Cryptanalysis of CRUSH hash structure[1]

Nasour Bagheri [1] n_bagheri@iust.ac.ir

Majid Naderi [1] m_naderi@iust.ac.ir

Babak Sadeghiyan [2] basadegh@ce.aut.ac.ir

[1] Electrical Engineering Department, Iran University of Science and Technology (IUST)

[2] Computer Engineering Department, Amirkabir University of Technology

Tehran, Iran

Abstract

In this paper, we will present a cryptanalysis of CRUSH hash structure. Surprisingly, our attack could find pre-image for any desired length of internal message. Time complexity of this attack is completely negligible. We will show that the time complexity of finding a pre-image of any length is *O(1)*. In this attack, an adversary could freely find a pre-image with the length of his own choice for any given message digits. We can also find second pre-image, collision, multi-collision in the same complexity with our attack.
In this paper, we also introduce a stronger variant of the algorithm, and show that an adversary could still be able to produce collisions for this stronger variant of CRUSH hash structure with a time complexity less than a Birthday attack.

Keywords: CRUSH hash structure, Hash function, Collision, Pre-image, Second pre-image, Multi-collision attack.

1. Introduction

Hash functions are used widely as a cryptographic primitive for generating digital signatures and message authentication codes. Each cryptographic hash functions should satisfy some security criteria. The main security requirement for a hash function is its collision freeness. Informally, it means that no attacker should be able to find a pair of different messages M and M' leading to equal hash values. Moreover, hash functions with output block length smaller than 160 bits are nowadays considered as insecure, due to general birthday attack.
In practice, building a cryptographic function with an input of variable size is not a simple task. Most hash functions are based on an iterated construction that makes use of a compression function, whose inputs have fixed sizes. A well-known family of such a construction are MDx hash function family, including MD4 [1], MD5 [2], and SHA [3]. The principle behind this structure is that if there is a computationally collision-free function f from m bits to n bits, where $n < m$ [4] then there is a collision-free hash function h mapping a message of arbitrary polynomial length to a k-bit string. Due to Merkle-Damg°ard theorem, it is claimed that if IV is fixed and if the padding procedure includes the length of the input into the padding bits, then h is collision-resistant if f is collision-resistant [4,5]. Hence, it has been generally thought that the problem of designing a collision-resistant hash function has been reduced to the problem of designing a collision-resistant compression function. However, the multi-collision attack[6] and the multi-block differential collision attack on MD5, SHA-0 and SHA-1 [7-9] indicates that a collision-resistant compression function is not a sufficient condition of a collision-

[1] Relatively same work, but independent, has been published in SAC 2007 by Matthew Henricksen and Lars R. Knudsen .

resistant hash function, but only a necessary condition. It means that a secure and collision- resistant hash function will be based not only on a collision-resistant compression function, but also on a collision-resistant structure.
In *Cryptographic Algorithms and their Uses 2004 conference*, Praveen Gauravaram et al. proposed a new [10] hash schema called CRUSH based on iterated halving (IH) as a new approach for creating hash function. They believed that their proposed schema would offer a good trade off between security and efficiency. This class of algorithms may be instantiated with any reduced round block cipher which allows a subtle security/performance tradeoff. They have described good characteristics for their new structure.
In this paper, we will show that their proposed schema could not satisfy any of the described characteristics. Our studies show that the proposed algorithm is strongly week in fact. In our attack, an adversary can determine the pre-image for any hash value with a time complexity of O(1). The strength of algorithm against second pre-image attack is not better than pre-image. To enhance CRUSH, we propose to modify its B function. We show that this change makes this algorithm more secure, but not perfect.
This paper is organized as following. Section 2 is a brief description of CRUSH hash function. Section 3 describes our attack for finding pre-image, second pre-image and collision for this structure. In section 4, we discuss on different variant of algorithm and its security properties. Conclusions will be presented in section 5.

2. CRUSH Hash Structure

The CRUSH hash structure can be considered as a general hash function construction. This structure is illustrated in figures 1 and 2. To build an n-bit hash function, it employs a bijective function B, which is hence a reversible function. This function could be any 128-bit block cipher. According to [10], this function can also be seen as $\{0,1\}^{128} \times \{0,1\}^{196} \to \{0,1\}^{128}$ mapping. CRUSH structure might be considered as a modified CBC mode application of a block cipher. While CBC mode converts a sequence of input blocks into an equal length sequence of output blocks, CRUSH structure modifies its input and only half of each output block is kept as data. The process of halving the output block can be repeated until only one block is yielded. Hence, $2m$ block encryptions is performed in this process, where m is the total number of 128-bit blocks obtained after the padding operation. Another difference with CBC is the application of the other half of each output block as input to a Bijection Selection Component (BSC). If n-bit message digests are needed, it is produced by truncating the iterated halving process up to the state with down to n bits. Figure 1 shows the overall idea of IH, while the proposed structure for the halving stage is shown in Figure 2, according to [10].
For creating message digests with CRUSH, an initialization operation must first be performed. For this purpose, the 256-bit initial value of the BSC is set to a fixed constant. The four blocks are set to following values:

$$x_{-1} = 0x12835B0145706FBE$$
$$x_{-2} = 0x243185BE4EE4B28C$$
$$x_{-3} = 0x550C7DC3D5FFB4E2$$
$$x_{-4} = 0x72BE5D74F27B896F$$

Similar to other hash schemes, CRUSH requires a padding function P, which is included in the pre-processing step. The standard preprocessing step involves padding the original message M to a multiple of 64-bits (which is half the block size of B) and inserting a coded presentation of $len(H)$ and $len(M)$ in the last block, where $len(H)$ is the length of desired message digest and $len(M)$ is the length of message before the padding operation. The padded message consists of the original message appended with a single bit ‘1’ followed by a number of ‘0’ bits as required. For short messages, some necessary half-blocks of zeroes are inserted to ensure the padded message contains at least fourteen half-blocks of 64-bits each. Then, the padded message is completed by appending a final half-block of forty-four 0 bits, 20-bit HLEN field which specifies the length of the desired message digest and a 64-bit MLEN field which specifies the length of the original or unpadded message.

After the pre-processing stage, the Iterated Halving process is performed. For each stage, $i = 0$ to $D-1$, and for each pair of input data of half-blocks, $\{M\}$, the value of $x_i \| y_i = F(f1_i, f2_i, f3_i, D_{2i}, D_{2i+1})$ is calculated, where $f1, f2$ and $f3$ are Boolean functions and are operated on previous values of x. Half-blocks of y are stored to become the input message for the next stage. These functions are defined as below:

$$f1_i = (x_{i-1} \oplus x_{i-2}x_{i-3} \oplus x_{i-1}x_{i-2}) + 0x\text{B5C0FBCFEC4D3B2F}$$
$$f2_i = (x_{i-2} \oplus x_{i-3} \oplus x_{i-2}x_{i-3} \oplus x_{i-1}x_{i-2} \oplus x_{i-1}x_{i-3}) + 0x\text{E9B5DBA58189DBBC}$$
$$f3_i = (1 \oplus x_{i-1} \oplus x_{i-3} \oplus x_{i-2}x_{i-3}) + 0x\text{3956C25BF348B538}$$

for every stage. When the number of input blocks is not even, there will be only a single block remained for processing at the end of the stage. In this case the Bijection Selection half-block x_{i-2} is copied and used as D_{2i+1} at the final input for the stage. The current value of the key-feedback half-blocks $\{x_i,...,x_{i-3}\}$ is retained among the stages, as the initial value of the data for the next stage.

After stage D, the Iterated Halving process is terminated and the last stored set of y values, e.g. the most recently generated 256-bits, is used as the final hash output. Unlike MDx hash schemes, there is no global feedforward addition for this hash structure.

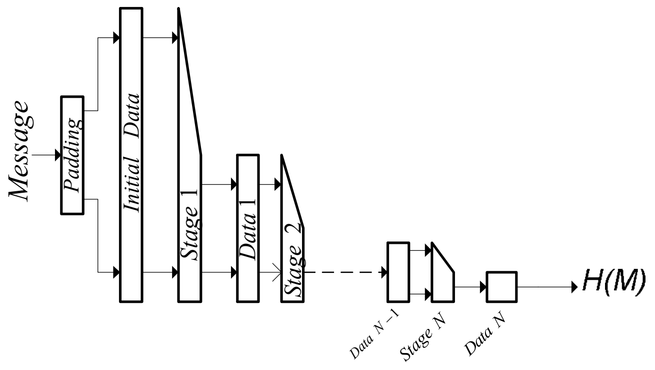

Figure 1. Iterated Halving (IH), it is a multi-stage compression process without any global feedforward.

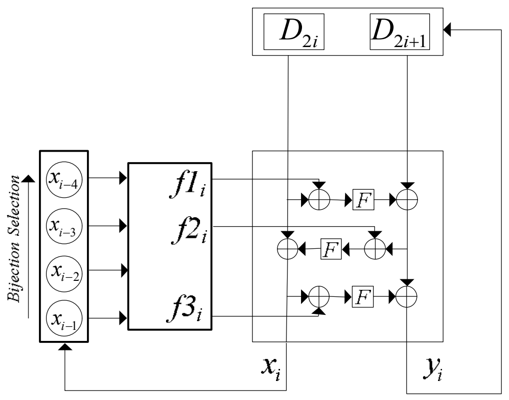

Figure 2. The CRUSH compresses function.

Without loss of generality, we assume that B is a secure block cipher of $\{0,1\}^{128} \times \{0,1\}^{196} \to \{0,1\}^{128}$, and denote it as B. Obviously, this is the strongest version of CRUSH. If there is any attack to this structure, it could be applied to the original CRUSH scheme with its described B function. Figure 3 illustrates the generalized CRUSH hash structure.

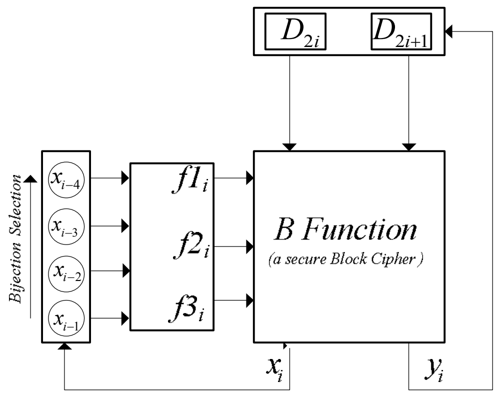

Figure 3. Generalized CRUSH hash structure.

3. Attack on CRUSH

According to the definition of B function, the value of $x_i \| y_i = F(f1_i, f2_i, f3_i, D_{2i}, D_{2i+1})$. B function is a bijective function and hence a reversible one. Considering $f1_i, f2_i, f3_i$ definitions, if an adversary determines $x_{i-1}, x_{i-2}, x_{i-3}, y_i, x_i$, he can apply F^{-1} to obtain D_{2i}, D_{2i+1}, where F^{-1} is the inverse of F:

$$D_{2i} \| D_{2i+1} = F^{-1}(f1_i, f2_i, f3_i, x_i, y_i)$$

If B is a block cipher, then we may do following assignments:

$$\begin{aligned}
&Key: k_i = f1_i \| f2_i \| f3_i, \\
&Plain\,text: P_i = D_{2i} \| D_{2i+1}, \\
&Cipher\,text: C_i = x_i \| y_i, \\
&Encryotion\,System: C_i = B(k_i, P_i), \\
&Decryotion\,System: P_i = B^{-1}(k_i, C_i).
\end{aligned}$$

It can be shown that CRUSH Hash Structure is vulnerable to pre-image attack, second pre-image attack, and collision attack. Lemmas 1 to 3 prove these claims.

Lemma 1: CRUSH hash structure is vulnerable to pre-image attack, where the time complexity of attack is $O(1)$.

Proof: Our attack scheme is rather easy. To obtain a pre-image, an adversary should do the operations in the reverse order. He starts with Y and tries to find a message M in such a way that $H(M)=Y$, where H is the CRUSH hash structure. The adversary should define the original padded message length of his choice, where we show it by $len(M)$. The depth of *IH* process is calculated as $D \le \log_2 \frac{len(M)}{len(H)}$, when D is rounded up to the next integer, where $len(H)$ is the number of bits of the hash value. $D_{\min}$ have to be greater than two. Given the minimum $len(H)$ of 256 bits, a lower bound of 1024 bits is imposed on $len(M)$. The padded message of minimum length can be shown as below:

$$M = M_0 \| M_1 \| .. \| M_{15}$$

Where, M_i is a message block with 64 bits length . According to CRUSH hash structure:

$$\begin{aligned}
&D_0 = M_0, ..., D_{15} = M_{15},\\
&x_{-4} = 0x72BE5D74F27B896F, x_{-3} = 0x550C7DC3D5FFB4E2,\\
&x_{-2} = 0x243185BE4EE4B28C, x_{-1} = 0x12835B0145706FBE,\\
&x_0 \| D_{16} = C_0, .., x_{11} \| D_{27} = C_{11},\\
&H(M) = D_{24} \| D_{25} \| D_{26} \| D_{27}.
\end{aligned}$$

If an adversary requires to find a pre-image for a given message digest Y, he sets the values for $D_{24}, ..., D_{27}$ in such a way that:

$$D_{24} = Y_{1-64}, D_{25} = Y_{65-128}, D_{26} = Y_{129-196}, D_{27} = Y_{197-256}$$

Where $D_l = Y_{i-j}$ denotes that the content of register Y, from *i*th bit through *j*th bit, is copied to register D. He should guess $x_0, ..., x_{11}$, while $x_{-1}, ..., x_{-4}$ are assigned the prefixed values. For each stage of CRUSH, the adversary calculates $D_{23}, ..., D_0$ as follows:

$$P_i = D_{2i} \| D_{2i+1} = B^{-1}(k_i, C_i) = B^{-1}((f1_i \| f2_i \| f3_i), x_i \| y_i)$$

To follow the operations between stages, he must retain the current value of the key feedback half blocks $x_i, ..., x_{i-3}$ as the initial value of data for the next stage. After appropriate processing, he obtains the pre-image for a given message digest as:

$$M_0 \| .. \| M_{15} = P_0 \| .. \| P_7$$

Table 1 shows how the adversary obtains these values. For finding a pre-image message with another length, only the number of stages is changed. In this way, an adversary may find infinite number of pre-images for a given $H(M)$. The complexity of generating each pre-image is exactly one CRUSH operation, but with the use of B^{-1} instead of B. Since in most block ciphers B^{-1} is as computationally complex as B, the complexity of finding pre-image is $O(1)$.□

Lemma 2: CRUSH hash structure is vulnerable to second pre-image attack with the time complexity of $O(1)$.

Proof: The procedure of finding a second pre-image is rather similar to what we have done for finding a pre-image. Assume that an adversary with a given message M and its related message digest Y desires to obtain $M' \neq M$ where $H(M') = H(M)$. For this purpose he can run the following algorithm:

1. Calculate the message digest Y so that $Y = H(M)$.
2. Suggest values for x to be used in the pre-image obtaining procedure.
3. Use the discussed procedure to obtain pre-image and find M', the pre-image for Y.
4. If $M' = M$, change the suggested values for x and repeat 3rd step of algorithm.
5. Output the M' as the M second pre-image.

The time complexity of this algorithm is related to 3rd step. Since the probability of 4th step satisfaction is completely low, adversary can hope to win in the first running of 3rd step, otherwise he will win in negligible number of running the algorithm. So, the time complexity of this attack is $O(1)$.□

Table 1: Calculations for the minimum length pre-image for the given value of $Y = H(M)$.

Stage	step	Known	guess	Computation	output
2	1	$y_{11} = D_{27}$	$x_7,\ldots,x_{11}$	$D_{22} \Vert D_{23} = B^{-1}((f1_{11} \Vert f2_{11} \Vert f3_{11}), x_{11} \Vert y_{11})$	D_{22}, D_{23}
	2	$y_{10} = D_{26}, x_7 x_8, x_9, x_{10}$	x_6	$D_{20} \Vert D_{21} = B^{-1}((f1_{10} \Vert f2_{10} \Vert f3_{10}), x_{10} \Vert y_{10})$	D_{20}, D_{21}
	3	$y_9 = D_{25}, x_6, x_7, x_8, x_9$	x_5	$D_{18} \Vert D_{19} = B^{-1}((f1_9 \Vert f2_9 \Vert f3_9), x_9 \Vert y_9)$	D_{18}, D_{19}
	4	$y_8 = D_{24}, x_5, x_6, x_7, x_8$	x_4	$D_{16} \Vert D_{17} = B^{-1}((f1_8 \Vert f2_8 \Vert f3_8), x_8 \Vert y_8)$	D_{16}, D_{17}
1	5	$y_7 = D_{23}, x_4, x_5, x_6, x_7$	x_3	$D_{14} \Vert D_{15} = B^{-1}((f1_7 \Vert f2_7 \Vert f3_7), x_7 \Vert y_7)$	D_{14}, D_{15}
	6	$y_6 = D_{22}, x_3, x_4, x_5, x_6$	x_2	$D_{12} \Vert D_{13} = B^{-1}((f1_6 \Vert f2_6 \Vert f3_6), x_6 \Vert y_6)$	D_{12}, D_{13}
	7	$y_5 = D_{21}, x_2, x_3, x_4, x_5$	x_1	$D_{10} \Vert D_{11} = B^{-1}((f1_5 \Vert f2_5 \Vert f3_5), x_5 \Vert y_5)$	D_{10}, D_{11}
	8	$y_4 = D_{20}, x_1, x_2, x_3, x_4$	x_0	$D_8 \Vert D_9 = B^{-1}((f1_4 \Vert f2_4 \Vert f3_4), x_4 \Vert y_4)$	D_8, D_9
	9	$y_3 = D_{19}, x_{-1}, x_0, x_1, x_2, x_3$	-	$D_6 \Vert D_7 = B^{-1}((f1_3 \Vert f2_3 \Vert f3_3), x_3 \Vert y_3)$	D_6, D_7
	10	$y_2 = D_{18}, x_{-2}, x_{-1}, x_0, x_1, x_2$	-	$D_4 \Vert D_5 = B^{-1}((f1_2 \Vert f2_2 \Vert f3_2), x_2 \Vert y_2)$	D_4, D_5
	11	$y_1 = D_{17}, x_{-3}, x_{-2}, x_{-1}, x_0, x_1$	-	$D_2 \Vert D_3 = B^{-1}((f1_1 \Vert f2_1 \Vert f3_1), x_1 \Vert y_1)$	D_2, D_3
	12	$y_0 = D_{16}, x_{-4}, x_{-3}, x_{-2}, x_{-1}, x_0$	-	$D_0 \Vert D_1 = B^{-1}((f1_0 \Vert f2_0 \Vert f3_0), x_0 \Vert y_0)$	D_0, D_1

Lemma 3: CRUSH hash structure is vulnerable to collision attack with the complexity of $O(1)$.

Proof: The procedure of finding a collision is similar to what we have done for finding a second pre-image. Assume that an adversary desires to obtain M and M' so that $M' \neq M$ and $H(M') = H(M)$. For this purpose he just requires to select a message M in random way and obtain $M' \neq M$, a second pre-image related to it. Now he have a collided pair of messages which is M and M'.□

Obviously, the attacker can use the discussed procedures in lemma 1 to lemma 3 to generate multi-pre-image, multi-second pre-image, and multi-collision respectively in negligible time complexity.
Surprisingly, all the procedures for finding the values of pre-image, second pre-image, and collision are deterministic. The only simplification is the elimination of the padding in the founded pre-image, second pre-image, and collision. Such simplifications are negligible in many attacks of this kind.
Hence, we conclude that this hash structure is completely far from a random oracle required for hash definition.

4. Modification of CRUSH

To enhance CRUSH, we propose to change B function from a bijective function to a random oracle. We show that this modification makes this structure more secure, but not as perfectly as required secure. If we consider B function as a $\{0,1\}^{128} \times \{0,1\}^{196} \rightarrow \{0,1\}^{128}$ mapping, we also show that it could not be secure for $len(H) > 128$. When B is a random oracle, since it is not a reversible function any more, the previous attacks could not be applied to CRUSH.

Definition 1: A **random oracle** is a theoretical black box that responds to every query with a (truly) random response chosen uniformly from its output domain, except that for any specific query, it responds the same way every time it receives that query.

The following Lemma gives an upper bound for the output length of this modified structure, which we call it M-CRUSH.

Lemma 4: M-CRUSH hash structure is immune against a collision attack for at most $len(H) \leq 128$. For a longer output, the adversary could find collisions in a time complexity less than that of birthday attack.

Proof: Consider M-CRUSH definition. It is similar to CRUSH. The only difference with CRUSH is in B mapping being a random oracle instead of a bijective function in the original CRUSH. If an adversary could find two different pairs of message blocks $M_0 \| M_1 = D_0 \| D_1$ *and* $M'_0 \| M'_1 = D'_0 \| D'_1$ in such a way that:

$$x_0 \| y_0 = F(f1_0, f2_0, f3_0, D_0, D_1) = F(f1_0, f2_0, f3_0, D'_0, D'_1),$$

then, he could find a collision. For example, he wants to create a collision with minimum length of message. He sets $M = M_0 \| M_1 \| M_2 \| \ldots \| M_{15}$ and $M' = M'_0 \| M'_1 \| M_2 \| \ldots \| M_{15}$. Since the initial value for $x_0, \ldots, x_{-3}$ is remained the same for this messages and $x_1 = x'_1$ *and* $y_1 = y'_1$, all internal states for these two messages are the same, and therefore the final output would be equal. The computational complexity of finding this collision is related to the first part of attack, i.e., finding those two messages block in such a way that $x_1 = x'_1$ *and* $y_1 = y'_1$ satisfied. Since B is a random oracle function, and *x* and *y* have the same length of 64-bit, the time complexity of finding this collision is $O(2^{64})$, which is proportional to a 128-bit length hash function.□

Lemma 5 shows that this enhanced structure is vulnerable to multicollision attack and complexity of finding 2^k -way collision is $O(k2^{64})$ which far from expected value from birthday attack $\Omega\left(2^{n\left(\frac{k-1}{k}\right)}\right)$.

Lemma 5: M-CRUSH hash scheme with hash value longer than 64-bit is vulnerable to multi-collision attack.

Proof: Consider M-CRUSH definition. To find a multi-collision on this structure, the adversary can run the following algorithm:

1. for i=1 to k do:
    a. Find two different pairs of message blocks $M_{2i} \| M_{2i+1} = D_{2i} \| D_{2i+1}$ *and* $M'_{2i} \| M'_{2i+1} = D'_{2i} \| D'_{2i}$ in such a way that:
    $$x_i \| y_i = F(f1_i, f2_i, f3_i, D_{2i}, D_{2i+1}) = F(f1_i, f2_i, f3_i, D'_{2i}, D'_{2i+1}),$$
    b. The x half-block of output is used to update BSC component values,
    c. The y half-block is saved for the next stage of Iterated Halving,
2. Do padding process and attach HLEN and MLEN,
3. Do ordinary Iterated Halving processes down to the message digest length, in the remaining stages,
4. Output the 2^k messages of the form $(m_1, m_2, ..., m_k, Padding)$, where m_i is one of the two blocks of $M_{2i} \| M_{2i+1}$ or $M'_{2i} \| M'_{2i+1}$.

Clearly, these 2^k different messages built as above all yield the same final value. The time complexity of this attack is related to 1st step of the attack which is $O(k2^{64})$. Hence the structure is secure against multi-collision for hash values up to 64-bit. □

5 Conclusion

In this paper, we have shown that finding pre-image, second pre-image, and collision on CRUSH Hash Structure is possible in time complexity of $O(1)$. We proved that attacker could find infinite number of pre-images or second pre-images for a given length of his choice. Obviously, the proposed hash structure is completely vulnerable.

We also showed that if we modify B function of this structure to a random oracle, it makes the hash structure much stronger and it can be immune against our proposed attacks for output length up to 128 bits, but it is still vulnerable to a multi-collision attack.

6 Reference

[1]. Rivest.R.L,"The MD4 Message – Digest Algorithm ", Network MIT laboratory for Computer Science and RSA Data Security , Inc RFC 1320 , April 1992.

[2]. Rivest R.L, "The MD5 message-digest algorithm", Request for Comments (RFC1320), Internet Activities Board, Internet Privacy Task Force, 1992.

[3]. National Institute of Standards and Technology (NIST) , Computer Systems Laboratory. Secure Hash Standard. Federal Information Processing Standards Publication (FIPS PUB) 180-2, August 2002.

[4]. Ivan Damgard, “A design principle for hash functions,” in Advances in Cryptology –CRYPTO 89, Springer-Verlag, 1989, pp. 416–427.

[5]. R. C. Merkle, “One-way hash functions and DES.” in Advances in Cryptology – Crypto’89 (G. Brassard, ed.), no. 435 in Lecture Notes in Computer Science, pp. 428–446, Springer-Verlag, 1990.

[6]. Antoine Joux, “Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions,” Advances in Cryptology-CRYPTO ’04, Springer-Verlag, 2004, pp. 306–316.

[7]. Eli Biham, Rafi Chen, and Antoine Joux etc, “Collisions of SHA-0 and Reduced SHA-1,” Advances in Cryptology-EUROCRYPT 2005, Springer-Verlag, 2005, pp. 36–57.

[8]. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu, “Finding collisions in the full SHA-1,” Advances in Cryptology—CRYPTO ’05, Springer-Verlag, 2005, pp. 17–36.

[9]. Xiaoyun Wang and Hongbo Yu, “How to Break MD5 and Other Hash Functions,” Advances in Cryptology - EUROCRYPT ’05, Springer-Verlag, 2005, pp. 19–35.

[10]. Praveen Gauravaram, William Millan, Lauren May, “CRUSH: A New Cryptographic Hash Function using Iterated Halving Technique.”, c4: 28-39.