The Sponge Structure Modulation Application to Overcome the Security Breaches for the MD5 and SHA-1 Hash Functions

Zeyad A. Al-Odat and Samee U. Khan
Department of Electrical and Computer Engineering
North Dakota State University
Fargo, ND, USA
Email: zeyad.alodat@ndsu.edu, samee.khan@ndsu.edu

***Abstract*—This paper presents a Sponge structure modulation of the MD5 and SHA-1 hash functions. The work employs the Keccak permutation function to build the proposed scheme. The work discusses the main two security breaches that threaten the cryptography hash standards which are collision and length extension attacks. Through analyzing several examples of collided messages of both algorithms (SHA-1 and MD5), we describe the potentials to overcome the collision and length extension attacks. Moreover, a proper replacement technique to avoid such attacks is discussed in this paper.**

***Index Terms*—Length extension attack, collision attack, cryptography, hash.**

I. Introduction

Secure Hash Algorithm (SHA) is the most popular cryptography technique for message authentication and verification. The SHA functions were standardized by the National Institute of Standards and Technology (NIST). SHA standards follow different structure models to construct the compression function. The most popular hash standards follow Merckle-Damgard (MD) and Sponge structure models. Where, MD4, MD5, SHA-1, and SHA-2 standards follow the MD structure, While SHA-3 hash standard follows Sponge structure model.

MD4 developed by Rivest in 1990 [1], then it was replaced by MD5 in 1991 [2]. Both MD4 and MD5 maintain 128-bit hash output with 512-bit block size. For security issues and early signs of collision attack, MD5 was replaced by the SHA-1 in 1993 with 160-bit hash and 512-bit block size [3].

In 2001 SHA-2 was developed and standardized by the NIST as the next version of the secure hash algorithm that follows the same structure model (MD). SHA-2 has six different flavours SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256 [4]. Then in 2012, NIST announced the next SHA-3 standard Keccak, which was selected by a competition between 63 competitors through three rounds of selection. Keccak was standardized as the SHA-3 hash standard comprises six flavors, four fixed and two extensible size hashes [5].

Three challenges exist to verify the completeness of any hash standard: preimage, 2^{nd} preimage and collision resistance. Preimage resistance property means to easily obtain the hash from a given message, but difficult to extract it back from a given hash. 2^{nd} preimage resistance means that it is difficult to find two messages M1 and M2 generate the same Hash. While collision resistance property means the resistant of the probability to generate the same output hash for two messages or more, even though they are different or equal [6].

All secure hash algorithms were tested toward security properties of hash standards, especially collision resistance property. MD5 hash standard was fully exposed to collision attack in 2005 by Wang *et al.* [7]. their work was the first published work that provided a collision example of full MD5. In their work, they used the modular difference technique to construct their attack. More details will be presented in Section III.

The security analysis of the SHA-1 hash standard, against collision attack, was also explored by different publications [8], [9]. Using the concept of modular difference to construct collision path, Wang *et al.* in [8], theoretically, succeeded to find collision attack on full SHA-1. Recently, in 2017, Stevens *et al.* found the first real example of messages that collided when processed using SHA-1 compression function.

However, the secure hash algorithms (MD5 and SHA-1) are still be used by different entities, particularly the SHA-1. Therefore, the efforts of researchers and developers were employed to overcome the collision dilemma which prone systems and applications into a serious security breach.

This paper analyzes the collision and length extension attacks of the secure hash algorithms, MD5 and SHA-1. The analysis is carried out by testing several examples of collided messages that were generated by the help of ChameleonCloud which is a configurable experimental environment for large-scale cloud research [10]. This paper presents a versatile modification to the compression functions of MD5 and SHA-1 to counter the collision and length extension attacks. The modification employs the internal round functions of the Keccak hash standard. Yet, Keccak is the most secure hash standard against security breaches. The strength of Keccak comes from the sturdiness of the compression function of Keccak standard [11].

The rest of paper is organized as follows: Section 2 presents

background information related to the subject of the paper. Section 3 presents the literature review of the previous works. Section 4 shows the proposed work. results and discussions are presented in section 5. Section 6 concludes the paper.

II. Background

Before going through the details of our proposal, a brief description of secure hash algorithm standards and collision attack will be presented. In addition, we collected all notations that are used in this paper in Table. I.

TABLE I: Notations

Symbols	Meaning
IHV	Initial Hash Value
WS	Working State Variables
$\oplus, \lll$	XOR, Shift left
$[x, y]$	state matrix parameters. Take values 0,1,2,3,4
ROT	Rotate left operation
f	Permutation function
θ	Theta step of the SHA-3 compression function
ρ	Rho step of the SHA-3 compression function
π	Pi step of the SHA-3 compression function
χ	Chi step of the SHA-3 compression function
ι	Iota step of the SHA-3 compression function
b	State matrix size
r	Bit rate which equal to the message block size
c	Capacity, where $b = r + c$.

A. Brief Description of Secure Hash Algorithms

Secure hash algorithms accept a message of predefined size, then through several steps of compression function calculations, the output hash is produced. Table II shows different hash standards and their corresponding parameters.

TABLE II: SHA Family and Parameters

SHA Family				
Algorithm	**Hash Size/bit**	**Block Size/bit**	**Msg Size/bit**	**#Round**
MD5	128	512	$< 2^{64}$	64
SHA-1	160	512	$< 2^{64}$	80
SHA2				
224	224	512	$< 2^{64}$	64
256	256	512	$< 2^{64}$	64
384	384	1024	$< 2^{128}$	80
512	512	1024	$< 2^{128}$	80
512/224	224	1024	$< 2^{128}$	80
512/256	256	1024	$< 2^{128}$	80
SHA3				
224	224	1152	UL	24
256	256	1088	UL	24
384	384	832	UL	24
512	512	576	UL	24
SHAKE-128	Arbitrary	1344	UL	24
SHAKE-256	Arbitrary	1088	UL	24

For MD hash standards, the maximum message size that each algorithm accepts is depending on the block size, where the 512-bit block size accepts messages of size less than 2^{64}, while the others accept a message size of 2^{128}-bit. However, the SHA-3 (Keccak) hash standard accepts messages of unlimited (UL) size. The Table shows that SHA-3 supports the same hash lengths that SHA-2 supports beside to two extensible output hashes SHAKE-128 and SHAKE-256. But, the block sizes that SHA-3 incorporate are different from the block sizes that SHA-2 incorporated. Moreover, the number of internal rounds of SHA-3 is different from the MD hash standards rounds, this is because of the difference in the nature between the MD and Sponge structures. The Only hash standard that provides a variable hash output is the SHA-3[1].

All hash standards perform pre-processing steps before starting the compression function calculations. These pre-processing steps summarized as follows:

- Message padding. In this phase, the message is padded with a sufficient number of zeros to make the message size divisible by the message's block-size. However, in the case of Sponge structure, the padding process is carried out according to the state size. where the state size is equal 1600-bit for the 24 rounds.
- Message divide. After the message padding phase, the message is divided into equal size blocks, each of size equal to block size. However, in the case of SHA-3, the message is divided according to the state size (b).
- Compression function calculation. The message's blocks are processed sequentially, one at a time, using the round compression functions according to the used hash. Each block is processed a number of times equal to the number of rounds according to the desired hash function. The output of each block is fed as an input to the second block, after finishing all blocks.
- Output hash generation. After processing all message's blocks, the output hash is taken the concatenation of part or all of the output of the last block calculation round. However, the SHA-3 provides diversity to select the output hash, where the output of each block can be squeezed more time during the same round.

For more details about secure hash algorithms and their compression functions, the reader is referred to [5].

B. Sponge Structure Model

In Sponge structure model, the data are absorbed in and squeezed out using permutation function. Each block is processed as state matrix and divided into two parts, the bit rate part which denoted by r, and the capacity part which denoted by c. The sum of the two values $r + c = b$, where r is equal to the block size. The state matrix is initialized with zeros then the message blocks (p_i) are processed sequentially as shown in Fig. 1. After processing all steps the output hash is taken from any output z_i, according to the desired hash.

III. Related Work

Because the MD structure model was prone to security breaches, the researches try to protect hash standards that follow MD structure, particularly MD5 and SHA-1. For instance, collision attack and length extension attack are considered as main security issues that threaten the MD hash standards (MD4, MD5, SHA-1, and SHA-2).

[1]Whenever we mention SHA-3 in this paper, it implicitly means Keccak.

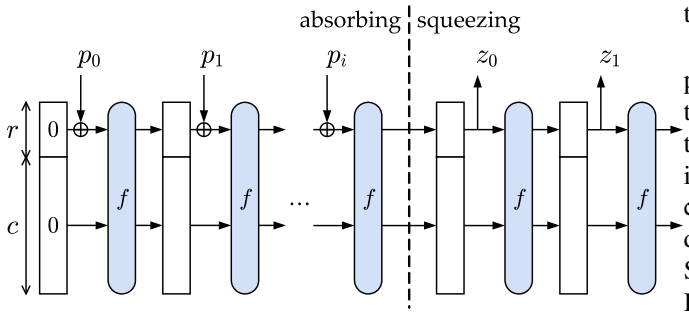


Fig. 1: Sponge structure model, where the data are absorbed in and squeezed out to generate the desired output hash.

MD5 and SHA-1 hash standards were exposed to collision attack by Wang *et al.* [8], [12]. They presented a novel technique to construct the first collision attack of MD5 and SHA-1. They used a modular difference technique to build a disturbance vector that leads to a path for collision. Their work was supported with examples of two blocks messages that have the same hash value. Then in 2017, Stevens *et al.* presented the first real example of collided messages that have the same SHA-1 value [9]. Their work built upon the idea of the differential attack that Wang *et al.* came up with. After the announcement of the first real example of collision attack, many entities dropped SHA-1 from their systems, and they tried to find the best replacement to the SHA-1. However, The SHA-1 and MD5 are still be used by some entities.

To support the entities that still using the weak hash standards, many researchers suggest different replacements and modifications to overcome the collision issue. Two approaches were adopted by the researches. The first approach detects the possibility of a collision before it takes place. While the second approach improves the weak hash standards. The first approach was presented by different publications [6], [13], [14]. Stevens *et al.* developed a novel technique to detect the possibility of a collision attack toward the SHA-1. They got benefited from different disturbance vectors that are used to build a collision attack. The proposed technique was able to detect the occurrence of collision among the used disturbance vectors. The authors used the top 32 vectors that have a high possibility to establish a collision when they are employed [13]. The authors in [14] used the technique of unavoidable bit conditions to speed up the detection of collision attack. They built upon the work of Stevens *et al.*, and they were able to get better speed than the previous technique. Both of the aforesaid techniques build a sibling message from a given message and carry out the calculations for both massages for all possibilities. However Alodat *et al.* in [6] proposed a new technique to detect the collision attack using the idea of two block collision. Authors claimed that any collision attack built upon two block collision attack, and if we are able to detect the collision of the first two blocks then we will save time and efforts. Then to speed up the detection process the authors constructed their design without any need to build the sibling of a given message.

On another hand, the second approach was adopted by many publications [15]–[18]. Instead of detecting the collision attack the researchers of the second approach worked on improving the weak hash functions. Hakim *et al.* in [15] proposed improvement for the MD hash standards. They worked on combining the MD5 and SHA-256 hash functions in one design. The authors changed the expansion equation of the SHA-256 to add more randomness to the proposed scheme. In their work, the structure of MD5 standards was used twice and fed back as an input to the SHA-256 function. Rogel *et al.* proposed a modified version of the SHA-1 by changing adding multiplexer box in the internal calculation of the intermediate hash value [17]. Moreover, Authors in [18] presented a new architecture that used the Sponge structure model to construct a secure hash standard. The proposed design (Titanium) was able to produce a 576-bit hash, which considered as a new hash length that the other hash standards do not have.

The other threat that makes the MD structure model vulnerable is the length extension attacks. This attack exposed many hash standards including MD5, SHA-1, SHA-256, and SHA-512. This kind of attack threatens the Message Authentication Code (MAC) with the hash standards by generating a valid hash value without any need to know the secret code [19]. Some published and online publications proved the inability of the MD structure model against length extension attack [20], [21]. Where the attacker intercepts the sender message and makes the proper modifications without the receiver notice of that modifications even though using the hash function.

In this paper, we used the Sponge structure model to construct the MD5 and SHA-1 hash functions. Moreover, the internal structure of the Keccak hash function will be employed in our scheme.

IV. Proposed Methodology

We propose a Sponge structure implementation for the MD5 and SHA-1 hash functions. A pre-processing is accomplished before starting with the Sponge operations (absorb and squeeze). The message is padded first using $10*1$ technique, where 1 is added at the end of the message and followed by a sufficient number of zeros then ended with 1. The final message size needs to be multiple of block size P_i. Afterward, the message is divided into equal size blocks (P_i) each of r bits [5]. Notice that, the padding technique here is different from the technique used in the MD structure.

According to the Sponge structure, the message goes into two phases, the absorb and squeeze phases. In the absorbing phase, the state of b-bit is initialized with 0's, where each state is represented by two values, bit-rate (r) and capacity (c), where, r is the block size P_i, and c complete the state size to 1600-bit. The XOR operations are carried out for the message blocks P_i with r bits of the state. A permutation function f is applied to get next state value [11]. For our proposal, we set $r = 576$ and $c = 1024$. In the squeezing phase, the output hash (z) is taken from the least significant bits of z_0 according

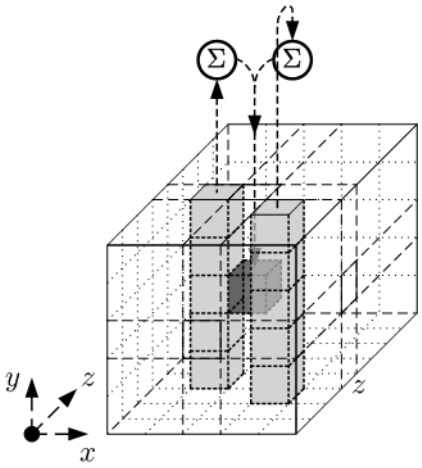


Fig. 2: Theta step

to the corresponding hash size (128, 160). Where 128 is the MD5 digest and 160 is the SHA-1 digest.

The permutation function consists of five steps represented by Greek symbols, Theta (θ), Rho (ρ), Pi (π), Chi (χ), and Iota (ι).

$Theta$ (θ) step.

This step operates on a 3D-Array ($5 \times 5 \times 64$) as shown in Fig. 2. A single 5×5 array is a slice, where the 1D-array, toward z axis, of 64 bits is a lane. Theta step manipulates the state array according to 1, 2, and 3. Where $C[x]$ and $D[x]$ represent lanes and $A[x, y]$ represents slice. Theta computes the parity of each column, then combines them with the XOR operator.

$$C[x] = A[x, 0] \oplus A[x, 1] \oplus A[x, 2] \oplus A[x, 3] \oplus A[x, 4], \quad (1)$$

$$D[x] = C[x - 1] \oplus ROT(C[X + 1], 1), \quad (2)$$

$$A[x, y] = A[x, y] \oplus D[x], \quad (3)$$

where $x = 0, 1, 2, 3, 4$ for all cases.

Rho (ρ) step

In this step, one element (lane) of the state matrix $A[x, y]$ is rotated by i-bit, as seen in Eq. 4. The rotation offset value denoted by $r[x, y]$ is a constant value assigned according to Table III.

$$step(\rho) = ROT(A[x, y], r[x, y]) \quad (4)$$

TABLE III: Values of constants r[x,y] in the Rho step

	$x = 3$	$x = 4$	$x = 0$	$x = 1$	$x = 2$
$y = 2$	25	39	3	10	13
$y = 1$	55	20	36	44	6
$y = 0$	28	27	0	1	62
$y = 4$	56	14	18	2	61
$y = 3$	21	8	41	54	15

TABLE IV: Iota Round Constants

RC[i] starting from RC[0] to RC[23] from left to right top to down.	
$0x0000000000000001$	$0x000000008000808B$
$0x0000000000008082$	$0x800000000000008B$
$0x800000000000808A$	$0x8000000000008089$
$0x8000000080008000$	$0x8000000000008003$
$0x000000000000808B$	$0x8000000000008002$
$0x0000000080000001$	$0x8000000000000080$
$0x8000000080008081$	$0x000000000000800A$
$0x8000000000008009$	$0x800000008000000A$
$0x000000000000008A$	$0x8000000080008081$
$0x0000000000000088$	$0x8000000000008080$
$0x0000000080008009$	$0x0000000080000001$
$0x000000008000000A$	$0x8000000080008008$

Pi (π) step

Pi step is a complement to rho step, where it takes the rotated lanes from rho step and put them in different positions in the array matrix ($B[x, y]$), without modifying any value. It only permutes the matrix according to 5,

$$B[y, 2x + 3y] = ROT(A[x, y], r[x, y]), \quad (5)$$

where $x, y = 0, 1, 2, 3, 4$.

Chi (χ) step

In this step the B matrix, that was generated from the previous step, is manipulated and put the result back in array matrix A according to Eq. 6.

$$A[x, y] = B[x, y] \oplus ((\bar{B}[x + 1, y]) \wedge B[x + 2, y]), \quad (6)$$

where $x, y = 0, 1, 2, 3, 4$, $\wedge$ bit-wise AND operation, and $\bar{B}[]$ is the bit-wise NEGATION of lane.

$Iota$ (ι) step

Iota step adds the round constant $RC[i]$ to the state matrix A at location $A[0, 0]$ according to 7, where each round has a distinct 64-bit round constant.

$$A[0, 0] = A[0, 0] \oplus RC[i], \quad (7)$$

where $RC[i]$ is a round constant.

The above permutation steps (θ, ρ, π, χ, and ι) are carried out for 24 rounds. Each round has a distinct round constant $RC[i]$ and their values are assigned according to Table IV.

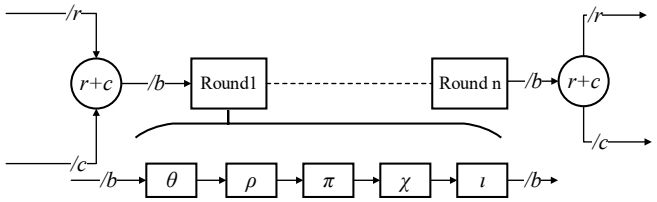


Fig. 3: Internal Round of the compression function.

Fig 3 shows the integration between the five permutation steps and the compression components. The input state consists of r and c values which together conform the $b = 1600$ value. The

TABLE V: Two examples of PDF files that have the same SHA-1 value.

		SHA-1
SHA-1		
Fig.4-a	SHA-1:	eda6de91-e04bcf93-342c63ee-c7de45b4-720e4a9e
	proposed:	e4667d12-70318267-b8612f69-1100285e-f5fbbd29
Fig.4-b	SHA-1:	eda6de91-e04bcf93-342c63ee-c7de45b4-720e4a9e
	proposed:	a6f02b7a-b36964ae-664d5676-70eb1492-f93fcaa4
MD5		
Fig.4-c	MD5:	39885429-fe2761b1-0b263b34-8dfdd1b2
	Proposed:	75a40711-19b15b99-6eaca068-4d57341b
Fig.4-d	MD5:	39885429-fe2761b1-0b263b34-8dfdd1b2
	Proposed:	288eb391-cb79b77e-c97a0ea9-f3c78274

input message's block p_i is processed using the permutation functions for 24 times. Afterward, the output b value is fed as IHV to the next state calculations.

V. Results and Discussions

The Sponge structure modulation helps to protect the MD5 and SHA-1 hash standards. Two main security breaches are considered for analysis, collision and length-extension attacks. The experiments were tested using a configurable experimental environment for large-scale cloud research, Chameleon [22]. Where all message samples and testing results were generated on the Chameleon environment. Readers can access all codes, datasets, and other artifacts by referring to [23].

A. Collision Attack

To test the validity of the proposed scheme, we prepared real examples of collided messages that produce the same hash value. Fig. 4 shows two examples of collided messages. Fig. 4-a and Fig. 4-b are two bank checks that have different information and produce the same SHA-1 value. While, Fig. 4-c and Fig. 4-d are two different text with the same MD5 hash. Table V shows the corresponding hash values for each figure. The SHA-1 value for Fig. 4-a and Fig. 4-b are equal, but when we applied our proposal we were able to produce different hash values. Moreover, the MD5 hash values for Fig. 4-c and Fig. 4-d are the same, but with our proposal the MD5 values are different.

B. Length Extension Attack

Length extension attack not only targeted MD5 and SHA-1 hash functions but all MD structure hashes. One of the main contributions of the proposed scheme is to protect the MD5 and SHA-1 against length extension attack by modulates both functions using Sponge structure.

The attacker uses the padding technique of MD structure models to produce a successful length extension attack. Where the MD-structure padding ends with the message length, the attacker starts from where the padding has ended and continues to add more zeros and the modified information then append the new hash value. Because the receiver deals with payload and hash, he is unable to detect the alteration, because the received information matches the received hash.

TABLE VI: Length extension attack bank transaction example

	account_from	account_to	amount	append
Digest	123456	112233	100	5
MD5_1	6036708eba0d11f6ef52ad44e8b74d5b			
MD5_2	8c792805248678cfd72c2a99298748d1			
MD5_3	8c792805248678cfd72c2a99298748d1			
SHA1_1	a65181853e5a1b94b4c7a84b7fc561bdcb9b75e8			
SHA1_2	281385ea5e434561b439dbe4e2417d83fac16833			
SHA2_3	281385ea5e434561b439dbe4e2417d83fac16833			

Table VI shows example of money transactions between client and server. The hash value has three phases: the client hash phase which represented by MD5_1 and SHA1_1, the after-attack hash phase which represented by MD5_2 and SHA1_2, and the server hash phase which represented by MD5_3 and SHA1_3. The hash value in the client side is computed by hashing the concatenation of `account_from` $\|$ `account_to` $\|$ `amount` values. The attacker intercepts the transaction and performs the length extension attack by appending 5 to the transaction to make the amount equal to 105. The new hash is appended to the modified message and resent to the server. At the server side, the received message is hashed and compared with the new hash value. Then the transaction is approved by the server and the process completed.

However, this kind of attack is weak against of Sponge structure model. The message size is not appended to the end of the message in the padding phase, so that, the attacker has no clue about the message size or the amount of extension needed. Our design is strong against length extension attack because we used the Sponge structure. Moreover, different padding technique in the prepossessing phase was carried out.

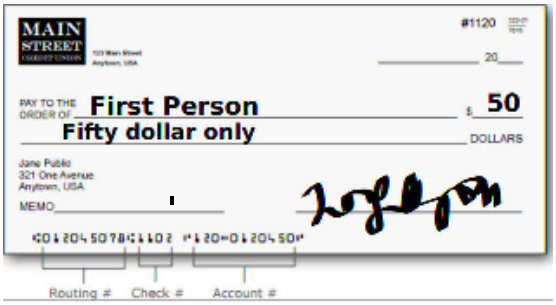
VI. Conclusions and Future Work

In this paper, a Sponge structure modulation for the MD5 and SHA-1 is presented. The proposed design helps to solve the weaknesses of the MD5 and SHA-1 against security breaches. We investigated our proposal toward collision and length extension attacks. The results showed that the proposed design is resistant to the aforesaid attacks. The strength of our design comes from the strength of the Sponge structure, where the internal permutation function manipulates the data many times with five different steps $(\theta, \rho, \pi, \chi, \iota)$.

The Sponge structure model can be extended to include other hash or cryptography functions because Sponge structure proved its strength against many security bugs that threaten cryptography functions.

Acknowledgments

The work of Samee U. Khan was supported by (while serving at) the National Science Foundation. Any opinion, findings, and conclusions or recommendations expressed in



(a) Original

(b) Forged

```
4b48 414e f37e 2764 2643 5c93 fb5c 0341  KHAN.~'d&C\..\.A
c362 cdc0 849c 7796 0753 5858 ebc2 e4ba  .b....w..SXX....
1852 e7e5 42b2 c0f5 84ec 7890 bd58 f540  .R..B.....x..X.@
6095 2f99 9a52 bbe1 a9fa 1c5a d678 eaa0  `./..R.....Z.x..
4c20 f77f d070 dacd 0f42 e71f 3dc1 7232  L ...p...B..=.r2
e618 51fa 2247 2c56 a1ec 2e3f 4a80 c914  ..Q."G,V...?J...
416b 8cf0 a78c bb56 8823 024d d577 59cc  Ak.....V.#.M.wY.
01c4 8d10 f142 1387 c26e c4f4 d673 4c02  .....B...n...sL.
```

(c) Text1

```
4b48 41ce f37e 2764 2643 5c93 fb5c 0341  KHA..~'d&C\..\.A
c362 cdc0 849c 7796 0753 58d8 ebc2 e4ba  .b....w..SX.....
1852 e7e5 42b2 c0f5 84ec 7890 bd58 f540  .R..B.....x..X.@
6095 2f99 9a52 bbe9 a9fa 1c5a d678 eaa0  `./..R.....Z.x..
4c20 f7ff d070 dacd 0f42 e71f 3dc1 7232  L ...p...B..=.r2
e618 51fa 2247 2c56 a1ec 2ebf 4a80 c914  ..Q."G,V....J...
416b 8cf0 a78c bb56 8823 024d d577 59cc  Ak.....V.#.M.wY.
01c4 8d10 f142 137f c26e c4f4 d673 4c02  .....B...n...sL.
```

(d) Text2

Fig. 4: Two examples of collided real examples. (a) and (b) belong to SHA-1 collision. (c) and (d) belong to MD5 collision, where to the left of each text is the corresponding hexadecimal equivalent.

this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation.

REFERENCES

[1] R. L. Rivest, "The md4 message digest algorithm," in *Advances in Cryptology-CRYPTO' 90*, A. J. Menezes and S. A. Vanstone, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 303–311.

[2] R. Rivest, "Rfc 1321: The md5 message-digest algorithm," Internet Activities Board, Tech. Rep., 1992.

[3] P. FIPS, "180-1. secure hash standard," *National Institute of Standards and Technology*, vol. 17, p. 45, 1995.

[4] F. PUB, "Secure hash standard (shs)," *FIPS PUB 180*, vol. 4, 2012.

[5] N. D. F. PUB and P. FIPS, "202," *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*, 2014.

[6] Z. Al-Odat, M. Ali, and S. U. Khan, "Mitigation and improving sha-1 standard using collision detection approach," in *2018 International Conference on Frontiers of Information Technology (FIT)*. IEEE, 2018, pp. 333–338.

[7] X. Wang and H. Yu, "How to break md5 and other hash functions," in *Advances in Cryptology – EUROCRYPT 2005*, R. Cramer, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 19–35.

[8] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full sha-1," in *Annual international cryptology conference*. Springer, 2005, pp. 17–36.

[9] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov, "The first collision for full sha-1," in *Annual International Cryptology Conference*. Springer, 2017, pp. 570–596.

[10] "Home | Chameleon," Feb 2019, [Online; accessed 8. Feb. 2019]. [Online]. Available: https://www.chameleoncloud.org

[11] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "The keccak sha-3 submission," *Submission to NIST (Round 3)*, vol. 6, no. 7, p. 16, 2011.

[12] X. Wang and H. Yu, "How to break md5 and other hash functions," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2005, pp. 19–35.

[13] M. Stevens, "Counter-cryptanalysis," in *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 129–146.

[14] M. Stevens and D. Shumow, "Speeding up detection of sha-1 collision attacks using unavoidable attack conditions." *IACR Cryptology ePrint Archive*, vol. 2017, p. 173, 2017.

[15] S. Hakim and M. Fouad, "Improving data integrity in communication systems by designing a new security hash algorithm," *Journal of Information Sciences and Computing Technologies*, vol. 6, no. 2, pp. 638–647, 2017.

[16] M. Bellare, J. Jaeger, and J. Len, "Better than advertised: Improved collision-resistance guarantees for md-based hash functions," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 891–906.

[17] R. P. M. Rogel L. Quilala, Ariel M.Sison, "Modified sha-1 algorithm," in *Indonesian Journal of Electrical Engineering and Computer Science*. ijeecs.v11, 2018, pp. 1027–1034.

[18] M. A. AlAhmad, "Design of a new cryptographic hash function–titanium," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 10, no. 2, pp. 827–832, 2018.

[19] "Everything you need to know about hash length extension attacks » SkullSecurity," Feb 2019, [accessed 10. Feb. 2019]. [Online]. Available: https://blog.skullsecurity.org/2012/everything-you-need-to-know-about-hash-length-extension-attacks

[20] D. Nuñez, I. Agudo, and J. Lopez, "Attacks to a proxy-mediated key agreement protocol based on symmetric encryption." *IACR Cryptology ePrint Archive*, vol. 2016, p. 1081, 2016.

[21] "MD5 Length Extension Attack Revisited | Vũ's Inner Peace," Feb 2019, [accessed 10. Feb. 2019]. [Online]. Available: https://web.archive.org/web/20141029080820/http://vudang.com/2012/03/md5-length-extension-attack

[22] K. Keahey, P. Riteau, D. Stanzione, T. Cockerill, J. Mambretti, P. Rad, and P. Ruth, "Chameleon: a scalable production testbed for computer science research," in *Contemporary High Performance Computing: From Petascale toward Exascale*, 1st ed., ser. Chapman & Hall/CRC Computational Science, J. Vetter, Ed. Boca Raton, FL: CRC Press, 2018, vol. 3, ch. 5.

[23] "zeyadodat/sponge-stucture-modulation," Feb 2019, [accessed 12. Feb. 2019]. [Online]. Available: https://github.com/zeyadodat/sponge-stucture-modulation