MD5 and SHA-1 Collision Attacks: A Tutorial

Alan Savage

Abstract— **MD5 and SHA-1 are well developed and popular cryptographic hash functions used for various security purposes, including password storage and integrity checking. Both of these algorithms have been shown to be vulnerable to collisions. In this tutorial, collision attacks for both of these algorithms will be presented and analyzed. In addition, the implications of these attacks on the development of future hash functions will be discussed.**

I. Introduction

Hash functions are many-to-one functions designed to compress some data, known as a message, into a fixed number of bits, known as a hash or a digest. They are widely used in many areas for integrity checking, password storage, message signing, SSL, timestamping, random number generation, and various other cryptographic protocols. These applications rely on the fact that hashes are a nearly unique identification of a particular message and that they are not reversible. In order for the messages to be nearly unique, hash functions are designed such that it should be computationally infeasible to find collisions, i.e. two different messages that have the same hash value. The act of searching for collisions for a particular function is known as a collision attack.

MD5 and SHA-1 are two of the most popular hash functions and are in widespread use. However, MD5 and SHA-1 are vulnerable to collision attacks based on differential cryptanalysis. MD5 is completely broken in that collisions can now be found within a few minutes on modern machines. SHA-1, while not completely broken, is showing signs of weakness. That is, the attacks on SHA-1 have a lower time complexity than a brute force approach. In this paper, we will briefly overview each of these functions and then examine the attacks. We will conclude with some implications of these attacks.

II. A Brief Overview of MD5

MD5 is a hash function, using the Merkle-Damgärd construction, that takes messages of arbitrary length ($< 2^{64}$ bits) and compresses them into a 128-bit hash value[1]. The message M is divided into 512-bit blocks such that $M = (M_0, M_1, ..., M_n - 1)$. If the message length is not a multiple of 512 then it is padded by appending a single 1 bit followed by enough 0s to bring the length of the message to 64-bits less than a multiple of 512. The final 64-bits are an integer representing the bit-length of the original message.

The algorithm iterates over 512 bits of the message at a time, broken down into 16 32-bit messages $M_i = m = (m_0, m_1, .., m_{15})$. Each iteration has 4 rounds that each has 16 steps for 64 steps in total. The hash itself is broken into four 32-bit words and is initialized to the value given in Table I. Each round operates on four 32-bit words A, B, C, and D which, at the beginning of each iteration, are set component wise to the current value of the hash. Each round executes according to the following rules:

TABLE I
MD5 Initialization Vectors -

$H0$	=	0x01234567
$H1$	=	0x89ABCDEF
$H2$	=	0xFEDCBA98
$H3$	=	0x76543210

TABLE II
MD5 Round Functions

Round (i)	$F(X, Y, Z)$	g
0	$(X \wedge Y) \vee (\neg X \wedge Z)$	i
1	$(X \wedge Z) \vee (Y \wedge \neg Z)$	$(5 \times i + 1) \bmod 16$
2	$(X \oplus Y \oplus Z)$	$i(3 \times i + 5) \bmod 16$
3	$(Y \oplus (X \vee \neg Z))$	$(7 \times i) \bmod 16$

$$A = D$$
$$B = B + ((A + F(B, C, D) + K_i + m_g) <<< R_i)$$
$$C = B$$
$$D = C$$

F is a non-linear boolean function, K is a step specific constant, m_g is one word of the message indexed by g, and finally R_i is a round specific constant that specifies a left rotation by R_i bits. F and g are shown in Table II and are round specific. All addition is done mod2^{32}. At the end of each iteration, A,B,C, and D are each added compenent wise to the hash:

$$H0 = H0 + A$$
$$H1 = H1 + B$$
$$H2 = H2 + C$$
$$H3 = H3 + D$$

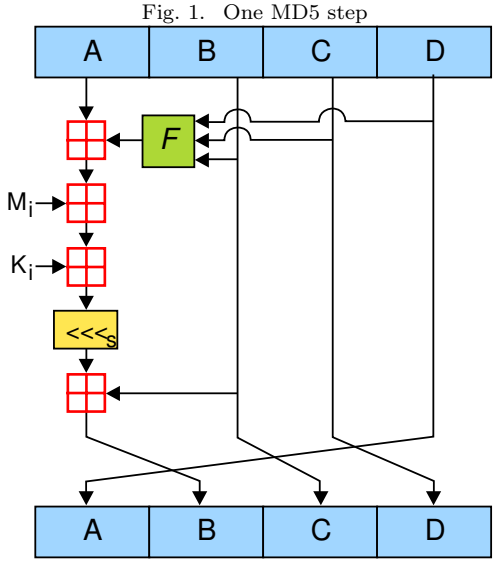
Figure 1 is a graphical representation of one step of MD5[2]. The final hash is a concatenation of the four hash words.

III. MD5 Collision Attack

An MD5 collision attack using differential cryptanalysis was exposed by Wang in [3]. That required at most 2^{39} MD5 operations to find a collision, and was subsequently improved by various research groups around the world and reduced to 2^{29} for the 2nd block by Klima in [4]. Black et al. give an execellent, in-depth explanation of the attack in [5]. Here we give a simplified overview of how the attack is carried out.

The generated collisions consist of two 1024-bit messages $M = (M_0, M_1)$ and $M' = (M'_0, M'_1)$, such that MD5(M) =

Fig. 1. One MD5 step



MD5(M'). Define:

$$C_0 = (0, 0, 0, 0, 2^{31}, 0, 0, 0, 0, 0, 0, 2^{15}, 0, 0, 2^{31}, 0)$$
and
$$C_1 = (0, 0, 0, 0, 2^{31}, 0, 0, 0, 0, 0, 0, -2^{15}, 0, 0, 2^{31}, 0)$$

Let $M_0' = M_0 + C_0$ and $M_1' = M_1 + C_1$, added component wise and mod2^{32}. Let $Q_i = B$ for sub-step i in MD5(M) and $Q_i' = B$ in the ith step of MD5(M'). $Q_i - Q_i'$ is a the differential for step i.

A set of differentials are given by [3], a_i where $0 <= i < 127$, such that if $Q_i' - Q_i = a_i$ for all i, then a collision occurs. This set of differentials is known as the 'differential path'. There is a set of conditions on Q_i that will ensure that all the differentials hold true if the conditions are satisfied. The sufficient conditions are conditions on the individual bits of the intermediate variables Q_i. For example, the least significant 6 bits of Q_8 must be 0, and the most significant bit of $Q_1 6$ must be equal to the most significant bit of $Q_1 5$. In [3], a set of conditions were given and said to be sufficient. However, Liang et al. gave counter-examples for those conditions and provided new conditions that are believed to be correct and sufficient in [6].

To perform the attack, we satisfy as many conditions on Q_i as possible, starting from Q_0, in a more or less deterministic way by intelligently modifying M. After that, we satisfy the rest of the conditions probabalistically by modifying the unfixed bits of M until all conditions are satisfied and a collision is found.

Here is the pseudocode from [5].

1: **for all** Block **do**
2: collision_found = false
3: **while** collision_found is false **do**
4: Select values $Q_{0:15}$ arbitrarily
5: Modify $Q_{0:15}$ to satisfy all first block conditionals and differentials
6: Compute $m_{0:15} from Q_{0:15}$
7: Satisfy all possible second block conditions and differentials using multi-message modification methods
8: Compute all Q_i and Q_i' and check if all differentials hold
9: **if** all differentials hold **then**
10: collision_found = true
11: **end if**
12: **end while**
13: **end for**

Single message modification is changing M such that all the conditions for the first round hold. The algorithm is as follows:

1: **for** i from 0 to 15 **do**
2: Change Q_i to satisfy conditions
3: $m_i = ((Q_i - Q_{i-1}) >>> s_i) - T_i - Q_{i-4} - F(Q_{i-1}, Q_{i-2}, Q_{i-3})$
4: **end for**

One of the key points of the attack is multi-message modification. That is, after all the first round conditions are satisfied, alter several message blocks to satisfy the second round conditions while keeping first round conditions satisfied. Lets say $Q_{16,31}$, ie. the most significant bit of sub-step 16 for round two, is 1 and needs to be zero to satisfy its condition. Because the shift amount for sub-step 16 is 5, adding 2^{26} to m_1 would satify the condition. However, this would also change Q_1. Other message modifications must be made to absorb that change. The chain of modifications that must be made is specific to that particular condition. These modifications seem to be calculated by checking each change that would be made and working backward to undo them similar to the single-message modification.

IV. A Brief Overview of SHA-1

SHA-1, like MD5, uses the Merkle-Damgärd construction, so they are very similar. It hashes a message of an arbitrary length ($< 2^{64}$ bits) into a 160-bit hash value. The message is broken into 512-bit blocks each of which is processed for 80 rounds. Each round has 160-bits of input broken down into five 32-bit words, A, B, C, D, and E. At the beginning of each block the round inputs are given the current hash value just as in MD5. Table IV gives the initialization vector for the first round of the first block. This input undergoes the following computation once each round:

$$A = F(B, C, D) + E + (A <<< 5) + W_i + K_i$$
$$B = A$$
$$C = B <<< 30$$
$$D = C$$
$$E = D$$

F is a non-linear boolean function in Table III, K_i is a round specific constant, W_i is the expanded message word of round i. W_i is defined as follows:

$$m_i = \begin{cases} m_i & 0 <= i <= 15 \\ (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) <<< 1 & 16 <= i <= 79 \end{cases}$$

TABLE III
SHA-1 Rounds Functions

Round (i)	$F(X,Y,Z)$
0..19	$(X \wedge Y) \vee (\neg X \wedge Z)$
20..39	$(X \oplus Y \oplus Z)$
40..59	$(X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z)$
60..79	$(X \oplus Y \oplus Z)$

TABLE IV
SHA-1 Initialization Vector

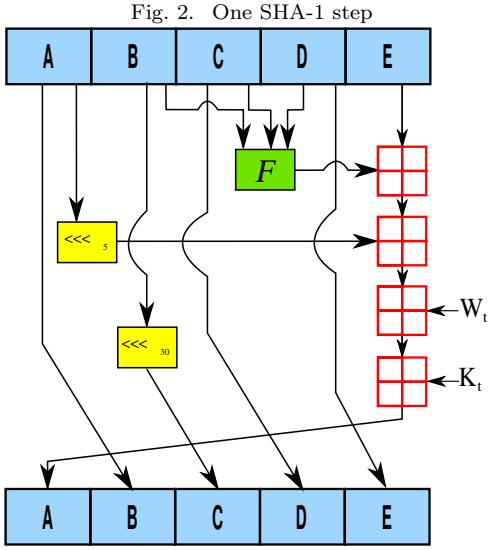
$H0$ = 0x67452301
$H1$ = 0xEFCDAB89
$H2$ = 0x98BADCFE
$H3$ = 0x10325476
$H4$ = 0xC3D2E1F0

At the end of each iteration, A,B,C,D, and E are each added compenent wise to the hash:
$H0 = H0 + A$
$H1 = H1 + B$
$H2 = H2 + C$
$H3 = H3 + D$
$H4 = H4 + E$
The final hash is the concatenation of the five hash words. Figure 2 shows one round of SHA-1[2].

V. SHA-1 Collision Attack

The first attack on the full SHA-1 with complexity $O(2^{69})$ was introduced by Wang in [7]. That attack was further reduced to $O(2^{63})$ in [8] by finding a better collision path, this was corroborated in [9]. The attack is similar to the MD5 attack in many ways. It is also a differential attack, so there is a differential path a_i with some conditions on it. The same message modification techniques are used to satisfy the conditions on the differential path. This section provides an overview of that attack.

Fig. 2. One SHA-1 step



Some key ideas in the attack are local collisions and disturbance vectors. A local collision is a collision within a few steps of the hash function, such that a difference in some step will be absorbed by the next five steps. The disturbance vector specifies the starting starting steps for the local collisions. For SHA-1, there are 80 disturbance vectors of 32-bits each. The vectors satisfy the recurrence relationship given by the SHA-1 message expansion in Section IV for rounds >= 16. Because of this, once 16 consecutive vectors have been chosen the other 64 have also been determined. In addition, corrections needed by differences in message bits for a local collision are satisfied by the same recurrence.

The complexity of the attack is proportional to the Hamming weight of the disturbance vector, so the key to an efficient attack is to find disturbance vectors with low Hamming weights then construct valid differential paths from those vectors. To find the disturbance vectors we note that they can be viewed as an 80 by 32 matrix where each entry, indexed i, j is a single bit. If a particular entry is 1 then a local collision is introduced at step i at bit j. Matrices with low hamming weights are likely to to concentrate non-zero entries in several consecutive columns. So, to find the disturbance vectors with low hamming weights we pick two entries i, j and $i, j-1$ as the top of two 16-bit columns, ie. the columns are $(i,j),(i+1,j),...,(i+15,j)$ and $(i,j-1),(i+1,j-1),...,(i+15,j-1)$. We then vary those columns through all 2^32 possibilities. Using the above strategy, first search for vectors predicting one-block collisions then vectors that predict near collisions and two-block collisions. Finally, compute the minimal Hamming weight disturbance vectors.

Now, given a disturbance vector with low Hamming weight, use it to construct a valid differential path. Unfortunately [7] is somewhat light on the details of how to do this construction. The main approach is (1) analyze the propagation of differences, (2) identify wanted and unwanted differences, and (3) use the Boolean function and the carry effect (given below) to introduce and absorb these differences. They offer the following high level ideas:

- Use subtraction instead of XOR for difference
- Take advantage of special properties of first boolean function. It can preserve, flip, or absorb an input difference.
- Take advantage of the carry effect to introduce extra bit differences: $2^j = -2^j - 2^{j+1} - ... - 2^{j+k-1} + 2^{j+k}$ for any k.
- Use difference message differences for the 6-step local collision
- Introduce extra bit differences to produce the impossible bit-differences in the consecutive local collisions corresponding to the consecutive disturbances in the first 16 steps, or to offset the bit differences of chaining variables produced by truncated local collisions.

Once a valid differential path is constructed, a set of sufficient conditions on the message and chaining variable can be derived. The conditions are defined in the same way

the MD5 conditions were defined except instead of using variable B, in SHA-1 the chaining variable is A. Again, details of how the conditions are derived are lacking, but the idea is to find what bits of M and A must be set in every round and what their differences are from the previous or next round.

The conditions can be satisfied by using the same message modification techniques on M that were used in the MD5 attack. All of this will generate a near collision on the first block. Then the second block will begin using the has of the first block as input. To construct the differential path for the second message blocks we apply the techniques previously used so that the input difference is absorbed by the first 16 rounds of the new differential path. Next, we set the conditions on the second block of the message so that the output difference has the opposite sign as the input difference. The differences will cancel out and thus, a two-block collision is born.

VI. Implications and Conclusion

The collision attack on MD5 has widespread implications. Many passwords are stored as an MD5 hash and are now vulnerable to exploitation. MD5 is also used for integrity checking in security protocols. The fact that collisions are now easily generated means that it can no longer be reliably used in this way. However, not all applications require collision resistance. For instance, hashes can be used for error detection when downloading large files.

Unlike MD5, the SHA-1 attack is still not feasible on current hardware, so current applications using this algorithm are still safe. However, it is possible and even likely that the attacks on SHA-1 will improve and hardware will improve making collision finding feasible. Alternatives to SHA-1 that are much harder to break exist, such as SHA-256 and SHA-512. But because these algorithms are in the same class as SHA-1 they are likely to be vulnerable to similar techniques. For that reason, NIST is orchestrating a competition to design a new hash function, SHA-3, to replace the older standards[10].

References

[1] R. Rivest, "The MD5 Message-Digest Algorithm," RFC 1321 (Informational), Apr. 1992.

[2] "Wikipedia," `http://en.wikipedia.org`.

[3] Xiaoyun Wang and Hongbo Yu, "How to break md5 and other hash functions," in *EUROCRYPT*, Ronald Cramer, Ed. 2005, vol. 3494 of *LNCS*, pp. 19–35, Springer.

[4] Vlastimil Klima, "Finding md5 collisions on a notebook pc using multi-message modifications," 2005.

[5] John Black, Martin Cochran, and Trevor Highland, "A study of the md5 attacks: Insights and improvements," in *FSE*. 2006, vol. 4047 of *LNCS*, pp. 262–277, Springer.

[6] Jie Liang and Xuejia Lai, "Improved collision attack on hash function md5. cryptology eprint archive," Tech. Rep., 2005.

[7] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu, "Finding collisions in the full sha-1," 2005, pp. 17–36, Springer.

[8] Xiaoyun Wang, "Cryptanalysis of sha-1 hash function," NIST Cryptographic Hash Workshop, 2005.

[9] Martin Cochran, "Notes on the wang et al. 2^{63} sha-1 differential path," Cryptology ePrint Archive, Report 2007/474, 2007, `http://eprint.iacr.org/`.

[10] "Cryptographic hash algorithm competition," `http://csrc.nist.gov/groups/ST/hash/sha-3/index.html`.