Collision Attack on SHA-1

Danish Malik
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
malik083@morris.umn.edu

ABSTRACT

The purpose of this research is to examine the first successful in practice identical-prefix collision attack on SHA-1. In specific, this paper discusses the structure of the files that were used in relation to the compression function of SHA-1 to obtain a collision. Additionally, the methods used to perform the collision are analyzed such as construction of a linear and non-linear differential path, as well as the procedure for selecting disturbance vectors using joint local-collision analysis. The paper also provides an understanding of the computational power that was used to conduct the attack.

Keywords

SHA-1, Collision Attack, Cryptography, Cyber-security

1. INTRODUCTION

A hash function H is an algorithm that computes for any input data x of arbitrary bit-length, a fixed number of bits known as the hash value z such that $H(x) = z$. Hash functions find common use for rapid data search in collaboration with a data structure known as a hash table. Hash functions accelerate database queries by finding duplicated data. Furthermore, hash functions are commonly used in cryptography. Cryptographic hash functions belong to a special class of hash functions as they exhibit several properties making it ideal in the world of cyber security. The first property of a cryptographic hash function is that it must be deterministic. This implies that irrespective of the number of times a particular input is passed through a hash function, the returned value must be the same. The deterministic property of cryptographic hash functions is critical since it would be impossible to keep track of the input if it produces different hashes for any number of computations. The second property of cryptographic hash functions is quick computation, which means they must be capable of returning hash values quickly to maintain an efficient system. The third property of hash functions is known as Pre-Image Resistance, which implies it must be computationally infeasible to reverse a hash value z to its corresponding value x such that $H(x) = z$ (one-way function). Additionally, this property is reinforced as cryptographic hash functions produce hash values with large number of bits. For instance, SHA-1 computes hash values of 160 bits. As a result, using the traditional brute-force attack, an attacker must compute the hash of $2^{160} - 1$ inputs to successfully reverse a given hash value. The fourth property of cryptographic hash functions is known as the avalanche effect. According to this property, a slight change in input causes a significant change in the output hash value. The fifth and most important property of hash function is collision resistance. This property implies that a cryptographic hash function must compute distinct hash values for unique inputs. A *collision* occurs when two distinct inputs x and y map to the same hash value such that $H(x) = H(y)$. The major purpose of cryptographic hash functions is based on the assumption that, in practice, it should be nearly impossible to find collisions when computing the hash values of distinctive input data. Thus, such properties make cryptographic hash functions find common use in the world of cyber security for maintaining the integrity of data. They are used in several authentication applications such as message authentication code, password hashing and digital signature schemes.

The MD-SHA family is the most popular family of hash functions up to date. The structure of the hash functions belonging to the MD-SHA family is based upon the iterative Merkle-Damgard construction. Additionally, its compression function is built upon a block cipher in Davies-Meyer Mode. In 1990, R. Rivest introduced MD4 [5] to the world of cyber security, marking the start of the MD-SHA family. Just two years later, in 1992, MD4 was replaced with MD5 [4] due to serious security flaws. MD5 found common use in the software world for several years despite the weakness of its foundational compression function. In 1993, the National Security Agency created and published SHA-0 as a US standard. However, SHA-0 was quickly replaced with SHA-1 in 1995. The only difference between SHA-1 and SHA-0 is a 1-bit rotation that occurs during message expansion of the input message. In 2004, Stevens et al. [8] demonstrated an extremely powerful attack, known as a chosen-prefix attack, against MD5. This groundbreaking development eventually led to the widespread use of false digital signatures that allowed malicious users to impersonate any website especially those of digital commerce even under the protection of the HTTPS protocol [9]. After the deprecation of MD5, SHA-1 has been consistently subject to several attacks primarily during the last ten years. Additionally, several teams worked on generating collisions for reduced versions of SHA-1 which essentially assumes lesser rounds within its compression algorithm. The very first theoretical collision attack on SHA-

UMM CSci Senior Seminar Conference, April 2018 Morris, MN.

Stage t	Round i	Constant k_t	Function f_t
1	0...19	$k_1 = 5A827999$	$f_1\ (B,\ C,\ D) = (B \wedge C) \vee (\sim B \wedge D)$
2	20...39	$k_2 = 6ED9EBA1$	$f_2\ (B,\ C,\ D) = B \bigoplus C \bigoplus D$
3	40...59	$k_3 = 8F1BBCDC$	$f_3\ (B,\ C,\ D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
4	60...79	$k_4 = CA62C1D6$	$f_4\ (B,\ C,\ D) = B \bigoplus C \bigoplus D$

Table 1: Stage functions and constants for SHA-1

1 was presented in 2005 by Wang et al. [11], which was ground-breaking since it displayed run-time, although still exponential it was significantly faster than the traditional brute-force attack. Chabaud and Joux [1] presented the first theoretical attack on SHA-0 which had a complexity of 2^{61}. However, after the integration of neutral bits by Biham and Chen [2], full collisions on SHA-0 reduced to a complexity of 2^{51}. The first theoretical collision attack on SHA-1 had a complexity of 2^{69} after Wang et al introduced modular addition differentials as well as message modification techniques. In 2013, Stevens [7] presented the currently best known collision attack on full SHA-1 with an estimated cost of 2^{61} calls to the SHA-1 compression function. However, a publicly demonstrated collision attack was still out of reach. Finally, in 2017, a team led by Elie Bursztein from Google and Marc Stevens from the Centrum Wiskunde Informatica (CWI) collaborated over a period of two years to perform the first successful in practice collision attack [10] on the full (80 rounds) of SHA-1 with a complexity of $2^{63.1}$.

2. BACKGROUND

2.1 SHA-1

SHA-1 takes an input message X of arbitrary bit-length l and yields an output of 160 bits. First, the input X is padded to obtain a multiple of 512 bits. Before the padded message X_{padded} can be fed into the compression function c, which is the heart of SHA-1's algorithm, it is divided into n 512-bit message blocks $B_1, B_2, \ldots, B_n$. Furthermore, each message block B_j can be sub-divided into exactly sixteen 32-bit segments known as *message words* as shown below.

$B_j = W_0,\ W_1,\ ...,\ W_{15}$

These 16 message words from each 512-bit message block B_j are expanded to a set of 80 message words by the *message schedule* of SHA-1 according to the recursive equation below.

$$W_i = \begin{cases} W_i & 0 \leq i \leq 15 \\ (W_{i-16} \oplus W_{i-14} \oplus W_{i-8} \oplus W_{i-3}) \lll_1 & 16 \leq i \leq 79, \end{cases}$$

where $X \lll_n$ specifies a circular left shift of the word X by n bit positions and $\oplus$ denotes bit-wise XOR.

As a result, using all 512-bit message blocks $B_1, B_2, \ldots, B_n$, the message schedule prepares n sets of 80 message words with length 32-bit each for the compression function of SHA-1. Additionally, the compression function contains a 160-bit internal state known as the *chaining value* CV_i that is initialized to a predefined value $CV_0 = IV$, where IV denotes the initialization vector, before the computation of the first set of 80 message words (derived from B_1). Every set of 80 message words is processed in 80 rounds. Each round i of the compression function uses a message word W_i and updates the chaining value such that $CV_{i+1} = c(CV_i, W_i)$. The chaining value CV_{80} acquired after the computation of the first set of 80 message words (derived from B_1) becomes the initial chaining value CV_0 prior to the computation of the second set of 80 message words (derived from B_2). The chaining value CV_{80} obtained after processing the second set of 80 message words (derived from B_2) becomes the initial chaining value CV_0 prior to the computation of the third set of 80 message words (derived from B_3). The initial chaining value CV_0 for subsequent sets of 80 message words is defined similarly. The final chaining value CV_{80} obtained after processing the final set of 80 message words (derived from B_n) is output as the hash value z.

At every round i, the chaining value is parsed as five 32-bit strings known as *state words* denoted by A, B, C, D and E as shown below.

$CV_i = (A_i \ ||\ B_i \ ||\ C_i \ ||\ D_i \ ||\ E_i)$

where $||$ denotes concatenation.

The compression function c processes the set of 80 message words derived from each message block B_j in four stages. In each stage, the compression function uses different internal functions f_t and constants k_t, where t corresponds to the stage of the compression function. The internal function f_t uses bit-wise Boolean operations such as AND ($\wedge$), OR ($\vee$), NOT ($\sim$) and XOR ($\bigoplus$) as shown in Table 1.

At every round i of the compression function, the state words that form the chaining value are updated using the message word W_{i-1} and the state words (that form the chaining value CV_{i-1}) from the previous round. For each round $i = 1...80$, the state words that form the chaining value are updated as shown below.

$$\begin{aligned} A_i &= (A_{i-1} \lll_5) + f_t(B_{i-1}, C_{i-1}, D_{i-1}) + E_{i-1} + W_{i-1} + k_t \\ B_i &= A_{i-1} \\ C_i &= B_{i-1} \lll_{30} \\ D_i &= C_{i-1} \\ E_i &= D_{i-1} \end{aligned}$$

where $+$ denotes addition modulo 2^{32}

3. ATTACK

3.1 Construction of the hash input

The attack was initiated by constructing two *near-collision* PDF images x and y, where x denotes the first PDF image and y denotes the second PDF image. Near collision refers to the property that the content of the PDF images x and y have some similarity but, as a whole, are not entirely the same. Each image is divided into five 512-bit message blocks $B_1, B_2, \ldots, B_5$ that are contained in three different segments.

```
25 50 44 46 2d 31 2e 33 0a 25 e2 e3 cf d3 0a 0a   %PDF-1.3.%......
0a 31 20 30 20 6f 62 6a 0a 3c 3c 2f 57 69 64 74   .1 0 obj.<</Widt
68 20 32 20 30 20 52 2f 48 65 69 67 68 74 20 33   h 2 0 R/Height 3
20 30 20 52 2f 54 79 70 65 20 34 20 30 20 52 2f    0 R/Type 4 0 R/
53 75 62 74 79 70 65 20 35 20 30 20 52 2f 46 69   Subtype 5 0 R/Fi
6c 74 65 72 20 36 20 30 20 52 2f 43 6f 6c 6f 72   lter 6 0 R/Color
53 70 61 63 65 20 37 20 30 20 52 2f 4c 65 6e 67   Space 7 0 R/Leng
74 68 20 38 20 30 20 52 2f 42 69 74 73 50 65 72   th 8 0 R/BitsPer
43 6f 6d 70 6f 6e 65 6e 74 20 38 3e 3e 0a 73 74   Component 8>>.st
72 65 61 6d 0a ff d8 ff fe 00 24 53 48 41 2d 31   ream......$SHA-1
20 69 73 20 64 65 61 64 21 21 21 21 21 85 2f ec    is dead!!!!!./.
09 23 39 75 9c 39 b1 a1 c6 3c 4c 97 e1 ff fe 01   .#9u.9...<L.....
```

Figure 1: Identical prefix in hexadecimal [10].

The first segment, which is known as the *prefix* P, contains the first three 512-bit message blocks B_1, B_2 and B_3. Furthermore, the prefix P is identical in both images x and y.

$$P^{(x)} = (B_1||B_2||B_3)^{(x)} = (B_1||B_2||B_3)^{(y)} = P^{(y)}$$

where $P^{(x)}$ and $P^{(y)}$ represent the Prefix P of images x and y

The second segment of both files is known as the *first near-collision block* M_1. Moreover, this segment contains the fourth 512-bit message block B_4. The content of this segment is different for both x and y. Hence,

$$M_1^{(x)} = (B_4)^{(x)} \neq (B_4)^{(y)} = M_1^{(y)}$$

The third and final segment of x and y is known as the *second near-collision block* M_2 which contains the fifth and final 512-bit message block B_5. However, the content of this segment also differs for x and y.

$$M_2^{(x)} = (B_5)^{(x)} \neq (B_5)^{(y)} = M_2^{(y)}$$

The second and third segments of both PDF images, x and y, have very minimal differences from each other as underlined through red and blue hexadecimal symbols in figure 3 which is why they are referred to as near-collision block pairs. This type of specific collision attack, where the input is structured in the configuration described, is known as a *chosen-prefix attack*.

$$\text{SHA-1}(P||M_1^{(x)}||M_2^{(x)}) = \text{SHA-1}(P||M_1^{(y)}||M_2^{(y)})$$

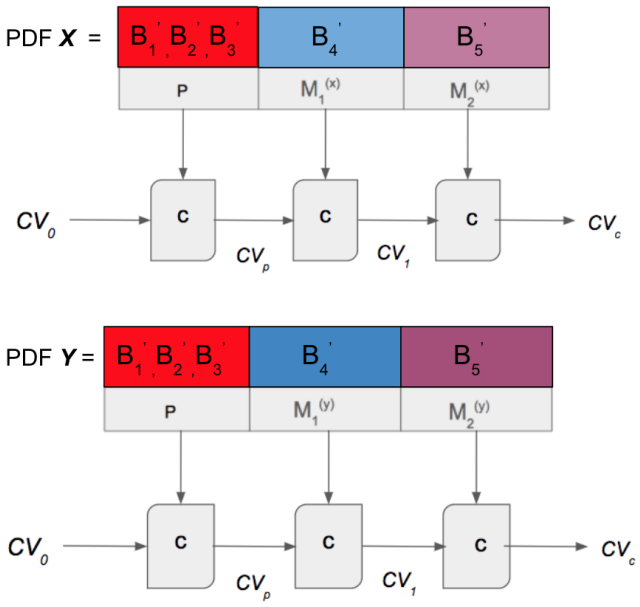

Figure 2: Attack overview.

3.2 Overview of Collision Attack

For the PDF images x and y to go through the compression function c, each message block B_j of its five 512-bit message blocks $B_1, \ldots, B_5$ must be expanded by the message schedule. Recall, every message block B_j can be subdivided into 16 message words of 32-bit length. The message schedule expands the 16 message words $W_0, \ldots W_{15}$ into a set of 80 message words $W_0, \ldots W_{79}$. As a result, the message schedule prepares five *expanded message blocks* $B_1^{'}$,..., $B_5^{'}$ for each PDF image x and y, where every expanded message block $B_j^{'}$ contains 80 message words.

The chaining value CV_{80} obtained at the eightieth round after processing the final message word W_{80} of an extended message block $B_j^{'}$ is used as the initial chaining value before the compression function processes the first message word W_0 of the next expanded message block $B_{j+1}^{'}$. After the computation of the final message word W_{79} of the final expanded message block $B_5^{'}$, the chaining value obtained is output as the hash value z.

As shown in figure 2, the five expanded message blocks of x are represented in orange whereas the five expanded message blocks of y are represented in green. Due to the predefined initialization vector, the compression function c starts with identical chaining values for both computations of x and y. Thus, prior to the start of the first round before the computation of the first segment P for both x and y.

$$CV_0^{(x)} = CV_0^{(y)}$$

where $CV_0^{(x)}$ and $CV_0^{(y)}$ represent the initial chaining value of images x and y

After processing all three blocks $B_1^{'}$, $B_2^{'}$ and $B_3^{'}$ that are contained in P, represented by the red blocks in Figure 2, the chaining value obtained is identical for both computa-

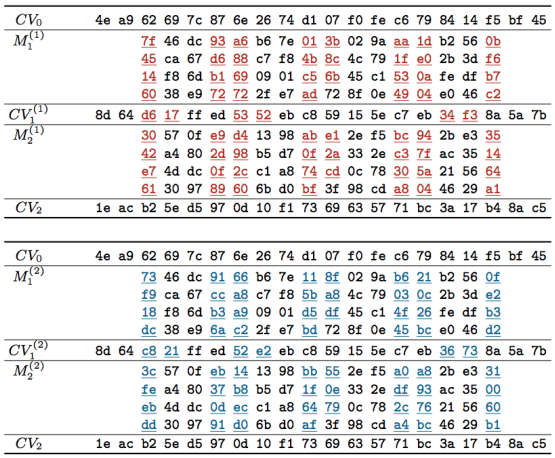

Figure 3: Near-collision block pairs in hexadecimal [10].

tions of x and y since P is identical in both images. Thus,

$$c(CV_0^{(x)}, P) = CV_P = c(CV_0^{(y)}, P)$$

Next, as the hash computation of x and y continue, the fourth expanded message block $B_4^{'}$, which resides in the first near-collision block pair $M_1^{(x)}$, represented by light blue in Figure 2, and $M_1^{(y)}$, represented by dark blue in Figure 2, are input through the compression function c. The fourth expanded message block $B_4^{'}$ starts with updating the identical chaining value CV_P which was obtained after processing the third expanded message block $B_3^{'}$. The chaining value obtained after the compression function processes $B_4^{'}$ is different for both computations of x and y since the content for $B_4^{'}$ is different in both images as shown through red and blue hexadecimal symbols in Figure 3. Thus,

$$c(CV_P^{(x)}, M_1^{(x)}) = CV_1^{(x)} \neq CV_1^{(y)} = c(CV_P^{(y)}, M_1^{(y)})$$

Finally, the last expanded message block $B_5^{'}$ which resides in the second near collision block pair $M_2^{(x)}$, displayed as light magenta in Figure 2, and $M_2^{(y)}$, displayed as dark magenta in Figure 2, are input through the compression function. The fifth and final expanded message block $B_5^{'}$ starts by updating the conflicting chaining value CV_1 that was acquired after the computation of the fourth expanded message block $B_4^{'}$. As shown in Figure 3, the bit differences in the construction of $B_5^{'}$ in each image update the chaining value such that after processing the final message word W_{79} of the block, the chaining value CV_C obtained for both computations of x and y are identical. This is the final collision that occurs during the attack since all five expanded message blocks that reside in different segments of the images have been processed by the compression function. Additionally, this identical chaining value CV_1 obtained is output as the hash value z. Thus,

$$c(CV_1^{(x)}, M_2^{(x)}) = CV_C = c(CV_1^{(y)}, M_2^{(y)}) \text{ where } CV_C = z$$

4. METHODS

One of the major techniques used to conduct the attack is finding a *differential. path*, a form of crypt-analysis, that is the study of how differences in input affect the resultant output. In the case of SHA-1, it allows us to obtain a precise description of bit differences in state words (A, B, C, D, E) that form the chaining value and the message words W_0 ,..,W_{79} that update the chaining value. Additionally, constructing a differential path helps us to understand how these bit differences should propagate over the 80 rounds of the compression function. The composition of differential attacks on SHA-1 have become increasingly specific overtime; consisting of a *non-linear* differential path with low probability as well as a *linear* differential path of high probability. The terms non-linear and linear correspond to how the paths were obtained in relation to the internal functions of SHA-1. More specifically, the linear path works around the XOR operations as well as the permutations from the message expansion schedule whereas the non-linear path deals with AND, OR, and NOT operations. This is because XOR is a linear operation since bits undergoing it can be represented through linear equations while AND, OR, and NOT operations cannot be expressed nor solved as linear equations.

The linear path is constructed by evaluating message pairs until one is found that behaves linearly. The goal is to find two sets of message words such that after 6 rounds, both sets of message words produce the same chaining value thus resulting in a *local collision*.

Consider two sets of message words, where $M = W_i$,..,W_{i+5} is the original set and $M^{'} = W_i^{'}$,..,$W_{i+5}^{'}$ is the modified set. A local collision can be achieved by modifying one set of message words $M^{'}$ such that the first message word $W_i^{'}$, within the set, creates a disturbance, through a perturbing bit, on one of the state words $(A, B, C, D, E)_{i+1}$ that form the chaining value of the following round. The attacker can then modify subsequent message words $W_{i+1}^{'}$, $W_{i+2}^{'}$, $W_{i+3}^{'}$, $W_{i+4}^{'}$ and $W_{i+5}^{'}$ from the same set $M^{'}$ to correct or diffuse the disturbance, that was created initially, within the next five rounds. Thus, producing a local collision through two different paths from $(A, B, C, D, E)_i$ to $(A, B, C, D, E)_{i+6}$. The first path uses the original set of message words M and the second path uses the modified set of message words $M^{'}$ [1].

The attacker must consider both XOR differences as well as modular differences while expressing the description of the state and message words as those operations form the core foundation of the internal function within SHA-1. The differences between the state and message words can be expressed in the form of signed XOR differences which is similar to XOR differences with the exception of having a value for the differing bit represented by a sign.

Local collisions are a fit basis to generate differential paths of good probability. The main obstacle to achieve local collisions is the fact that the attacker does not control all of the message words as most of them are generated through the message schedule. This obstacle can be overcome by chaining local collisions along a *disturbance vector* (DV) such that the pattern of local collisions is compatible with the

message expansion of SHA-1 [11]. The disturbance vector is a set of 16 message words of 32-bit length each, that has been expanded by the message schedule of SHA-1. Every "1" bit position of the disturbance vector marks the start of a local collision.

4.1 Disturbance Vector Selection

The message expansion schedule within the compression function may be defined through two directions: Forward and Backward expansions. The sequence of the first 16 words $W_0, ..., W_{15}$ is known as the *information window* since they can be defined by the attacker [3]. Furthermore, these initial 16 words may be expanded forward to obtain the remaining 64 message words $W_{16}, ..., W_{79}$ using the recursive linear equation demonstrated in Section 2 which is the standard expansion method of SHA-1.

Similarly, we can expand the initial fixed 16 words through backward expansion to obtain $W_{-64}, ..., W_{-1}$ using the recursive linear equation below.

$$W_i = (W_{i+16} \ggg 1) \oplus W_{i+13} \oplus W_{i+8} \oplus W_{i+2}$$

$$\text{for } -64 \leq i \leq -1$$

Any sequence of consecutive 80 message words $W_i, ..., W_{i+79}$ with $-64 \leq i \leq 0$ is a valid expanded message which can be fed into the compression function. For a given information window, we can construct an *extended expanded message* (EEM) which contains 144 message words obtained through both forward and backward expanded words of the initial 16 words that form the information window [3]. Thus, the EEM consists of the following 32-bit message words:

$$W_{-64}, ..., W_{-1}, W_0, ..., W_{15}, W_{16}, ..., W_{79}$$

Each EEM consists of 65 valid expanded message words, each of which is a potential candidate as a disturbance vector.

4.2 Construction of a non-linear differential path

Once a disturbance vector has been selected, the linear part of the differential path is established. Next, the procedure for the attack requires constructing an appropriate non-linear path over the first 16 rounds that connects the chaining value differences to the "1" bit positions of the disturbance vector through the remaining rounds.

For the first near-collision attack, an arbitrary prefix can be included to fulfill some conditions on the chaining value. Consequently, this allows much more freedom in terms of constructing a non-linear path as it is not restricted to a specific value of the chaining value pair. However, the non-linear path for the second near-collision attack has to start from the resultant chaining value pair. The second near-collision attack, however, must cancel the specific difference in the resultant chaining value pair [10].

5. COMPUTATION

The computation of the near-collision block pairs were scattered across the world to shape the attacks into a distributed computation model with two specific purposes. Firstly, to lessen the wasted time of computational failures. Secondly, to conduct computations independently without duplication of work.

The first part of the attack, which corresponds to the computation of the first near-collision block pair $M_1^{(x)}$ and $M_1^{(y)}$, was conducted on a heterogeneous CPU cluster hosted by Google which was distributed over 8 different physical locations. The computation was divided into small jobs that were to produce partial solutions up to round 61 of the compression function with an hour of expected run time. Consequently, this run time of one hour proved to be successful against several kinds of failures such as machine or network issues. The second and final phase of the attack that corresponds to the generation of the second near-collision block pair $M_2^{(x)}$ and $M_2^{(y)}$ was conducted on a collection of heterogeneous GPUs hosted by Google. The generation of the second near-collision block pair was significantly more expensive than that of the first near-collision block pair. The first near-collision block pair was found after spending 3583 core years up to round 61, whereas, the second near collision block required 2987 core years of computation [10].

The full-collision attack is 100,000 faster than the brute force attack that relies on the birthday paradox. The full-collision attack can be conducted with processing power equivalent to 6,5000 years of single-CPU computations and 110 years of single-GPU computations. On the other hand, the brute-force attack would require 12,000,000,000 GPU years to complete and is therefore impractical [6].

6. CONCLUSION

Any Certification Authority abiding by the CA/Browser Forum regulations is not allowed to issue SHA-1 certificates anymore. Additionally, Chrome will consider any websites protected with SHA-1 certificates as insecure.

The full collision attack on SHA-1 has proved to be successful in practice and thus requires immediate reconsideration of SHA-1's use in many applications such as GIT. Although, the computational power required to conduct a full collision attack on SHA-1 is enormous, it is not impossible to do given enough time and numerous powerful CPUs and GPUs. Other alternatives such as SHA-236 or SHA-3 should be used in future security protocols to prevent potential hacks.

References

[1] Florent Chabaud and Antoine Joux. "Differential Collisions in SHA-0". In: *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*. Vol. 1462. Lecture Notes in Computer Science. Springer, 1998, pp. 56–71. DOI: 10.1007/BFb0055720.

[2] Rafi Chen Eli Biham. *Near-Collisions of SHA-0*. Cryptology ePrint Archive, Report 2004/146. https://eprint.iacr.org/2004/146. 2004.

[3] Stephane Manuel. *Classification and Generation of Disturbance Vectors for Collision Attacks against SHA-1*. Cryptology ePrint Archive, Report 2008/469. https://eprint.iacr.org/2008/469. 2008.

[4] R. Rivest. "The MD5 Message-Digest Algorithm". In: United States: RFC Editor, 1992.

[5] Ronald L. Rivest. "The MD4 Message Digest Algorithm". In: *Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO '90. Berlin, Heidelberg: Springer-Verlag, 1991, pp. 303–311. ISBN: 3-540-54508-5. URL: http://dl.acm.org/citation.cfm?id=646755.705223.

[6] *Shattered*. https://shattered.io/. Accessed: 2018-03-20.

[7] Marc Stevens. "New Collision Attacks on SHA-1 Based on Optimal Joint Local-Collision Analysis". In: *Advances in Cryptology - EUROCRYPT 2013*. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 245–261. DOI: 10.1007/978-3-642-38348-9_15. URL: https://www.iacr.org/archive/eurocrypt2013/78810243/78810243.pdf.

[8] Marc Stevens, Arjen Lenstra, and Benne Weger. "Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities". In: *Proceedings of the 26th Annual International Conference on Advances in Cryptology*. EUROCRYPT '07. Barcelona, Spain: Springer-Verlag, 2007, pp. 1–22. ISBN: 978-3-540-72539-8. DOI: 10.1007/978-3-540-72540-4_1. URL: http://dx.doi.org/10.1007/978-3-540-72540-4_1.

[9] Marc Stevens et al. *Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate*. 2009. DOI: 10.1007/978-3-642-03356-8_4. URL: https://www.iacr.org/archive/crypto2009/56770054/56770054.pdf.

[10] Marc Stevens et al. *The first collision for full SHA-1*. Cryptology ePrint Archive, Report 2017/190. https://eprint.iacr.org/2017/190. 2017.

[11] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. "Finding Collisions in the Full SHA-1". In: *In Proceedings of Crypto*. Springer, 2005, pp. 17–36.