Attacks on and Advances in Secure Hash Algorithms

Neha Kishore, *Member IAENG*, and Bhanu Kapoor

Abstract— **In today's information-based society, encryption along with the techniques for authentication and integrity are key to the security of information. Cryptographic hashing algorithms, such as the Secure Hashing Algorithms (SHA), are an integral part of the solution to the information security problem. This paper presents the state of art hashing algorithms including the security challenges for these hashing algorithms. It also covers the latest research on parallel implementations of these cryptographic algorithms. We present an analysis of serial and parallel implementations of these algorithms, both in hardware and in software, including an analysis of the performance and the level of protection offered against attacks on the algorithms.**

Index Terms—**Cryptographic Hash Function, Parallel Algorithm, Cryptography, Security, Secure Hashing Algorithm**

I. Introduction

The number of the computing devices have grown exponentially over the years. Now information mostly exists in the digital form, whether it belongs to a government organization, to a private sector enterprise, or to an individual. There is a huge amount of information on the Internet and it travels through various types of networks from user to user.

Before the arrival of data processing tools, the security of vital information of an organization was primarily provided by physical means like lockers, signatures, and safe boxes. But now, with the development of many digital data processing tools, it has become necessary to have automated tools in order to protect the information not only when it is stored on various types of computing devices but also when it's being communicated over the networks. The security of digital information need to be maintained both when it's static and when it's dynamic. For static security, the information stored on the computing devices must be properly encrypted and its access must be controlled. For dynamic security, appropriate network security measures must be in place to protect the data during its transmission.

The security of digital information is not just a single service but is a collection of various services. These services include: authentication, access control, data confidentiality, non-repudiation, and data integrity[1]. A system has to ensure one or more of these depending upon the security requirements for a particular system. For example, in addition to the encryption of the data, we may also need authentication and data integrity checks for most of the situations in the dynamic context [2]. The development of cryptographic hashing algorithms, to ensure authentication and data integrity services as part of ensuring information security, has been an active area of research.

For ensuring data integrity, SHA-1[1] and MD5[1] are the most common hashing algorithms being used in various types of applications. Some of these applications include digital signature, password protection, digital forensics, SSL protocol, micropayment, text and content based image retrieval[4], and image encryption[5]. There have been several advances in these algorithms over the years to speed up the overall hashing process and to secure these algorithms from the attacks. This paper presents an overview of cryptographic hashing algorithms, including both software and hardware-based implementations, to achieve the goals of improved security and performance gains.

The organization of the paper is as follows: In Section II, we discuss the background of cryptographic hash functions followed by the level of security offered by these algorithms in Section III. Section IV describes the Secure Hash Algorithm (SHA-1) along its successors and variants as well as the latest on various successful attacks on these algorithms. Section V discusses some recent advances in the parallelization of hashing algorithms with goals of improving their performance. Section VI provides the key conclusions from this paper.

II. Background

The security of digital data means protecting data, such as a database, from destructive forces and from the unwanted actions of unauthorized users. Now automated information systems are being used to replace the traditional security measures. These systems use signatures, keys, dates, and code words to secure digital data. These systems provide the security measures (preserving the integrity, availability, and confidentiality) in digital form to all of the information system resources. The main objective is to provide the security to data processing systems to prevent attacks on the digital assets. The key security services in a digital information systems include:

- *Authentication* – to assure that communicating entity is the one who has claimed to be.

Manuscript received October 08, 2015; revised January 29, 2016 . This work is part of the research work done on Cryptographic Hash Functions for the fulfillment of the degree of Doctorate.

N. Kishore is with Chitkara University, Barotiwala, Himachal Pradesh, 174103, India (phone: +91-9592405665, 01795-661026; e-mail: nehakishore.garg@gmail.com).

B. Kapoor is Consultant/Owner, Mimasic, Dallas, TX USA, and teaches at Walden University. He is also an adjunct faculty in the Department of Computer Science and Engineering at Chitkara University, Barotiwala, Himachal Pradesh, 174103, India (e-mail: bkapoor@mimasic.com).

- *Access Control* – to prevent the unauthorized use of the resources.
- *Data Confidentiality* – to protect information from unauthorized disclosures.
- *Data Integrity* – to assure that message received is same as one sent by an authorized body.
- *Non-Repudiation* – to protect against denial by any of the parties involved in a communication.
- *Availability* – to assure resource accessibility and usability to provide information services.

These security services can be achieved by using one or more of the security mechanisms. Security mechanisms are features designed to identify, avert, or recuperate from a security attack. Cryptographic techniques are elements that underlie many of these security mechanisms. Cryptography is an art of disguising a message and hiding the information such that it is not readable by any unauthorized party. There are three broad areas [6] of study in cryptography: symmetric cryptosystems, asymmetric cryptosystems, and keyless cryptosystems. Symmetric cryptosystems use single private key to convert a plain text into cipher text (i.e. in disguised form) whereas asymmetric cryptosystems use set of two keys, public and private, for the encryption and the decryption processes, respectively. Both of these primitives provide secrecy as a service. Cryptographic Hash Functions (CHFs) [1] act as symmetric primitives when using keyed hash functions and as keyless primitives when using keyless hash functions. The problem of message authentication, message integrity, and confirming the identity of the sender in ecommerce applications is perhaps equally or more important than the encryption of the data. Message authentication via CHFs ensure the reliability of a message, validate the identity of the initiator, and ensure non-repudiation of the originator.

CHF is a function that takes a block of data or a long message as input and returns a fixed-size hash or a unique code, known as the Message Digest. More precisely, a hash function H maps bit-strings of arbitrary length from a domain D to strings of fixed length (n) in range R with $H: D \rightarrow R$ and $|D| > |R|$. It is considered, relatively easy to compute a hash value *h* for a given message M through the use of CHF. Any accidental or intentional change to the data leads to a complete change in the hash value. This is useful in ensuring data integrity as a change in data. Some of the common security applications of CHFs include digital signatures, message authentication codes (MACs) for use in the SSL protocol, finger-printing of any type of data, forensic applications, and checksums to detect any accidental data corruption.

A cryptographic hash function must be able to withstand all known types of cryptanalytic attack. At a minimum, it must have the following properties of a secure cryptographic function:

The CHF H can be applied to a block of message of an arbitrary length.

1. It produces an output h of fixed length.
2. It is relatively easy to compute h for a given M.
3. *Pre-image Resistance:* Given h, it is infeasible to generate M such that H(M)=h.
4. *Second Pre-image Resistance:* Given M, it is hard to find another message, M″, such that H(M)=H(M″).
5. *Collision Resistance:* Given M≠M″, it is infeasible to find H(M)=H(M″).
6. *Pseudo-randomness:* The value h must be deterministic and it must random in relation to its input.

In recent years, there has been great development in CHFs that satisfy these properties. A CHF that satisfies the above stated first five properties are referred to as a weak hash function. One of the simplest hash function [1, 57] uses bit-by-bit exclusive-OR (XOR) of the data for every block of the message and combines it with a one-bit circular shift or rotation of the resulting hash code for each block. Although this procedure gives a good measure of data integrity, ideally it doesn't provide enough security in terms of collision protection when the encrypted hash value on a simple plaintext message. The most widely used CHF have been the Secure Hash Algorithm (SHA) and the Message Digest (MD) family. We mainly look into the SHA family in this paper. The MD family algorithms also have a structure that is similar to the SHA family algorithms.

The next section of the paper covers the security of the cryptographic hash functions in terms of some of the desirable properties mentioned before.

III. Security of Cryptographic Hash Functions

The first three properties of CHFs defined in Section II represent some of the basic requirements for practical application of a CHF in various applications. The CHFs are said to be secure if they satisfy at least three of the basic properties: *pre-image resistance, collision resistance,* and *second pre-image resistance*. In addition, there exist many more application specific security properties that a CHF should also preserve for a given application.

Next, the basic security properties of CHFs and the nature of attacks against these properties are covered. When it is said that an intended attack has succeeded in breaking a CHF, it doesn't necessarily imply that it has been practically broken as well. While M\many of the attacks have been theoretically proven, it is still practically infeasible to crack them. These types of attacks mainly prove the structural or constructional weakness of the CHFs that can be exploited to make an attack practically feasible later. The MD5 algorithm has been subjected to such testing attacks, first theoretically broken, and later practically as well on the basis of theoretical work [7, 8].

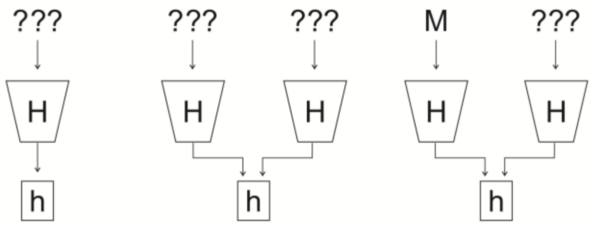

Fig. 1. (a) Pre Image Collision Resistance, (b) Weak Collision Resistance, (c) Strong Collision Resistance

A. Pre-Image Resistant (PIR)

CHFs are considered to be computationally non-invertible which means, if a hash code H(M) is generated for a message M, it is considered to be computationally infeasible for an adversary (A) to retrieve the original message (M) back (illustrated in Fig. 1(a)). The pre-image resistant property assures the non-reproducibility of the original message. The advantage of an adversary (A) for finding a collision in a CHF can be defined mathematically as:

$$Adv_H^{pir[m]}(A) = \Pr[M \xleftarrow{\$} \{0,1\} : X \leftarrow H(M);$$
$$M' \xleftarrow{\$} A(X) : H(M') = X;] \qquad (1)$$

Unlike the encryption, there should be no "dehash" function. A good pre-image resistant function should be "hard" to invert. Brute force attacks are considered to be best attacks for a CHF that is pre-image resistant. In the brute force attack method, random values of a message M″ are taken and tried until a collision is found. The level of effort required to find a collision is proportional to 2^n, for an n-bit hash value. On an average an attacker needs to try 2^{n-1} different values of M″ in order to generate the same hash code h. The attack is not dependent on any specific algorithm but only on the bit length of the hash value. So, the complexity of finding collisions increases with the increase in the hash code length and greater the hash code length, the more secure is the hash function.

B. Collision Resistance (CR)

For a CHF to be weak collision resistant or second pre-image resistant, it should be computationally infeasible for an adversary (A) to find two different messages M and M″ which can generate same hash values from that CHF. That is, to find M, M″; such that H(M) = H(M″) but M ≠ M″ (illustrated in Figure 1(b)). This can be expressed mathematically as:

$$Adv_H^{cr[m]}(A) = \Pr[M \xleftarrow{\$} \{0,1\}^n ; M' \xleftarrow{\$} A(M) :$$
$$M \neq M' \wedge H(M) = H(M');] \qquad (2)$$

What this says is that given complete control over picking any messages you want, it should be "hard" to find two of them such that have the same hash value. This property thwarts the falsification of the message in case an encrypted hash code is used. The level of effort required to find a collision is proportional to $2^{n/2}$, for an n-bit hash value. Rogaway has stated in [9] that for any keyless hash function, there will always be a collision although it could be difficult for humans to detect but collision will still be there due to the pigeonhole principle. He has also named this illusion of humans as foundation-of-hashing dilemma.

If the SCR property (sixth in Section II), is also satisfied, then the CHF is referred to as a strong CHF.

C. Second Pre-image Resistance (SCR)

For a CHF to be strong collision resistant, it should be computationally infeasible for an adversary (A) with given CHF H and message M to find another message M″ where M≠M″ and H(M) = H(M″) (illustrated in Figure 1(c)).

$$Adv_H^{scr[m]}(A) = \Pr[(M, M') \xleftarrow{\$} A :$$
$$M \neq M' \wedge H(M) = H(M');] \qquad (3)$$

This attack involves much less effort than a pre-image or second pre-image attack. Cryptanalysis attack can be used as an attack against this property. Cryptanalysis is another type of attack which is used to check the strength of the algorithm and is based on the weaknesses in a particular cryptographic algorithm, in contrast to the brute force attack.

The other desirable properties that a CHF should also preserve includes *semi-free-start collision resistance, near-collision resistance, pseudo collision resistance, chosen target Forced prefix pre-image resistance, partial pre-image resistance, non-correlation* etc. But satisfaction of these properties depends upon the type of the application and the level of security required in it.

In the next section of the paper, we discuss the design and of the widely used family of CHF called the SHA algorithms.

IV. Secure hash Algorithm

The National Institute of Standards and Technology (NIST)[10] publishes a family of cryptographic hash functions, the Secure Hash Algorithm as a U.S. Federal Information Processing Standard (FIPS)[11]. This family includes a number of cryptographic hash functions being advanced over the years to meet stronger security requirements.

Most of these hash functions are composed of two components: a compression function and a domain extender.

- *Compression function:* It's a function H which associates the fixed-length input to a fixed-length output i.e. $H : \{0,1\}^{b+n} \rightarrow \{0,1\}^n$ where *H* maps *b+n* bits to *n* bits.
- *Domain extender:* It's a generic process that uses the compression function *H* with fixed-length input iteratively and transforms into a hash function which can handle arbitrary length of input.

Generally the domain extender used is the Merkle-Damg˚ard construction[12] which works as follows:

Step 1: Compression function $H : \{0,1\}^{b+n} \rightarrow \{0,1\}^n$; n-bit constant initialization vector (IV) as shown in Fig. 2.

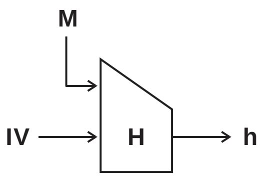

Fig. 2. Compression Function of Merkle Damg˚ard Construction

Step 2: Iterate the compression function until all the blocks of n bits have been hashed as shown in Fig. 3.

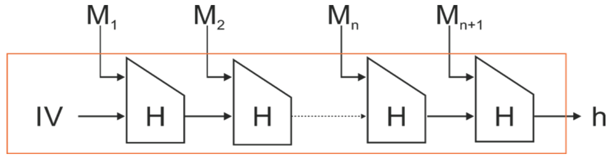

Fig. 3. Domain Extender of Merkle Damg˚ard Construction

Next sections discuss various available SHAs including variants that are being looked into for applications in near future: SHA-0 in sub-section *A*, SHA-1 in *B*, SHA-2: SHA-256, SHA-378, SHA-512 in *C*, and SHA-3 in *D* followed by a discussion on the experimented attacks on these algorithms. Apart from these, researchers have also proposed many more hash functions based on the SHA family [13].

A. SHA-0

In 1993, NIST developed the Secure Hash Algorithm (SHA) and published it as the FIPS 180 publication of the NIST. The design of the algorithm is closely based on the MD4 hash function. It was initially created to be used with the Digital Signature Standard (DSS) but due an undisclosed flaw, it was withdrawn after two years. It was replaced by a revised version of the algorithm called SHA-0 and its name was later changed to SHA-1. Since there was just one more instruction more the SHA-1 compared to the SHA-0, there were no reasons to keep the initial version and SHA-1 came into existence. SHA-1 is discussed in the next section in more detail.

Attacks on SHA-0:

An attack on SHA-0 was firstly presented at CRYPTO 98 by Florent Chabaud and Antoine Joux[14] and they proved that the collisions of the hash values can be found with a complexity of 2^{61}, much fewer than 2^{80} for a perfect CHF of the same size.

In the year 2004, Biham and Chen[15] found two different messages that hash to closely the same value with 142 out of the 160 bits are equal, a near-collision for SHA-0.

Then, on 12 August 2004, Joux, Carribault, Lemuet, and Jalby[16] announced a collision for the full SHA-0 algorithm. The chances of the collision was now 1 in 2^{51} and about 80,000 CPU hours were used on a supercomputer with 256 Itanium 2 processors to demonstrate the attack. This was a generalization of the Chabaud and Joux attack discussed earlier.

Later an attack with a chance of 1 in 2^{40}, much better than the previous one, was announced by Wang, Feng, Lai, and Yu[17] at the CRYPTO 2004 Rump Sessions. The attacks worked on MD5, SHA-0, and other similar hash functions.

Again in February 2005, an attack with a chance of 1 in 2^{39} complexity was found by X. Wang, Y. Lisa Yin, and H. Yu in SHA-0[18].

TABLE I
ATTACKS MADE ON SHA-0

Attacker/ Publisher	Outcome	Year	Paper
Florent Chabaud and Antoine Joux	Collisions with complexity 2^{61}	1998	Differential collisions in SHA-0
Biham and Chen	Full Collisions of 65 round and collision with 142 bits equal	2004	Near-collisions of SHA-0
Joux, Carribault, Lemuet, and Jalby	Collisions with complexity 2^{51}	2004	Collision in SHA-0
Wang, Feng, Lai, and Yu	Collisions with complexity 2^{40}	2004	Collisions for hash functions MD4
X. Wang, Y. Lisa Yin, and H. Yu	Collisions with complexity 2^{39}	2005	Efficient collision search attacks on SHA0

Table I summarizes various attacks on SHA-0 algorithm by different researchers and it includes the years when the attacks were published along with the respective complexities as outcome.

After several successful collision attacks with progressively reduced complexity, SHA-0 and MD4 were considered to be insecure for further use in authentication purposes.

B. SHA-1

SHA-1[RFC3174] was designed by the National Security Agency (NSA) and published by the NIST as the FIPS 180-1 publication in 1995. It is also standardized as a dedicated CHF in the ISO/IEC 10118 standard. The design of the algorithm was based on the MD4 and the MD5 algorithms. The compression function of SHA-1 is based on block cipher and its domain extender in the Merkle-Damg˚ard [12] construction. The maximum message or file size for the algorithm is 2^{69}-1 bits and it produces a message digest of 160 bits [19].

The compression function takes a block of size 512 bits as an input which is then further subdivided into sixteen 32-bit blocks. In SHA-1, there are 4 rounds for the updates of the internal state each containing 20 steps. A single round of the compression function in the algorithm is shown in Fig. 4, which transforms the five 32-bit variables to form the final hash value.

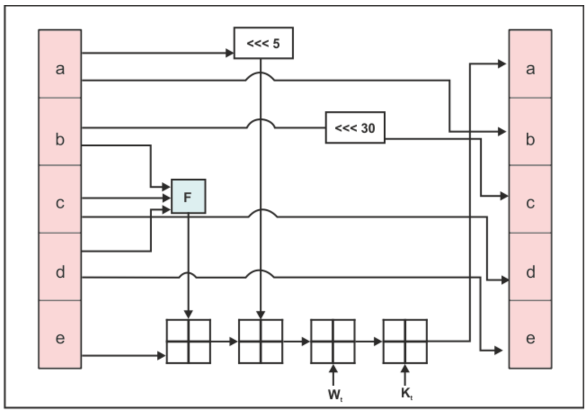

Fig. 4. One Iteration of Compression Function of SHA-1

SHA-1 algorithm has an Avalanche Effect i.e., even when only one bit of the message is changed, more than half of the generated hash value changes.

Attacks on SHA-1:

After the attacks were found on SHA-0, experts suggested that the usage of SHA-1 in forthcoming cryptosystems should be given a second thought. The results at CRYPTO 2004 insisted NIST to announce the use of SHA-2 variants and phase out the use of SHA-1 by 2010.

After the announcement of SHA-1, an attack was published [20] on a reduced version of SHA-1 that had only 53 out of 80 rounds. This attack could ultimately find collisions with a computational effort of less than 2^{80} operations.

Wang *et al.* [21] announced another attack on the full version of SHA-1 in their February 2005 publication. In this attack, collisions can be found requiring less than 2^{69} operations as compared to a brute-force search that would would require 2^{80} operations to find a collision.

On behalf of X. Wang, A. Yao, and F. Yao at the CRYPTO 2005 Rump sessions, an announcement was made on lowering of the complexity that is required to find a collision in SHA-1 to 2^{63}. Later in December 2007, this announcement and its results were explained in detail by Martin Cochran [22].

A significant theoretical attack was presented by Christophe De Cannière and Christian Rechberger [23] at ASIACRYPT 2006. They proposed a two-block collision for 64 rounds, found by using unoptimized methods with 2^{35} compression function evaluations. Grechnikov further extended their attack to 73 rounds (out of 80) in 2010 in response to challenge to catch a collision in the full 80 rounds of the hash function.

In 2008, Stéphane Manuel [24] announced an attack of hash collisions with a projected theoretical complexity of 2^{51} to 2^{57} operations but later, when he found out that the local collision paths were not autonomous, he just withdrew the claims.

In paper [25], authors claimed a hash collision attack with complexity of 2^{52} at the Rump session of Eurocrypt 2009. But later authors discovered about the incorrect estimate and withdrew that paper as well.

In November 2010, Marc Stevens also claimed a completely working near-collision attack against full SHA-1 with a projected complexity equivalent to $2^{57.5}$ SHA-1 compressions. He developed a project HashClash by making use of CPU power from cloud servers to break a single hash value of the SHA-1 algorithm.

Table II lists various attacks on the SHA-1 algorithm as reported by different researchers including the years of publication along with the respective complexities as outcome.

TABLE II
ATTACKS MADE ON SHA-1

Attacker/ Publisher	Outcome	Year	Paper
Rijmen and Oswald	Collisions possible for 53 rounds instead of 80	2005	Update on SHA-1
Xiaoyun Wang, Yiqun Lisa Yin and H. Yu	Collisions with complexity < 2^{69} operations	2005	Finding collisions in the full SHA-1
Wang et al., Marti Cochran	Collisions with complexity 2^{63}	2005	Notes on the Wang et al. 2^{63} SHA-1 Differential Path.
Christophe De Cannière and Christian Rechberger	two-block collision for 64-round	2006	Finding SHA-1 Characteristics: General Results and Applications
Manuel, Stéphane	Already known attack	2008 2011	Classification and generation of disturbance vectors for collision attacks against SHA-1
Cameron McDonald, Philip Hawkes and Josef Pieprzyk	Paper withdrawn, estimate was incorrect	2009	Differential Path for SHA-1 with complexity O(2^{52})
Marc Stevens	Complexity equivalent to $2^{57.5}$	2010	

These attacks have rushed the transition to newer and stronger versions of SHA. But the SHA-1 algorithm is still used in a wide variety of applications which include Digital Signatures, TLS/SSL, SSH, and PGP.

C. SHA-2

Three new revised versions of SHA were added into the SHA family by NIST in August 2002 as the FIPS 180-2 publication. These are known as SHA-256, SHA-384, and SHA-512 with the respective hash value lengths of 256, 384, and 512 bits [26]. Later in 2008, the FIP PUB 180-3 publication was issued as a revised document which added SHA-224[RFC3874] into the family as well. These algorithms together are recognized as SHA-2.

The new versions bear the same underlying resemblance of structure, modular arithmetic, and logical binary operations as that of SHA-1 without sharing its weaknesses. The algorithms SHA-256 and SHA-512 caries same basic design with the difference that SHA-256 operates on eight 32-bit words, while SHA-512 operates on eight 64-bit words as designed especially for the 64-bit processors. SHA-384 is a slight modification to SHA-512 and uses a composite of different initial values of the chaining variable and its hash code length is 384 bits. SHA-224 is a trimmed version of SHA-256 algorithm with a different initial value.

These hash functions are targeted to provide higher level of security. Apart from the hash size and the initial values, the four new functions differs from SHA-1 in the process of deriving sub-blocks from a block of a message. Fig. 5 shows single round of compression function of the SHA-2 family.

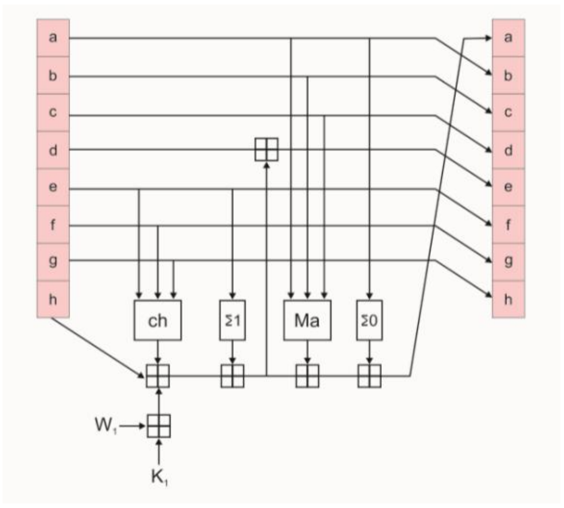

Fig. 5. One Iteration of Compression function of SHA-2 family

Attacks on SHA-2

SHA-2 family has also faced cryptographic attacks partly resulting from the SHA-3 competition which provoked the researchers and attackers to work on the analysis of SHA-2 variants. But as of now, only collision attacks found are with practical complexity and none of the attacks yet use the complete set of rounds as provided in SHA-2. Some of these attacks are listed in Table III along with the year of attack, method, and the complexity for collision as outcome.

At the Cryptology-INDOCRYPT 2008 meeting, Sanadhya, Somitra Kumar, and Palash Sarkar presented a deterministic collision in 24/64 rounds with $2^{28.5}$ complexity of SHA-256 and in 24/80 rounds with $2^{32.5}$ complexity of SHA-512[27].

TABLE III
ATTACKS MADE ON SHA-2 VARIANTS

Paper	Year	Attack Method	Variant of SHA-2	Collision
New Collision attacks Against Up To 24-step SHA-2	2008	Deterministic Collision	SHA-256	In 24/64 rounds with $2^{28.5}$ complexity
			SHA-512	In 24/80 rounds with $2^{32.5}$complexity
Preimages for step-reduced SHA-2	2009	Preimage , Meet-in-the-middle	SHA-256	In 42/64 rounds with $2^{251.7}$complexity In 43/64 rounds with $2^{254.9}$complexity
			SHA-512	In 42/80 rounds with $2^{502.3}$complexity In 46/80 rounds with $2^{511.5}$complexity
Advanced meet-in-the-middle preimage attacks	2010	Preimage , Meet-in-the-middle	SHA-256	In 42/64 rounds with $2^{248.4}$complexity
			SHA-512	In 42/80 rounds with $2^{494.6}$ complexity
Higher-Order Differential Attack on Reduced SHA-256	2011	Pseudo Collision , Differential	SHA-256	In 46/64 rounds with 2^{178} complexity In 46/64 rounds with 2^{46}complexity
Bicliques for Pre-images: Attacks on Skein-512 and the SHA-2 family	2011	Preimage , Biclique	SHA-256	In 45/64 rounds with $2^{555.5}$complexity
			SHA-512	In 50/80 rounds with $2^{511.5}$complexity
			SHA-256	In 52/64 rounds with 2^{555}complexity
			SHA-512	In 57/80 rounds with 2^{511}complexity

Then at ASIACRYPT 2009. Aoki *et al.* presented "Preimages for step-reduced SHA-2" [28] paper which discusses the meet-in-the-middle attack on SHA-256 and SHA-512 with different complexities.

Guo, Jian, San Ling, Christian Rechberger, and Huaxiong Wang also produced meet-in-the-middle attack on SHA-256 and SHA-512 in the paper [29], at the Advances in Cryptology-ASIACRYPT 2010.

Pseudo collision differential attack was presented in "Higher-Order Differential Attack on Reduced SHA-256" [30] by Lamberger, Mario, and Florian Mendel in 2011 on SHA-256 with 2^{178} and 2^{46} complexity.

In 2011, Khovratovich, Dmitry, Christian Rechberger, and Alexandra Savelieva also presented various attacks on SHA-256 and SHA-512 in their paper [31].

D.SHA-3

With the motivation from collision attacks on commonly used hash algorithms such as MD4, MD5, SHA-0, and SHA-1, NIST announced a public competition in the Federal Register to have a new hashing algorithm called SHA-3. The announcement was published in during November, 2007. SHA-3 is not meant to be either directly linked with SHA-2 family or to replace it but it will be preserving some of the properties of SHA-2. There were 64 submissions for the competition in October 2008, out of which 51 candidates were accepted for the first round and then 14 semi-finalists were selected in 2009. Later 5 finalists were selected in December, 2010: BLAKE[32], Grøstl[33], JH [34], Keccak [35] and Skein [36, 37]. On October 2, 2012, Keccak was announced as a winner of the competition by NIST[38].

Keccak[35] was designed by Guido Bertoni, Joan Daemen, and Gilles Van Assche of STMicroelectronics and Michaël Peeters of NXP. It has been found that Keccak has better performance in hardware implementations than the competitors and predecessors. It has an elegant design with ability to execute nicely on different computing devices.

The algorithm uses the sponge construction[39] which is different from the most famous Merkle-Damg˚ard construction. For SHA-3 competition, authors had proposed the largest permutation size of 1600 in the algorithm be named as Keccak-f[1600]. In the construction, the message blocks are first XORed into a subset of state of a 5×5 array of 64-bit values and then permuted as a whole. In each permutation, there is an iteration of a simple round function including operations like bitwise XOR, AND, NOT and rotations [40].

The excellence in hardware performance of Keccak can be seen by the work done by Gürkaynak *et al.* [41], Gaj *et al.* [42], Latif *et al.* [43], Kavun *et al.* [44], Kaps *et al.* [45] and Jungk[46] presented at the Third SHA-3 Candidate Conference. It also gives better software performance than SHA-2 on modern multicore processors. For 128-bit and 256-bit hash codes, you get 4.8 and 5.9 cycles/byte, respectively on a single AMD FX-8120 Bulldozer running at 3.1GHz and 5.4 and 6.9 cycles/byte on a single Intel Xeon E3-1225, Sandy Bridge core running at 3.1 GHz [47]. There are good counter-measures like quadratic round functions and no table look-ups in the keyed Keccak to protect against power analysis attacks, cache-timing attacks, and other variant attacks.

As of April 2014, a separate SHA-3 standard has been announced by NIST as the Draft FIPS Publication 202 and the contents are yet to be finalized for the standard. But the structure of the framework has already been used in various applications [48].

A complete comparison of SHA functions and their variants with respect to the hash size, message size, number of rounds, operations, security, and performance[49] is listed in Table IV.

V. TOWARDS PARALLELIZATION

In applications using CHFs, the performance of these algorithms is a crucial factor. Although performance optimized sequential implementation for these algorithms exist, many of them do not make use of the modern processor architectures that consist of multiple processing cores. Making hashing much faster on modern processors

would open the doors to potentially many more applications while making current usages more secure and convenient. Researchers are striving hard to parallelize the hashing process and make optimum use of the power of the multi-core processors that are commonly available today. In this section, we discuss some of the latest efforts to parallelize CHFs both at the hardware and at the software

TABLE IV
COMPARISON OF SHA FUNCTIONS

			SHA-2		SHA-3
Algorithm	SHA-0	SHA-1	SHA-224 SHA-256	SHA-384 SHA-512	SHA3-224 SHA3-256 SHA3-384 SHA3-512
Hash size (in bits)	160	160	224 256	384 512	224 256 384 512
Internal state size(no. of variables * size in bits)	160 (5×32)	160 (5×32)	256 (8×32)	512 (8×64)	1600 (5×5×64)
Block size (in bits)	512	512	512	1024	1152 1088 832 576
Max message size	2^{64-1}	2^{64-1}	2^{64-1}	2^{128-1}	∞
No. of Rounds	80	80	64	80	24
Operations	add mod 232, and, or, xor, rot	add mod 232, and, or, xor, rot	add mod 232, and, or, xor, shr, rot	add mod 264, and, or, xor, shr, rot	and, xor, not, rot
Security	<80 (collisions found)	<80 (theoretical attack in 2^{61})	112 128	192 256	112 128 192 256
Performance (MiB/s)	-	192	139	154	

levels.

In 1996, Bosselaers, Govaerts, and Vandewalle [50] discussed the possibilities of parallelization with the arrival of the Pentium processors. The cryptographic hash functions such as MD4, MD5, and SHA-1 became faster on the 32-bit processors. The implementation of these algorithms were able to exploit the power of Pentium processors utilizing instruction-level parallelism with the performance gain of approximately 60 percent as compared to the execution on non-parallel architectures. They had also shown that 10 percent of running time performance penalty is sustained by non-cached data and on the endianness conversion.

In contrast to the above claims, in the paper [51], the authors discussed that the implementation of MD4-based CHFs such as the RIPEMD-128, the RIPEMD-160, and the SHA-1 CHF contain more software level parallelism. They estimated that the parallelism found in SHA-1 was a design principle and realizing it will require a 7-way multiple-issue architecture. They have also shown that as the organization of RIPEMD-160 is in two independent lines, future architectures could easily achieve software parallelism due to this structure.

Junko Nakajima and Mitsuru Matsui [52] presented an exhaustive software performance analysis of CHFs MD5, RIPEMD-128 and -160, SHA-1, SHA-256, SHA-512, and Whirlpool on a Pentium III processor. In order to optimize the speed of 32-bit oriented hash functions, they have used pipeline scheduling and MMX registers for processing few of the message blocks in parallel. For 64-bit algorithms, SHA-512 and Whirlpool, they had utilized the 64-bit MMX instructions to maximize the performance. A complete analysis has been provided, which was a first for the SHA-512 and the Whirlpool algorithms.

In 2004, Praveen S.S. Gauravaram, William L. Millan, and Lauren J. May proposed a new cryptographic algorithm CRUSH[53]. In contrast to the standard Merkle Damg˚ard construction algorithms which can be easier in the hands of cryptanalysts, the proposed algorithm was based on iterated halving (IH) to ensure security and efficiency. The authors have claimed to achieve a secure CHF when the internal F-function of IH is instantiated with a half-complexity block cipher. According to [53] 120 Mbits/sec of speed was achieved with an initial un-optimized implementation of the algorithm.

In 2006, a hardware implementation of SHA 512 [54] was proposed. The FPGA implementation used a VHDL description which was synthesized and routed for high performance. Another FPGA-based implementation was designed in 2008 [55] where a digital signature security scheme has been implemented on a public-key crypto system-on-a-Chip (SoC) and included a SHA-2 hash core in combination with a 2048-bit RSA co-processor. The crypto SoC was implemented on an Altera Nios II Stratix FPGA-based prototyping system running on a 50 MHz system clock and showed a throughput of 644 Mbits/sec for the SHA-512 hardware core.

The hardware optimization techniques such as pipelining and unrolling were used [56] to present a new VLSI architecture for the SHA-256 and the SHA-512 hash functions. The processors were developed for implementation on the FPGAs and the results were analyzed and compared with other FPGA-based implementations. The aim of changing hardware implementation was to improve CHF's performance but these techniques were weak in exploiting parallelism in them. It was felt that there is a great need for a more secure along with a finer granularity of parallelism in CHF.

Another way of improving the performance was through the use of GPUs (Graphic Processing Unit). In 2009, an implementation of MD5-RC4 encryption was given using NVIDIA GPU cards in [57, 58]. A performance gain of about 3-5x was achieved on GeForce 9800GTX card. In [58], Hu implemented a parallel MD5 on CUDA-enabled GPU using task stream or task block.

Liu *et al.* [59] have proposed a parallel digital signature method using parallelizing SHA based on content chunk for improving de-duplication storage system performance and it used storage pipelining. The granularity of parallelism proposed by both Hu and Liu is coarse even though both had avoided to exploit parallelism of CHF among intra-stream or intra-task while inter-message work is performed in serial.

In 2009 Du[60] proposed a block cipher based on fine granularity of parallelism for CHFs. They provide theoretical analysis and computer simulation to prove the security and performance of the proposed algorithm. The author has claimed it to be good choice for e-commerce applications but the reduction method used in the paper cannot ensure the security for CHF when the number of message blocks varies.

Hashem Mohammed *et al.* [61] have proposed a parallel algorithm for improving the security performance in SSL bulk data transfer applications. They have proposed a framework in which encryption of the information and calculation of its MAC is done in parallel. The algorithm was simulated on two different processors with one processor performing the MAC calculation and the other one encrypting the data, simultaneously.

In paper [62], Yantao Li *et al.* proposed and analyzed chaotic maps (chaotic asymmetric tent map, chaotic piecewise linear map) based parallel hash algorithm framework with changeable parameters. The key features of proposed algorithm were the parallel processing mode and the message expansion. First, the algorithm converts the expanded message blocks into their respective ASCII codes and then, in order to generate intermediate hash values, iterates the chaotic asymmetric tent map. Once this is done then the chaotic piecewise linear map, uninterruptedly, with the dynamically obtained changeable parameters from the position index of the respective message blocks, generates decimal fractions, rounds the decimal fractions to integers, and cascades the integers. The XOR operation is performed to produce the final hash value of length 128-bit. The authors have claimed good statistical properties, collision resistance, and security against meet-in-the-middle attacks through theoretical analysis as well as computer simulations of the algorithm.

The MD6 Message-Digest Algorithm was one of the contestants of the SHA-3 competition and was designed by Rivest *et al.* [63]. The algorithm uses a Merkle tree-like structure to enable parallel processing while computing hashes for very long inputs. The authors have claimed a performance of 28 cycles/byte on an Intel Core 2 Duo for MD6-256 and verifiable resistance against differential cryptanalysis at the time of its submission. But later, it was found that the claims made regarding MD6’s resistance to differential attacks were for the submitted version and not for a faster reduced-round version. So, Rivest posted a comment at NIST on July 1, 2009, that MD6 is not ready to be a candidate for SHA-3 due to the lack of the proofs on attack resistance. Then in September 2011, a paper[64] was posted on MD6 website supporting MD6 with faster reduced-round versions which are resistant to differential attacks. Unfortunately, MD6 was out of the competition by that time.

In IACR Cryptology, Atighehchi *et al.* [36] had proposed a parallel hash algorithm based on Skein hashing which was one of the candidates of SHA-3 competition organized by NIST. Their preliminary work presents the parallel implementation and associated performance evaluation of available Skein algorithm. To parallelize Skein, they had used the tree hash mode with one virtual thread for each node of the tree. This provides a generic method for the fine grain maximal parallelism approach.

Blake [32], one of the SHA-3 finalists, was based on Dan Bernstein's ChaCha stream cipher. In the algorithm, before each ChaCha round, an input block is XORed after permutation with some of the round constants and added. There were two variants of this algorithm, a 32-bit BLAKE-256 and a BLAKE-224 with output hash sizes of 256 and 224 bits and a 64-bit BLAKE-512 and BLAKE-384 with output hash sizes of 512 and 384 bits, respectively. Some collision attacks were also presented in [65] on the BLOKE and BRAKE versions of the BLAKE algorithm.

Then later, BLAKE2[66] was presented as an improved version of the BLAKE algorithm. The authors claim to have highest security like the SHA-3 and performance similar to MD5 on 64-bit systems using at least 33% less RAM than SHA-2 or SHA-3. The algorithm is based on the same concept as that of the ChaCha stream cipher. There are also two variants of the BLAKE2 algorithm: BLAKE2b (BLAKE2) for 64-bit platforms producing hash of any size ranging 1 to 64 bytes and BLAKE2s for 8 to 32 bit platforms producing hash of any size ranging 1 to 32 bytes. Algorithms give increased performance on parallel systems with capability of keyed hashing, hashing with a salt, updatable or incremental tree-hashing, or any combination these. According to [66], BLAKE2 provides up to 890 MiB/s on a single Intel Xeon E3-1225, Sandy Bridge @3.1GHz core, and up to 559 MiB/s on a single AMD FX-8120 Bulldozer, running at 3.1GHz.

Grøstl [33] was another SHA-3 competition finalist. It is an iterated hash function with its compression function constructed from two fixed large distinct permutations. The components of the algorithm are based on block cipher AES algorithm, as the S-box and the diffusion layers construction are similar to that of AES algorithm leading to the desirable strong confusion and diffusion in the algorithm. The effect of well-known generic attacks has been made difficult by its wide pipe construction in which the size of output is significantly smaller than the size of internal state. The publication claims to have good performance of Grøstl on various platforms with counter-measures against the side-channel attacks. Keccak[35] is another parallel hashing algorithm accepted as new SHA-3 algorithm (discussed in section IV *C*).

In the 4th IEEE International Advance Computing Conference (IACC) 2014, N. Kishore and B.Kapoor [67] proposed a new way of parallelizing the CHFs. The algorithm implements recursive hash construction on multiple-core processor systems. The approach is to break the chain dependencies of the Merkle Damg˚ard construction leading to a faster and secure CHF. They have also discussed collision probability and the performance implications of the algorithm. It was implemented using the OpenMP API and run on AMD FX-8120 Bulldozer at 3.1GHz on an 8-core machine and showing a performance

TABLE V
COMPARISON OF PARALLEL HASHING ALGORITHMS

CPU architecture	Frequency	Algorithm	Technique	Cycles/byte	MiB/s
AMD FX(tm)-8320	3.5 GHz	SHA-1	Merkle Damg˚ard Construction	39.24	75
AMD FX(tm)-8320	3.5 GHz	SHA-256	Merkle Damg˚ard Construction	26.87	110
AMD Barcelona	2.2 GHz	MD6	Tree based Hashing	4.9	427
AMD Barcelona with 8800GT GPU Card	2.2 GHz	MD6	Tree based Hashing	5.5	375
Intel Core 2 Duo	3.1 GHz	Skein- 512	Tree based Hashing	6.5	454
AMD Opteron 6168	1.9 GHz	Grøstl-224/256	Permutation Based	20.7	101.35
Intel Core 2 Duo	2.4GHz	Blake	HAIFA Construction	28.3	80.87
AMD FX(tm)-8320	3.5 GHz	RSHA-1	EITRH Construction	6.35	465

gain of up to 3X. They have also proposed its implementation on the mobile devices as parallel implementation can take advantage of dynamic voltage and frequency scaling techniques to make it more energy efficient.

Table V shows a comparison of some of the recently developed parallel CHFs using the cycles/byte and the MiB/s metrics. The CPU architecture, core frequency, construction method, and the algorithm parallelized have been listed as well in the table.

VI. CONCLUSION AND SUMMARY

Cryptographic hash functions have gathered an unprecedented interest among researchers in recent years. With the advances in hardware and software, the attacks have also become more efficient and common. In this paper, we have surveyed the literature related to the advances in the CHFs for the readers. We have discussed the role of CHFs in security along with the basic security necessities for a function to become a secure CHF. In the second part of the paper, we have focused on the SHA family covering SHA-0, SHA-1, SHA-2, and SHA-3 advances along with the documented attacks on these algorithms. It is further supported by a tabular comparison of all the variants of SHA considering key parameters.

The third part of the paper covers the parallelization of CHFs both at hardware and software level. It also covers the parallelization of existing and recently developed parallel CHFs. The SHA-3 implementation and the related competition for secure CHFs has drawn everybody’s attention towards the parallelization research for the CHFs.

REFERENCES

[1] S. William and W. Stallings, "Cryptography and Network Security, 4/E," 2006.

[2] M. A. Alia, A. A. Tamimi, and O. N. AL-Allaf, " Cryptography Based Authentication Methods," *Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2014, WCECS 2014, 22-24 October, 2014, San Francisco, USA, pp 199-204.*

[3] C. Chandersekaran and W. R. Simpson, "Cryptography for a High-Assurance Web-Based Enterprise," *Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2013, WCECS 2013, 23-25 October, 2013, San Francisco, USA, pp 23-28.*

[4] N. Zhang, K. L. Man, T. Yu, and C.-U. Lei, "Text and content based image retrieval via locality sensitive hashing," *Engineering Letters,* vol. 19, pp. 228-234, 2011.

[5] M. A. B. Younes and A. Jantan, "Image Encryption Using Block-Based Transformation Algorithm," *IAENG International Journal of Computer Science,* vol. 35, pp15-23,2008.

[6] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*: CRC press, 2010.

[7] V. Klima, "Tunnels in Hash Functions: MD5 Collisions Within a Minute," *IACR Cryptology ePrint Archive,* vol. 2006, p. 105, 2006.

[8] M. Stevens, A. Sotirov, J. Appelbaum, A. Lenstra, D. Molnar, D. A. Osvik, *et al.,* "Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate," in *Advances in Cryptology-CRYPTO 2009*, ed: Springer, 2009, pp. 55-69.

[9] P. Rogaway, "Formalizing human ignorance," in *Progress in Cryptology-VIETCRYPT 2006*, ed: Springer, 2006, pp. 211-228.

[10] *NIST.* Available: http://www.nist.gov/

[11] *FIPS.* Available: http://www.nist.gov/itl/fips.cfm

[12] I. Mironov, "Hash functions: Theory, attacks, and applications," *Microsoft Research, Silicon Valley Campus. Noviembre de,* 2005.

[13] X. Chan and G. Liu, "Discussion of One Improved Hash Algorithm Based on MD5 and SHA1," *Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2007, WCECS 2007, 24-26 October, 2007, San Francisco, USA, pp 270-273.*

[14] F. Chabaud and A. Joux, "Differential collisions in SHA-0," in *Advances in Cryptology—CRYPTO'98,* 1998, pp. 56-71.

[15] E. Biham and R. Chen, "Near-collisions of SHA-0," in *Advances in Cryptology–CRYPTO 2004,* 2004, pp. 290-305.

[16] A. Joux, P. Carribault, C. Lemuet, and W. Jalby, "Collision in SHA-0," *Announced in sci. crypt on,* vol. 12, 2004.

[17] X. Wang, D. Feng, X. Lai, and H. Yu, "Collisions for hash functions MD4," *MD5, HAVAL-128 and RIPEMD, rump session of Crypto,* vol. 4, 2004.

[18] X. Wang, H. Yu, and Y. L. Yin, "Efficient collision search attacks on SHA-0," in *Advances in Cryptology–CRYPTO 2005,* 2005, pp. 1-16.

[19] (15th August). *SHA-1.* Available: http://en.wikipedia.org/wiki/SHA-1

[20] V. Rijmen and E. Oswald, "Update on SHA-1," *Topics in Cryptology–CT-RSA 2005,* pp. 58-71, 2005.

[21] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full SHA-1," in *Advances in Cryptology–CRYPTO 2005,* 2005, pp. 17-36.

[22] M. Cochran, "Notes on the Wang et al. 263 SHA-1 Differential Path," *IACR Cryptology ePrint Archive,* vol. 2007, p. 474, 2007.

[23] C. De Canniere and C. Rechberger, "Finding SHA-1 characteristics: General results and applications," *Advances in Cryptology–ASIACRYPT 2006,* pp. 1-20, 2006.

[24] S. Manuel, "Classification and generation of disturbance vectors for collision attacks against SHA-1," *Designs, Codes and Cryptography,* vol. 59, pp. 247-263, 2011.

[25] C. McDonald, P. Hawkes, and J. Pieprzyk, "Differential Path for SHA-1 with complexity O(2^52)," *IACR Cryptology ePrint Archive,* vol. 2009, p. 259, 2009.

[26] (15 August). *SHA-2.* Available: en.wikipedia.org/wiki/SHA-2

[27] S. K. Sanadhya and P. Sarkar, "New collision attacks against up to 24-step SHA-2," *Progress in Cryptology-INDOCRYPT 2008,* pp. 91-103, 2008.

[28] K. Aoki, J. Guo, K. Matusiewicz, Y. Sasaki, and L. Wang, "Preimages for step-reduced SHA-2," *Advances in Cryptology–ASIACRYPT 2009*, pp. 578-597, 2009.

[29] J. Guo, S. Ling, C. Rechberger, and H. Wang, "Advanced meet-in-the-middle preimage attacks: First results on full Tiger, and improved results on MD4 and SHA-2," *Advances in Cryptology-ASIACRYPT 2010*, pp. 56-75, 2010.

[30] M. Lamberger and F. Mendel, "Higher-Order Differential Attack on Reduced SHA-256," *IACR Cryptology ePrint Archive*, vol. 2011, p. 37, 2011.

[31] D. Khovratovich, C. Rechberger, and A. Savelieva, "Bicliques for preimages: attacks on Skein-512 and the SHA-2 family," in *Fast Software Encryption*, 2012, pp. 244-263.

[32] J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W. Phan, "Sha-3 proposal blake," *Submission to NIST*, 2008.

[33] P. Gauravaram, L. R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, *et al.*, "Grøstl–a SHA-3 candidate," *Submission to NIST*, 2008.

[34] R. Bhattacharyya, A. Mandal, and M. Nandi, "Security analysis of the mode of JH hash function," in *Fast Software Encryption*, 2010, pp. 168-191.

[35] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Keccak," in *Advances in Cryptology–EUROCRYPT 2013*, ed: Springer, 2013, pp. 313-314.

[36] K. Atighehchi, A. Enache, T. Muntean, and G. Risterucci, "An Efficient Parallel Algorithm for Skein Hash Functions," *IACR Cryptology ePrint Archive*, vol. 2010, p. 432, 2010.

[37] E. Andreeva, B. Mennink, B. Preneel, and M. Škrobot, "Security analysis and comparison of the SHA-3 finalists BLAKE, Grøstl, JH, Keccak, and Skein," in *Progress in Cryptology-AFRICACRYPT 2012*, ed: Springer, 2012, pp. 287-305.

[38] C. Boutin, "NIST selects winner of Secure Hash Algorithm(SHA-3) Competition," *Press release., October*, vol. 2, 2012.

[39] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Sponge functions," in *ECRYPT hash workshop*, 2007.

[40] (18 August). *SHA-3*. Available: http://en.wikipedia.org/wiki/SHA-3

[41] F. K. Gürkaynak, K. Gaj, B. Muheim, E. Homsirikamol, C. Keller, M. Rogawski, *et al.*, "Lessons learned from designing a 65nm ASIC for evaluating third round SHA-3 candidates," in *Third SHA-3 Candidate Conference (March 2012)*, 2012.

[42] K. Gaj, E. Homsirikamol, M. Rogawski, R. Shahid, and M. U. Sharif, "Comprehensive Evaluation of High-Speed and Medium-Speed Implementations of Five SHA-3 Finalists Using Xilinx and Altera FPGAs," *IACR Cryptology ePrint Archive*, vol. 2012, p. 368, 2012.

[43] K. Latif, M. M. Rao, A. Aziz, and A. Mahboob, "Efficient hardware implementations and hardware performance evaluation of sha-3 finalists," in *The Third SHA-3 Candidate Conference*, 2012.

[44] E. B. Kavun and T. Yalcin, "On the suitability of SHA-3 finalists for lightweight applications," in *ser. The Third SHA-3 Candidate Conference*, 2012.

[45] J.-P. Kaps, P. Yalla, K. K. Surapathi, B. Habib, S. Vadlamudi, and S. Gurung, "Lightweight implementations of SHA-3 finalists on FPGAs," in *The Third SHA-3 Candidate Conference*, 2012.

[46] B. Jungk, "Evaluation of compact FPGA implementations for all SHA-3 finalists," in *The Third SHA-3 Candidate Conference*, 2012.

[47] (1 September). *Keccak*. Available: http://keccak.noekeon.org/

[48] P. Caballero-Gil, F. Martın-Fernández, and C. Caballero-Gil, "Tree-Based Management of Revoked Certificates in Vehicular Ad-hoc Networks," *Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering 2013, WCE 2013, 3-5 July, 2013, London, U.K., pp 1425-1430*.

[49] *Performance on AMD Opteron*. Available: http://www.cryptopp.com/benchmarks-amd64.html

[50] A. Bosselaers, R. Govaerts, and J. Vandewalle, "Fast hashing on the Pentium," in *Advances in Cryptology—CRYPTO'96*, 1996, pp. 298-312.

[51] A. Bosselaers, R. Govaerts, and J. Vandewalle, "SHA: a design for parallel architectures?," in *Advances in Cryptology—EUROCRYPT'97*, 1997, pp. 348-362.

[52] J. Nakajima and M. Matsui, "Performance analysis and parallel implementation of dedicated hash functions," in *Advances in Cryptology—EUROCRYPT 2002*, 2002, pp. 165-180.

[53] P. Gauravaram, W. Millan, and L. May, "CRUSH: A New Cryptographic Hash Function using Iterated Halving Technique," in *Cryptographic Algorithms and their Uses*, 2004, pp. 28-39.

[54] L. Hong-Qiang and M. Chang-yun, "Hardware Implementation of Hash Function SHA-512," in *Innovative Computing, Information and Control, 2006. ICICIC'06. First International Conference on*, 2006, pp. 38-42.

[55] M. Khalil, M. Nazrin, and Y. Hau, "Implementation of SHA-2 hash function for a digital signature System-on-Chip in FPGA," in *Electronic Design, 2008. ICED 2008. International Conference on*, 2008, pp. 1-6.

[56] R. P. McEvoy, F. M. Crowe, C. C. Murphy, and W. P. Marnane, "Optimisation of the SHA-2 family of hash functions on FPGAs," in *Emerging VLSI Technologies and Architectures, 2006. IEEE Computer Society Annual Symposium on*, 2006, p. 6 pp.

[57] C. Li, H. Wu, S. Chen, X. Li, and D. Guo, "Efficient implementation for MD5-RC4 encryption using GPU with CUDA," in *Anti-counterfeiting, Security, and Identification in Communication, 2009. ASID 2009. 3rd International Conference on*, 2009, pp. 167-170.

[58] G. Hu, J. Ma, and B. Huang, "High throughput implementation of MD5 algorithm on GPU," in *Ubiquitous Information Technologies & Applications, 2009. ICUT'09. Proceedings of the 4th International Conference on*, 2009, pp. 1-5.

[59] C. Liu, Y. Xue, D. Ju, and D. Wang, "A novel optimization method to improve de-duplication storage system performance," in *Parallel and Distributed Systems (ICPADS), 2009 15th International Conference on*, 2009, pp. 228-235.

[60] M. Du, B. He, Y. Wang, J. Wu, and D. Xiao, "Parallel Hash Function Based on Block Cipher," in *E-Business and Information System Security, 2009. EBISS'09. International Conference on*, 2009, pp. 1-4.

[61] H. M. Alaidaros, "Enhancing Secure Sockets Layer Bulk Data Trnsfer Phase Performance With Parallel Cryptography Algorithm," 2007.

[62] Y. Li, D. Xiao, S. Deng, Q. Han, and G. Zhou, "Parallel Hash function construction based on chaotic maps with changeable parameters," *Neural Computing and Applications*, vol. 20, pp. 1305-1312, 2011.

[63] R. L. Rivest, B. Agre, D. V. Bailey, C. Crutchfield, Y. Dodis, K. E. Fleming, *et al.*, "The MD6 hash function–a proposal to NIST for SHA-3," *Submission to NIST*, vol. 2, p. 3, 2008.

[64] E. Heilman, "Restoring the differential security of MD6," in *ECRYPT II Hash Workshop*, 2011.

[65] J. Vidali, P. Nose, and E. Pašalić, "Collisions for variants of the BLAKE hash function," *Information Processing Letters*, vol. 110, pp. 585-590, 2010.

[66] (5 September). *BLAKE2*. Available: https://blake2.net/

[67] N. Kishore and B. Kapoor, "An efficient parallel algorithm for hash computation in security and forensics applications," in *Advance Computing Conference (IACC), 2014 IEEE International*, 2014, pp. 873-877.

Dr. Neha Kishore (M'14) was born in Chandigarh, India in 1984. She has done her PhD in Computer Science and Engineering from Chitkara University, India in year 2015. Her area of research includes Parallel Computing and Information Security.

She is working as an Associate Professor in Chitkara University, H.P., India for last six years. She has many research papers and poster presentations at International Journals/Conferences in her credits.

Dr. Kishore is a member of ACM, IAENG, Internet Society, UACEE. She has been certified as an ACM Ambassador.

Dr. Bhanu Kapoor started his technical career in 1987 with Texas Instruments. Since 1996, he has taught several undergraduate and graduate courses in the areas of computer science and electrical engineering. He has received six U.S. patents, and has participated in various industry panels. His written works include more than 50 papers that have been presented at IEEE/ACM conferences or published in journals.