "Finding Collisions in the Full SHA-1", inCRYPTO, 2005, pp.17-36.
[36] S. Lucks, “Design Principled for Iterated Hash Functions”, in IACR Cryptology ePrint Archive, 2004, pp. 253.
[37]A. Joux, "Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions", inCRYPTO, 2004, pp.306-316.
[38] J. Kelsey, and T. Kohno, “Herding Hash Functions and the Nostradamus Attack”, in EUROCRYPT, 2006, pp. 183–200.
[39] Y. Dodis, T. Ristenpart, and T. Shrimpton, "Salvaging Merkle-Damgård for Practical Applications", in EUROCRYPT, 2009, pp.371-388.
[40] M. Nandi, and S. Paul, "Speeding Up TheWidepipe: Secure and Fast Hashing", IACR Cryptology ePrint Archive, 2010, pp.193.
[41] E. Biham, and O. Dunkelman, "A Framework for Iterative Hash Functions - HAIFA", IACR Cryptology ePrint Archive, 2007, pp.278.
[42] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, “Sponge Functions”,in ECRYPT Hash Workshop, 2007.
[43] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "On the Indifferentiability of the Sponge Construction", in EUROCRYPT, 2008, pp.181-197
[44] G. Bertoni, J.Daemen, M. Peeters, and G. Van Assche, “Cryptographic Sponges”, [online] http://sponge.noekeon.org/.
[45] National Institute of Standard and Technology (NIST): Cryptographic Hash Algorithm Competition. [online] http://csrc.nist.gov/groups/ST/hash/sha-3/
[46]G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, “The Keccak Reference”, Submission to NIST (Round 3),2011.[online] http://csrc.nist.gov/groups/ST/ hash/sha-3/Round3/submissions_rnd3.html.
[47] B.den Boer, and A. Bosselaers, “Collisions for the compression function of MD5”, in EUROCRYPT, 1993, pp. 293-304.
[48] L. Knudsen. “Block Ciphers: Analysis, Design and Applications”, Ph.D.thesis, Aarhus University, Aarhus, Denmark, 1994
[49] O. Mikle, "Practical Attacks on Digital Signatures Using MD5 Message Digest", IACR Cryptology ePrint Archive, 2004, pp.356.
[50] H. Dobbertin, “Cryptanalysis of MD5 compress”, inEUROCRYPT, 1996
[51]R. D. Dean, “Formal Aspects of Mobile Code Security”, Ph.D. thesis, Department of Computer Science, Princeton University, Princeton, USA, 1999.
[52] E. Andreeva, G. Neven, B. Preneel, and T. Shrimpton, “Seven-Properties-Preserving Iterated Hashing: The RMC Construction”, ECRYPT document STVL4-KUL15-RMC-1.0, private communications, 2006.
[53] E. Andreeva, G. Neven, B. Preneel, and T. Shrimpton, "Seven-Property-Preserving Iterated Hashing: ROX", IACR Cryptology ePrint Archive, 2007, pp.176.
[54] M. Bellare, and T. Ristenpart, "Multi-Property-Preserving Hash Domain Extension and the EMD Transform", in ASIACRYPT, 2006, pp.299-314 .
[55] T. Duong, and J. Rizzo, “Flickr's API Signature Forgery Vulnerability”, 2009 [online] http://netifera.com/research/flickr_api_signature_forgery.pdf
[56] B. Kaliski, and M. Robshaw. “Message Authentication with MD5”. RSA Labs' CryptoBytes, Vol. 1, No. 1, Spring 1995.

[57] J. Kelsey, and B.Shneier, "Second preimages on n-bit Hash Functions for much less than 2^n Work",in EUROCRYPT, 2005, pp. 474-490.
[58] S. Bakhtiari, R. Safavi-Naini, and J Pieprzy. "Cryptographic Hash Functions: A Survey", Technical Report 95-09, Department of Computer Science, University of Wollongong, 1995
[59] E.Biham, and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", Journal of Cryptology, Vol. 4, No. 1, 1991, pp. 3-72.
[60] E.Biham, and A. Shamir, "Differential Cryptanalysis of FEAL and N-Hash", in EUROCRYPT, 1991, pp. 1-16.
[61] E. Biham, and A. Shamir, "Differential Cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer", in CRYPTO, 1991, pp. 156-171.
[62] M. Matsui, "Linear Cryptanalysis methods for DES Cipher", in EUROCRYPT, 1993, pp. 386-397.
[63] S. Miyaguchi, K. Ohta, and M. Iwata, "Confirmation that some Hash Functions are not Collisions Free" in EUROCRYPT, 1990, pp. 326 – 343.
[64] D. Khovratovich, and I. Nikolic, "Rotational Cryptanalysis of ARX", inFSE, 2010, pp.333-346.
[65] D. Khovratovich, I. Nikolic, and C. Rechberger, "Rotational Rebound Attacks on Reduced Skein", IACR Cryptology ePrint Archive, 2010, pp.538.
[66] B. Schneier, N. Ferguson, S. Lucks, D. Whiting, M. Bellare, T. Kohno, J. Walker, and J. Callas, "The Skein Hash Function Family", Submission to NIST (Round 3),2011. [online]http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html.
[67] FIPS 46-3, "Data Encryption Standard", National Institute of Standards and Technology, US Department of Commerce, WashingtonD. C., 1999.
[68] B. Preneel, "Differential Cryptanalysis of Hash functions based on Block Ciphers", ACM Conference on Computer and Communications Security, 1993, pp.183-188.
[69] V. Rijmen and B. Preneel, "Improved characteristics for Differential Cryptanalysis of hash functions based on Block Ciphers", in FSE, 1995, Vol. 1008, pp. 242-248.
[70] S. M. Matyas, C. H. Meyer, and J. Oseas, "Generating strong one-way functions with cryptographic algorithm", IBM Technical Disclosure Bulletin, Vol. 27, No. 10A, 1985, pp. 5658-5659.
[71] S. Miyaguchi. K. Ohtaand M. Iwata, "New 128-bit Hash functions", in 4th International Joint Workshop on Computer Communications, 1989, pp. 279 - 288.
[72] B. Preneel, R. Govaertsand J. Vandewalle, "Hash Functions Based on Block Ciphers: A Synthetic Approach", in CRYPTO, 1993, pp. 368- 378.
[73] B. Preneel and R. Govaerts, J. Vandewalle, "Cryptographically Secure Hash Functions: An Overview", ESAT Internal Report, K. U. Leuven, 1989.
[74] D. W. Davies and W. L. Price, "Digital Signature – An Update" in International Conference on Computer Communications, 1984, pp. 843-847.
[75] J. Black, P. Rogaway and T. Shrimpton,"Black-box analysis of the block-cipher-based hash function constructions from PGV."in CRYPTO, 2002, pp. 320-335.
[76] B.O. Brachtl, D. Coppersmith, M.M. Hyden, S.M. Matyas, C.H. Meyer, J. Oseas, S. Pilpel and M. Schilling, "Data Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function",1990, U.S. Patent Number 4,908,861.
[77] C. H. Meyer and M. Schilling, "Secure program load with manipulation detection code." in Securicon, 1988 pp. 111-130.
[78] J. J. Quisquarter and M. Girault, "2n-bit Hash functions using n-bit Symmetric block Cipher Algorithms", in EUROCRYPT , 1990, pp 102-109.
[79] W. Hohl, X. Lai, T. Meier and C. Waldvogel, "Security of Iterated Hash Functions based on Block Ciphers", in CRYPTO, 1994, pp. 379 – 390.
[80] X. Lai, "On the Design and Security of Block Ciphers," ETH Series in Information Processing, vol.1, Konstanz: Hartung-GeorreVerlag, 1992.
[81] X. Lai and J. Massey, " Hash functions based on Block Ciphers", in EUROCRYPT , 1992, pp. 55-70.
[82] GOST R 34.11- 94, Gosudarstvennyi Standard of Russian Federation, "Information technology. Cryptographic Data Security Hashing function. "Government Committee of the Russia for Standards, 1994 RFC 5831
[83] ISO. ISO N179 AR Fingerprint Function. Working document, ISOIEC/JTC1/SC27 WG2, International Organization for Standardization, 1992.
[84] P. S. L. M. Barreto and V. Rijmen, "The Whirlpool hashing function". Primitive submitted to NESSIE, September 2000, revised on May 2003.
[85] W. Stallings, Cryptography and Network Security, Pearson Prentice Hall,USA, 2009.
[86] Y. Zheng, J. Pieprzyk and J. Seberry, "HAVAL — A One-Way Hashing Algorithm with Variable Length of Output", in AUSCRYPT, 1993, pp. 83-104.
[87] H. Dobbertin, A. Bosselaersand B. Preneel, "RIPEMD-160: A Strengthened Version of RIPEMD",in Fast Software Encryption, 1996, pp. 71-82.
[88] R. C. Merkle, "A fast software one-way hash function", Journal of Cryptology, Vol. 3, No. 1, 1990, pp. 43-58.
[89] R. Anderson and E. Biham, "Tiger — A Fast New Hash Function" , in Fast Software Encryption, 1996, pp. 89-97.
[90] H. Wu: "The Hash Function JH", Submission to NIST (Round 3), 2011. [online] http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html
[91] J. P. Aumasson, L. Henzen, and W. Meier, "SHA-3 proposal BLAKE," Submission to NIST (Round 3), 2011. [online] http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html
[92]P. Gauravaram, L. R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen, "Grøstl- A SHA-3 Candidate", Submission to NIST (Round 3), 2011. [online] http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html
[93] E. Biham, "New techniques for Cryptanalysis of hash functions and improved attacks on Snefru" in FSE, 2008, pp. 444-461.
[94] P. Camion and J. Patarin, " The knapsack hash function proposed at Crypto'89 can be broken", in EUROCRYPT, 1991, pp. 39-53.
[95] A. Joux and L. Granboulan, " A Practical Attack against Knapsack based hash functions", in EUROCRYPT ,1995, pp. 58-66.
[96] S. Wolfram, "Cryptology with Cellular Automata", in CRYPTO, 1986, pp. 429-432.
[97] J. Daeman, R. Govaerts and J. Vandewalle, " A framework for the design of One-way hash functions including cryptanalysis of Damgard's One way function based on Cellular Automata", in ASIACRYPT, 1993, pp. 82-96.
[98] C. P. Schnorr, "An efficient Cryptographic Hash Functions" in CRYPTO, 1991.
[99] C. P. Schnorr, " FFT –Hash II, Efficient Cryptographic Hasing", in EUROCRYPT , 1993 pp. 45-54.

[100] C. P. Schnorr and S. Vaudenay, " Parallel FFT – Hashing", in Fast Software Encryption, 1994, pp. 149 – 156.
[101] J. Daeman, R. Govaerts and J. Vandewalle, " Collisions for Schnorr's hash function FFT-Hash" in CRYPTO, 1991 pp. 477-480.
[102] S. Vaudenay, " FFT-Hash II is not yet Collision Free", in CRYPTO, 1992, pp. 587 – 593.
[103] J. Black, M. Cochran and T. Highland, "A Study of the MD5 Attacks: Insights and Improvements", in FSE, 2006,Vol. 4047, pp. 262-277.
[104] W. E. Burr, "Cryptographic Hash Standards: Where Do We Go from Here", IEEE Security & Privacy, Vol. 4, No. 2, 2006, pp. 88-91.
[105] J. J. Hoch and A. Shamir, "Breaking the ICE - Finding Multicollisions in Iterated Concatenated and Expanded (ICE) Hash Functions", in FSE, 2006, Vol. 4047, pp.179-194.

Authors

Rajeev Sobti is heading School of Computer Science, Lovely Professional University, India. He has over 13 years of experience in industry, teaching and research. His research interest includes Cryptography and Computer System Architecture. He is also member, Consultant Board and Manuscript reviewer for Books on Discrete Mathematics, Operating System from Pearson Education (Singapore) PTE LTD.

Prof.G.Geetha is heading School of Computer Science and Applications, Lovely Professional University, India. She has nearly two decades of experience in industry, teaching and research. Her research interest includes Cryptography, Information security and Image Processing. She has published more than 50 research papers in refereed Journals and Conferences. She serves as Editorial Board member and reviewer in various Journals and Conferences. She is presently the President of Advanced Computing Research Society. She is an active member of various professional organizations like ISCA, ISTE, CRSI etc.