International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 07 | July 2020 www.irjet.net p-ISSN: 2395-0072

Hash functions and its security for Snags

Jalagam Lokesh Naidu[1], Akhil Chandra Gorakala[2], Shanmuk Srinivas Amiripalli[3]

[1]Student, Department of Computer Science and Engineering, GITAM deemed to be University, Visakhapatnam, Andhra Pradesh, India.
[2]Student, Department of Computer Science and Engineering, GITAM deemed to be University, Visakhapatnam, Andhra Pradesh, India.
[3]Assistant Professor, Department of Computer Science and Engineering, GITAM deemed to be University, Visakhapatnam, Andhra Pradesh, India.

Abstract - *Cryptographic hash functions arbitrarily compress long messages to mutates a short and fixed length, which are significant due to their use in data integrity, message authentication, and also in key generations (Symmetric and Public-key Cryptosystems). Many of the existing hash functions are designed to evaluate a finite domain compression function in a mode of operation, and the compression function itself is mostly based on block ciphers or permutations. This modular design approach allows for a rigorous analysis of security through both cryptanalysis and proven security. Different cryptographic implementations depend on the performance and strength of the hash functions to satisfy the need for integrity and authentication. This research paper provides an overview and importance of cryptographic hash functions in this field with their architectures, various attacks and progressive implementation. Hash functions selected in projects NESSIE and CRYPTREC are addressed briefly. SHA-3 Selection initiative is also enacted.*

Key Words: **Cryptography, Hash functions, Data integrity and Network Security.**

1. INTRODUCTION

Network Security means protecting User confidentiality, integrity, and resource availability[1]. Network Security initializes with authorization i.e. with the help of credentials such as a username and a password to access a specific device commonly. Network security consists of policies each network administrator has adopted to prevent and track illegitimate access privileges, alteration and denial of a computer network and network resources. When a user is approved to do something else, a firewall will require them to follow rules such as what resources the network user is allowed to access. Thus, these policies are reasonable to prevent unauthorized access to the device, but this component may fail to track potentially dangerous content such as software warms or network transmission of Trojans.

Anti-virus or Intrusion Detection System (IDS) software helps to detect the Malware. Communication between two hosts using a network can be used to encrypt a privacy policy. However apart from encryption-decryption techniques, hash functions are ubiquitously used for authentication. The world is becoming more interconnected with the Internet and with modern developments in networking. A substantial percentage of information and networking infrastructure and services available worldwide are personal, military, commercial, and governmental. As such, it is important to find out who will be transmitting critical data and who will be receiving it, the accountability policies take care of it as well. But it is an underlying and un-manipulated way of identifying that data received by one user is sent by the valid user, or the data received. All these issues are solved by using hash function to demonstrate the validity of the data and the user.

A cryptographic hash function maps a binary string of variable length (a message or a file) to a binary string (message digest) of fix length n, with n sometimes indicated in the name of the hash function algorithm (SHA-256, SHA-512, RIPEMD-160, etc.). Hash functions are used in cryptography for data integrity and message (data origin) authentication.

1.1 CRYPTOGRAPHIC HASH FUNCTIONS

The word " hash function", used in computer science, refers to a function which compresses an arbitrary length message to a fixed-length message called the Message Digest. Nevertheless, if it fulfills any additional criteria, then it can be used in cryptographic applications and then called as the Cryptographic Hash functions.

Cryptographic Hash functions are the most powerful tool in the Security and Cryptography stream and are used to achieve many security requirements such as authentication, digital signatures, generation of pseudo numbers, digital time-stamping, etc.

There are two types of Hash functions: Keyed and Un-Keyed. Keyed hash functions use a hidden key to compute the message digest and these are also known as MAC (Message Authentication Code) but we don't use any secret key in almost any.

Merkel defined one-way hash function (OWHF)[2] as that a hash function H meets the following requirements:

1. H can be applied to Block of data of any length. (any length means size of Block must be greater than size of Digest we conclude at the end).
2. H produces a fixed length output i.e., Message

Digest.

3. Given H and x (any given input), it is easy to computer Message Digest H(x).
4. Given H and H(x). it is computationally infeasible to find x.
5. Given H and H(x), it is computationally infeasible to find x und x ' such that H(x)=H(x')

For practical implementations of hash functions for message authentication and digital signatures the first three specifications are a must. Also known as the pre-image resistance or one-way property, the fourth requirement states that generating a message code from a given message is easy but hard to generate a message back from the given message digest. Also known as the Second Pre-Image Resistance or Collision Resistance Property, the fifth condition guarantees that an alternative message that has to the same code as a given message cannot be bound.

1.2 Usage of Hash Function

Hash functions are used in combination with digital signature systems to ensure data integrity. The message is initially hashed, and then the hash-value (message digest), as a representative of the message, is signed instead of the original message. This saves time and space compared to signing the entire message (block by block) in this way.

The hash value (message digest) corresponding to a particular (original) message is initially determined. In certain cases it preserves the credibility of this hash-value (but not the message itself). After, the following check is performed to determine whether the message has been changed, i.e. whether a message is the same as the original message. The message's hash value is calculated and compared to the protected hash value; if they are equal, one accepts that the inputs are also equal, and therefore the message was not altered. This reduces the question of maintaining the credibility of a potentially big message to that of a small fixed-size hash value[3].

In order to be effective from a cryptographic point of view a hash function must fulfill certain requirements:

(a) to be difficult for two distinct messages to have the same hash value
(b) Knowing a hash value to be computation infeasible is to find a message with that hash value.

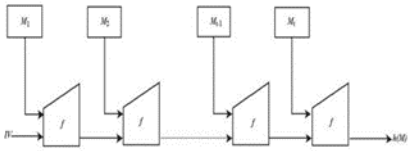
2. Structure of Hash Function

Different hash design structures are feasible, but we will rely on only two designs which are mainly used in all NIST standardized hash functions. They are:

a) Merkle-Damgard Iterated Hash Design and
b) Sponge Construction

2.1 Merkle-Damgard Construction

From the early onset of cryptographic hash functions, designers relied on the Merkle-Damgård construction (abbreviated to MD). In 1989 the MD architecture was independently discovered by Merkle[4] and Damgård[5] . Most of the famous hash functions like MD4[6], MD5[7], SHA-0[8], SHA-1[9], RIPEMD-160[10] and so on follow the iterative MD method. The central component of this construction is a compression function that takes a fixed input length value, and outputs a fixed-length hash value.

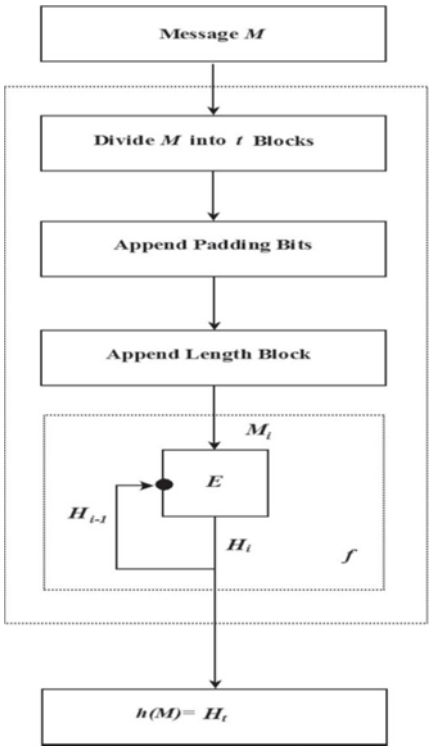


A compression function accepts two inputs: a chaining variable, and a message block. Let $f : \{0,1\}^b \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a compression function which takes a b-bit message block and an n-bit chaining value. Let $h: \{0,1\}^*$ be a MD construction built by iterating the compression function f in order to process a message of arbitrary length. A message M to be processed using h is always padded in a manner such that the length of the padded message is a multiple of the block length b of f. Bit-length b corresponds to input length of desired compression function f. The padding is done by adding after the last bit of the last message block a single 1-bit followed by the necessary number of 0-bits. Let |M| be a binary representation of the length of the message M. The binary encoding of the message length is also be added to complete the padding. This is called a Merkle-Damgård strengthening. Then input M subsequently divided into t blocks, each of bit-length b.

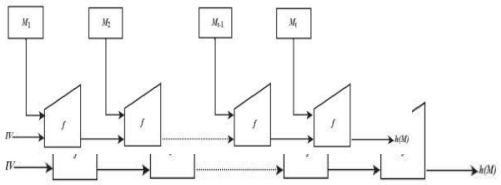
The hash function h can then be described as follows:

$$\left.\begin{aligned} H_0 &= IV, \\ H_i &= f(H_{i-1}, M_i), i = 1 \ldots t, \\ h(M) &= H_t \end{aligned}\right\}$$

where f is the compression function of h, H_i is the intermediate chaining variable between stage i-1 and stage i, and H_0 is a pre-defined starting value or the initial. value IV. The block diagram of the iterative hash function using the compression function is shown in the Fig.1. The computation of the hash value is dependent on the chaining variable. At the start of hashing, this chaining variable has a fixed initial value which is specified as part of the algorithm.



This process goes on recursively, with the chaining variable being updated under the action of another part of the message until the whole message is used. The chaining variable's final value is then output as the



corresponding hash value for that message. One of its distinctive features is that it promotes the compression function's collision resistance and pre-image resistance to the full hash function: For example, a collision on the compression function can be effectively deduced from a collision on the full hash function. For this situation, the inclusion of the length at the end of the message is important and is also important in order to prevent several attacks, including long-message attacks.

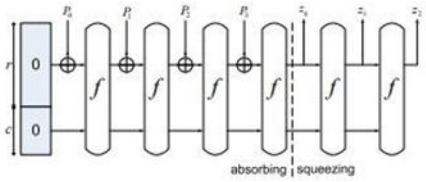
Merkle-Damgård construction proves that hash function safety is based on compression function security. Thus, designing a collision-resistant compression function is enough to build a collision-resistant hash function. Nonetheless, recent studies illustrate some inherent shortcomings of the MD method. This includes being vulnerable to multi-collision attacks[11], long second pre-image attacks[12] and an attack on herding[13]. A detailed view of the MD construction is shown in Fig.2.

2.2 Sponge Construction

G. Bertoniet[14] proposed the construction of a sponge to design hash functions that map the random oracle. The construction of the sponge operates at the state of b= r + c bits, r is called bit rate and c is capacity. All bits of state are initially set to zero, and the message is padded and separated into blocks of r bits each. Sponge construction then proceeds in two phases: Absorbing Phase and Squeezing Phase. In the first phase, at a given rate, the input is "absorbed" into the hash state, then at the same rate an output hash is "squeezed" from it. To absorb data bits r. The data is XOR into the state 's leading bits, and the block permutation is used.

To squeeze, the first r bits of the state are produced as output, and if the additional output is needed, the block permutation is applied. Capacity c of hash function is key to the Sponge Construction, and it can be modified according to security requirements.

SHA-3 final round candidate algorithm Keccak is a Sponge construction only hash function and it sets a conservative c=2n where n is the output hash size.



3. Properties of hash functions

In this section we are shortly presenting the hash functions evaluated and selected in NESSIE and CRYPTREC projects.

3.1 NESSIE research project

The NESSIE (New European Schemes for Signature, Integrity, and Encryption)[15], a European-funded IST initiative, aimed at selecting powerful cryptographic primitives of different types (block ciphers, stream ciphers, digital signature algorithms, hash functions, etc.). The project started with an open call for the submission of cryptographic primitives as well as methodologies for evaluation of those primitives. The call's spectrum was made public in March 2000[16]. The key selection criteria defined in NESSIE call for cryptographic primitives have been long-term security, market requirements, efficiency and flexibility. Along with the algorithms provided for evaluation, well established standard algorithms have been added for evaluation. Whirlpool algorithm for the category hash functions and UMAC and Two-Track-MAC for MAC was submitted. In February 2003, the project consortium NESSIE announced the final set of cryptographic algorithms. Whirlpool (proposed by Scopus Tecnologia S.A., Brazil, and K.U.Leuven, Belgium) and SHA-256, SHA-384,

and SHA-512 (added for testing, part of the USA FIPS 180-2 standard) were chosen in the category of hash function.

The SHA (Secure Hash Algorithm) hash functions are a set of cryptographic hash functions designed by the National Security Agency (NSA) and published by the NIST as a U.S. Federal Information Processing Standard. The three SHA algorithms (SHA-0, SHA-1, and SHA-2) have different structures. The SHA-2 family uses an identical algorithm with a variable digest size i.e. SHA-256, SHA-384, and SHA-512. SHA-2 family returns a number of bits identical with the number in the name and uses for operation messages not longer than 264-1, 2128-1, and 2128-1 bits respectively.

Whirlpool is a Square block cipher based hash function (also used to design Rijndael which was selected as the Advanced Encryption Standard (AES)). Whirlpool returns a 512-bit digest from a message which can have a total of two 256 bits. As part of the joint international standard ISO / IEC 10118-3 (International Organization for Standardization (ISO) and International Electro technical Commission (IEC)), Whirlpool was adopted.

Algorithm	*Hash length*	*Block size*	*Max. message length*
SHA-256, SHA-384, and SHA-512	256, 384, and 512	512, 1024, and 1024	2^{64}-1, 2^{128}-1, and 2^{128}-1 bits
Whirlpool	512	512	2^{256} bits

Two-Track-MAC (proposed by K.U.Leuven, Belgium and debis AG, Germany), UMAC (proposed by Intel Corp., USA, Univ. of Nevada at Reno, USA, IBM Research Laboratory, USA, Technion, Israel and Univ. of California at Davis, USA), and, CBC-MAC (ISO/IEC 9797-1) and HMAC (ISO/IEC 9797-1) have been selected at MAC category (last two, have been added for evaluation as MAC standards.)

3.2 CRYPTREC IPA research project

The Information-technology Promotion Agency (IPA) in Japan has introduced the CRYPTREC project (CRYPTography Study and Evaluation Committees) with the aim to establish standard cryptographic algorithms for use within the Japanese e-government infrastructure.[17]

CRYPTREC Project began in the year 2000. The formal Call for Cryptographic Techniques received various types of cryptographic techniques.

As for the NESSIE project, the CRYPTREC call was available to different types of primitives. Some of the algorithms evaluated for the NESSIE project (e.g. RC6, MISTY1, Camellia, AES) were also submitted for evaluation to CRYPTREC. In CRYPTREC, as in the evaluation of NESSIE, several well-known or common primitives were included for assessment. No algorithm has been provided for the hash function group. However, due to their use of internet security mechanisms the CRYPTREC team included MD5, RIPEMD-160, SHA-1 for evaluation.

After evaluation, only RIPEMD-160, SHA-1, SHA-256, SHA-384, SHA-512 with a note for RIPEMD-160, SHA-1 was suggested in 2003, that if any cipher with a longer hash value is available, it is preferable to select a 256-bit (or more) hash function. This does not apply, however, in cases where the hash to be used has already been constructed according to the requirements of the cryptographic public-key.

- RIPEMD-160 is a 160-bit message-digest algorithm (and cryptographic hash function) developed in K.U.Leuven (Belgium) and first published in 1996. It is an improved version of RIPEMD, and uses the design principles of MD4 and has performance similar to SHA-1.
- 128, 256 and 320-bit variants of this algorithm are RIPEMD-128, RIPEMD-256 and RIPEMD320. The 256 and 320-bit models reduce collision risks, without achieving a higher degree of protection compared to RIPEMD-160.

Algorithm	*Hash length*	*Block size*
RIPEMD-160	160	512
RIPEMD-256, RIPEMD -384, RIPEMD-512	256, 384, and 512	512, 512, 512

Table II summarizes the hash function parameters which satisfy the CRYPTREC research project requirements.

4. Attacks Targeting Hash Functions

The properties of hash functions are defined in this section using the expression "Computationally Infeasible." Given the current computing power, all properties are fulfilled; as computing power rises each year, even though no new attacks are created, the increase in computing power weakens the resistance of current hash functions.

Recently new attacks[18] have been released concerning collision resistance[19] of hash functions. These attacks attempt to evaluate 2 messages using fewer operations which have the same hash.

An attack on MD5[20] was presented in 2005, using differential analysis, which allows effectively finding collisions.

Applied on HAVAL-128, MD4, RIPEMD[20], and SHA-0[21], the same attack reduced the number of operations to determine a second message with the same hash.

Even if the number of operations needed to split the hash functions is significant, these attacks reduce the ideal number of operations that are supposed to be necessary. These results inspired NIST to discover new hash functions[22] that were resistant. Other attacks are posed that address hash functions and message authentication codes[23].

5. Hash Functions Based on Compression Functions

In this section, we consider open problems related to the construction of compression based hash functions. This is also known as domain-extending or operating mode, where a compression function f is applied to a hash function H with a virtually infinitely large domain with a certain fixed and finite domain.

The iterated hash function principle is central to many hash function designs: at the input of an initialization vector IV, the iterated hash function Hf[24], based on the compression function f, processes a padded message (M1,M2,.., Mk) as follows:

$\boldsymbol{H}^{f}$ (IV; $M1, \ldots, M_k$) = h_k, where:

$h0$ = IV,

$h_i = f(h_i-1, M_i)$ for $i = 1, \ldots, k$.

This principle is also called the plain Merkle-Damgard(MD) design[25][26].

The specific padding rule for Merkle- Damgard ensures suffix-freeness by appending the length of the message in the last block. Popular hash functions including SNERFU, MD4[27], MD5[28], RIPEMD, HAVAL[29],WHIRLPOOL[30], the SHA family[31], and numerous other hash functions have adopted the Merkle-Damg gleichard theory. In addition, other domain extensions such as HAIFA[32] and dither hash[33] expand the iterative method of Merkle-Damg by adding different counters or changing inputs to the compression function.[34]

5.1 Ideal Model Security Results

In the ideal model one assumes that the domain extensor ***H*** 's underlying compression function(s) f is an ideal function, namely a random oracle with a fixed input and output length[35]. These are commonly referred to as generic security results, which ensure no structural defects are visible in the hash function.

There are two types of results on security.[36] One type of results includes proving a typical ***H*** protection property atk when ***f*** is an ideal function and atk ∈{col,sec, pre}. Similarly, we have protection tests of in-differentiability[37] where the aim is to prove that ***H*** itself behaves like a random oracle on arbitrary input lengths provided that ***f*** is an ideal function. In-differentiability is a better form of in-distinguishability (or pseudo randomness), which is only a meaningful notion when keyed with the algorithms (e.g. block ciphers). In-differentiability comes therefore handy in the hash function setting most notably to allow one to obtain a bound on the adversarial

advantage against some atk for H denoted as $\mathbf{Adv}^{atk}$: for any hash function security ***H*** notion atk: $\mathbf{Adv}^{atk} \leq \mathbf{Pr}_{RO}^{atk} + \mathbf{Adv}_{H}^{pro}$, where $\mathbf{Pr}_{RO}^{atk}$ denotes the success probability of a generic attack against ***H*** behaving like a random oracle under **atk**. Coron et al. prove that the MD design with suffix-free padding and idealized compression function from a random oracle is not in-differentiable. Their observation formalizes the length extension attack: one can compute ***H*** (M||M') from ***H*** (M) and M' and M' even without understanding M, which is highly undesirable for certain applications (note that padding allows abstraction here for convenience, but this can be easily addressed). The MD construction is, however, indifferent whether it finishes with a chopping function (where a part of the output bits is truncated) or a final transformation, either where the underlying compression function is optimal or where the hash function is based on a PGV compression function. It has also been shown that the MD design based on an ideal compression function or idealized PGV construction, with prefix-free padding, is in-differentiable from a random oracle.

Nevertheless, security notions such as collision resistance are not retained in the regular model's MD design with prefix-free padding only. In the other hand, domain extenders that perform well on multiple protection of property, such as BCM[38] and XOR-(linear) hash[39], also lack a security review under the notion of in-differentiality.

5.2 New Domain Extenders

With the respective digest sizes, novel hash domain extensors based on compression functions need to come with at least comparable security and efficiency to SHA-2. Breaking the sequential structure is one way which may result in tree style designs. Known collision preserving tree-based hash functions are the (strengthened) Merkle tree[40], XOR-tree hash, and a variant of the trees preserving the latter second pre-image security[41][42]. Skein also comes with a tree alternative to the second round SHA-3 candidate hash function.

Finally, we mention the tree hash functions of Rivest et al.[43] and Bertoni et al.[44] where the latter also studied the properties of tree hash functions necessary for obtaining in-differentiable designs. There are diverse directions in this area. Some of them include analysis of tree-based alternatives or generic transformations that turn into their parallel (-izable) counterpart a ready-made sequential domain extensor. Bertoni et al used a similar approach in the design of the Sakura[45].

6. CONCLUSIONS

After the study, experiments, testing, and evaluation of the process and the program, it has drawn a remarkable conclusion. Cryptography and Hashing function's based on compression function's adds additional security to the system or device's. Among the three security results, Ideal Model Security results are proven to be the best by using domain extensor ***H***'s underlying compression function(s)

f followed by next best results of New Domain Extender's using novel hash domain extender's based on compression functions and efficiency to SHA-2 and SHA-3.

REFERENCES

[1] Kanta, Harshitha & Naidu, Jalagam Lokesh & Srinivas, Amiripalli. (2019). Data Security Using Cryptographic-Steganography. 10.13140/RG.2.2.16337.99684.

[2] R. C. Merkle, "One Way Hash Functions and DES," Crypto'89, 1989, LNCS, vol. 435, pp. 428-446.

[3] Amiripalli, S. S., & Bobba, V. (2018). Research on network design and analysis of TGO topology. International Journal of Networking and Virtual Organisations, 19(1), 72-86.

[4] R. C. Merkle, "One Way Hash Functions and DES," Crypto'89, 1989, LNCS, vol. 435, pp. 428-446.

[5] I. Damgård, "A Design Principle for Hash Functions," Crypto'89, 1989, LNCS, vol. 435, pp. 416-427.

[6] R. Rivest, "The MD4 Message Digest Algorithm," Request for Comments (RFC) 1320, Internet Engineering Task Force, 1992.
http://www.rfceditor.org/rfc/pdfrfc/rfc1320.txt.pdf.

[7] R. Rivest, "The MD5 Message Digest Algorithm," Request for Comments (RFC) 1321, Internet Engineering Task Force, 1992.

[8] NIST, "Secure Hash Standard (SHS)," Federal Information Processing Standards 180. 1993.

[9] NIST, "Secure Hash Standard (SHS)," Federal Information Processing Standards 180-1, 1995.

[10] B. Preneel, A. Bosselaers and H. Dobbertin, "RIPEMD-160: A Strengthened Version of RIPEMD," FSE'96, 1996, LNCS, vol. 1039, pp. 71–82.

[11] A. Joux, "Multicollisions in Iterated Hash Functions: Application to Cascaded Constructions," Crypto'04, 2004, LNCS, vol. 3152, pp. 306-316.

[12] J. Kelsey and B. Schneier, "Second Preimages on n-bit Hash Functions for Much Less than 2^n work," Eurocrypt'05, 2005, LNCS, vol. 3494,pp.474-490.

[13] J. Kelsey and T. Kohno, "Herding Hash Functions and the Nostradamus Attack," Eurocrypt'06, 2006, LNCS, vol. 4004, pp. 183–200.

[14] Guido Bertoni, Keccak sponge function family, Version 1.2, April 23,2009.

[15] NESSIE consortium, NESSIE project announces final selection of crypto algorithms, February 27, 2003, available at: https://www.cosic.esat.kuleuven.ac.be/

[16] NESSIE, Call for Cryptographic Primitives, Version 2.2, 8th March 2000, available at: https://www.cosic.esat.kuleuven.ac.be/ nessie/call/, last visited March 2009.

[17] CRYPTEC site, Evaluation of Cryptographic Techniques, available at: http://www.ipa.go.jp/ security/enc/CRYPTREC/index-e.html, last visited March 2009.

[18] Xiaoyun Wang and Hongbo Yu, How to Break MD5 and Other Hash Functions, Advances in Cryptology – EUROCRYPT 2005, 2005, pages 19-35.

[19] E. Biham, R. Chen, Near collision for SHA-0, Advances in Cryptology, Crypto'04, 2004, LNCS 3152, pp. 290- 305.

[20] X.Y. Wang, F.D. Guo, X.J. Lai, H.B. Yu, Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD, rump session of Crypto'04, E-print, 2004.

[21] A. Joux. Collisions for SHA-0, rump session of Crypto'04, 2004.

[22] S. Contini, Y. L. Yin, Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions, Advances in Cryptology – ASIACRYPT 2006, LNCS, November 15, 2006,pp.37-53.

[23] M Gorski, S Lucks, T Peyrin, Slide Attacks on Hash Functions, ASIACRYPT, Lecture Notes in Computer Science. Springer, 2008.

[24] Lai, X., Massey, J.: Hash function based on block ciphers. In: Advances in Cryptology - EUROCRYPT '92. Lecture Notes in Computer Science, vol. 658, pp. 55–70. Springer, Heidelberg (1992)

[25] Damg˚ard, I.: A design principle for hash functions. In: Advances in Cryptology - CRYPTO '89. Lecture Notes in Computer Science, vol. 435, pp. 416–427. Springer, Heidelberg (1990)

[26] Merkle, R.: One way hash functions and DES. In: Advances in Cryptology - CRYPTO '89. Lecture Notes in Computer Science, vol. 435, pp. 428–446. Springer, Heidelberg (1990)

[27] Rivest, R.: The MD4 message digest algorithm. In: Advances in Cryptology
- CRYPTO '90. Lecture Notes in Computer Science, vol. 537, pp. 303–311. Springer, Heidelberg (1991)

[28] Rivest, R.: The MD5 message-digest algorithm. Request for Comments (RFC) 1321 (April 1992)

[29] Zheng, Y., Pieprzyk, J., Seberry, J.: HAVAL - a one-way hashing algorithm with variable length of output. In: Advances in Cryptology - AUSCRYPT '92. Lecture Notes in Computer Science, vol. 718, pp. 83–104. Springer, Heidelberg (1993)

[30] Barreto, P., Rijmen, V.: The WHIRLPOOL hashing function (2003), http://www.larc.usp.br/ ~pbarreto/whirlpool.zip

[31] National Institute of Standards and Technology. Secure Hash Standard (SHS). Federal Information Processing Standards Publication 180-3 (October 2008)
nessie/deliverables/ press_release_feb27.pdf, last visited Ma

[32] Biham, E., Dunkelman, O.: A framework for iterative hash functions – HAIFA. Cryptology ePrint Archive, Report 2007/278 (2007)

[33] Rivest, R.: Abelian square-free dithering for iterated

hash functions (ECRYPT Hash Function Workshop 2005)

[34] Amiripalli, S. S., & Bobba, V. (2019). Trimet graph optimization (TGO) based methodology for scalability and survivability in wireless networks. International Journal of Advanced Trends in Computer Science and Engineering, 8(6), 3454-3460. doi:10.30534/ijatcse/2019/121862019.

[35] Amiripalli, S. S., & Bobba, V. (2019). Impact of trimet graph optimization topology on scalable networks. Journal of Intelligent & Fuzzy Systems, 36(3), 2431-2442

[36] Amiripalli, S. S., Kumar, A. K., & Tulasi, B. (2016, February). Introduction to TRIMET along with its properties and scope. In AIP Conference Proceedings (Vol. 1705, No. 1, p. 020032). AIP Publishing LLC.

[37] Coron, J., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damg˚ard revisited: How to construct a hash function. In: Advances in Cryptology - CRYPTO 2005. Lecture Notes in Computer Science, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)

[38] Andreeva, E., Preneel, B.: A three-property-secure hash function. In: Selected Areas in Cryptography 2008. Lecture Notes in Computer Science, vol. 5381, pp. 228–244. Springer, Heidelberg (2008)

[39] Bellare, M., Rogaway, P.: Collision-resistant hashing: Towards making UOWHFs practical. In: Advances in Cryptology - CRYPTO '97. Lecture Notes in Computer Science, vol. 1294, pp. 470–484. Springer, Heidelberg (1997)

[40] Merkle, R.: One way hash functions and DES. In: Advances in Cryptology - CRYPTO '89. Lecture Notes in Computer Science, vol. 435, pp. 428–446. Springer, Heidelberg (1990)

[41] Sarkar, P.: Construction of universal one-way hash functions: Tree hashing revisited. Discrete Applied Mathematics 155(16), 2174–2180 (2007)

[42] Lee, W., Chang, D., Lee, S., Sung, S., Nandi, M.: Construction of UOWHF: Two new parallel methods. IEICE Transactions 88-A(1), 49–58 (2005)

[43] Rivest, R.L., Agre, B., Bailey, D.V., Crutchfield, C., Dodis, Y., Elliott, K., Khan, F.A., Krishnamurthy, J., Lin, Y., Reyzin, L., Shen, E., Sukha, J., Sutherland, D., Tromer, E., Yin, Y.L.: The MD6 hash function (2008)

[44] Bertoni, G., Daemen, J., Peeters, M., Assche, G.: Sufficient conditions for sound tree and sequential hashing modes. Cryptology ePrint Archive, Report 2009/210 (2009)

[45] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sakura: a flexible coding for tree hashing. In: Applied Cryptography and Network Security - ACNS 2014. Lecture Notes in Computer Science, vol. 8479, pp. 217–234. Springer, Heidelberg (2014)

BIOGRAPHIES

Jalagam Lokesh Naidu,
B.Tech Final year,
Dept. of Computer Science and Engineering,
GITAM deemed to be University.

Akhil Chandra Gorakala,
B.Tech Pre-Final year,
Dept. of Computer Science and Engineering,
GITAM deemed to be University.

Shanmuk Srinivas A,
Assit. Professor,
Dept. of Computer Science and Engineering,
GITAM deemed to be University.