Reproducing And Improving MD5 Fast Collision Attack Algorithm

Meng Zheng
Electrical Engineering
Delft University of Technology
Delft, Netherlands
M.Zheng-3@student.tudelft.nl

Yuhang Tian
Embedded Systems
Delft University of Technology
Delft, Netherlands
Y.Tian-13@student.tudelft.nl

***Abstract*—This paper is the survey for exploring the MD5 fast collision based on M. Stevens' paper published in 2012 where he extends the research carried out by Wang et al. and shows that the original structure of differential path cannot be proved as satisfying the "sufficient conditions". Wang's collision method has the limitation of the identical prefix, whereas Steven breaks this limitation to make the method become chosen-prefix collision attacks, which is the main contribution for him compared to primitive achievements. Our work, in this survey, is mainly 1) to explain the significance of MD5 collision, 2) to reproduce the fast collision attack algorithm for MD5, 3) to improve the current collision method or 3) to apply the methods in a real attack.**

***Index Terms*—MD5, collision attack, cryptography**

I. Introduction

For a one-way hash function, the collision attack is to find two different messages M and M' that have the identical hash values $H(M) = H(M')$. In terms of the Merkle-Damgard 4 (MD4) family, the most efficient technology to find the collisions for them is *differential cryptanalysis*. The core idea is to investigate and control the differences between two preimages when they propagate through the hash blocks and to generate two identical digests at the end. Following this idea, Xiaoyun Wang et al., published a paper in 2004 and proposed an effective attack for MD5 and achieved the collision attack. The paper points out that using two consecutive blocks with some constraints can generate two 1024-bit messages with identical hash digests - M_1 varies from M_2 ($M_1 \neq M_2$) but $MD5(M_1, IHV_i) = MD5(M_2, IHV_i)$.

MD5 structure has the property that if $MD5(x) = MD5(y)$ then $MD5(x||z) = MD5(y||z)$ where "||" denotes concatenation. According to this structure-property and combining the result provided by Wang, Dan Kaminsky in 2005 created *Stripwire* to mislead integrity checking, and so did Ondrej Mikle. This kind of attacks based on Wang's knowledge is *identical-prefix collision attack*. However, this type of attacks is quite limited as it requires that the initial IHV for those two consecutive blocks ought to be the same.

Later on, in 2007, Marc Stevens made great progress who removed that constraint - no requirement for identical prefix anymore - and he generated *chosen-prefix collision attacks* for MD5 which was also his main contribution. Without the restriction of the identical prefix, he applied this attack to forgery certificates such as rogue X.509 CA. X.509 was widely adopted to guarantee the security of the HTTPS website. The result reassured that MD5 could not be a secure certificating scheme anymore.

The following report will be generally divided into 4 sections. Firstly, it will detail the structure of the MD5 algorithm and give demos with both python and java versions. Following that, it will demonstrate the identical-prefix collision attack created by Wang et al. which is a brilliant attacking method different from the original methods based on brute-forcing or birthday-paradox and will show how this affects the secrecy of MD5. After that, it will present another elegant and advanced method - chosen-prefix collision attack with its application. Finally, it will give a summary that consists of the reflection for the project and suggestions for future work.

II. MD5 Message-Digest Algorithm

The MD5 message-digest algorithm, MD5 algorithm for short, takes the number of N 512 bits messages including padding and an initial 128 bits IHV as inputs to generate a 128 bits fingerprint. The whole algorithm process can be separated into three parts.

Part I: Initially, it uses method 1 padding - append a bit '1' to the message and then append many bits '0' - to make the length of the padded message become 448 (mod 512). After that, for the last remaining 64 bits, it appends the length of the original unpadded message. As a consequence, the message length is now $512N$ where N is an integer.

Part II: It will divide the padded message into N 512-bit segments, for each of which it will assign a block that takes a segment and a 128-bit IHV_{in} as inputs and output a 128-bit IHV_{out}. Except for the first block which IHV is set manually, the input IHV_{in} of a block is given by the output IHV_{out} from the former block. All the blocks are concatenated in serial like Fig. 1.

Part III: For each block, it will continue to divide the input 512-bit segment into 16 words (32 bits). It will run 4 rounds and each round consists of 16 steps; as a result, there are 64 steps in total. Each step includes some bit-wise operations and their values may vary among steps. One-step structure has been shown in Fig. 2. More details can be found in the appendix. In the final step of a block, it will add the current A', B', C', D' to the very beginning $IHV_{in} = A, B, C, D$ as

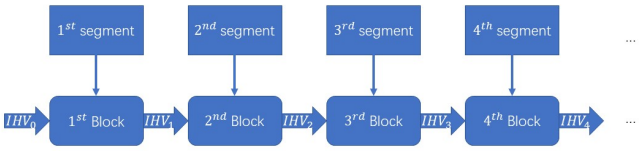

Fig. 1. Blocks Ordering

the output IHV_{out}. If the current block is not the last, it will propagate the output to the next block as the input for the next one; otherwise, it will output it as the final result.

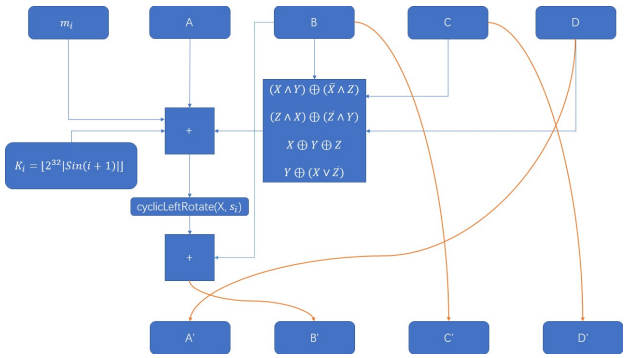

Fig. 2. MD5 Main Structure

For instance, the pre-image "TU Delft" is going to be compressed by MD5. Before being fed into the blocks directly, it needs to be encoded and its original length should be calculated before the padding.

$$Encode(\text{"}TUDelft\text{"}) = 54552044\ 656c6674\ _{hex}$$
$$|Encode(\text{"}TUDelft\text{"})| = 40\ _{hex}$$

Then it should be padded following the knowledge mentioned before. It should be noticed that the appending follows the *little-endian*.

$$54552044\ 656c6674\ 80000000\ 00000000$$
$$00000000\ 00000000\ 00000000\ 00000000$$
$$00000000\ 00000000\ 00000000\ 00000000$$
$$00000000\ 00000000\ 40000000\ 00000000$$

Since the total length after padding is merely 512 bits, it only needs one block to contain. After 64 steps computation, the 128 bits digest can be obtained. The code for the whole process is provided, either MD5.py or MD5.java.

$$467764e7\ f10c7769\ 5db73ab3\ d414c65d\ _{hex}$$

The MD5 algorithm invented by Ronald Rivest in 1992. Although it met some threatens in 1996, it had been used for approximately 12 years before Wang et al. found an effective way to find a collision. In the next section, it will show how Wang's elegant approach makes the MD5 algorithm become history.

III. Identical-Prefix Attack

The theorem behind the identical-prefix attack invented by Wang et al. is *differential path*. Taking two messages $M = \{m_0, m_1, ..., m_{n-1}\}$ and $M' = \{m'_0, m'_1, ..., m'_{n-1}\}$ $(\exists i \in [0, n)\ s.t.\ m_i \neq m'_i)$ and identical prefix $IHV_0 = IHV'_0$ as the initial inputs, the initial difference between IHVs is 0, denoted as $\Delta IHV_0 = 0$. When both propagate through MD5 blocks,

$$\Delta IHV_0 \overset{m_0, m'_0}{\longrightarrow} \Delta IHV_1 \overset{m_1, m'_1}{\longrightarrow} \Delta IHV_2 \rightarrow$$
$$... \rightarrow \Delta IHV_{n-1} \overset{m_{n-1}, m'_{n-1}}{\longrightarrow} \Delta IHV_n$$

at a position where $\Delta IHV_k = 0, (k \neq 0)$, from then on, all the remaining blocks' output will be identical. If any two messages found can generate this output, then the collision is found. Based on this differential path idea, Wang uses two consecutive blocks, fabulously, the latter of which can eliminate the differences caused by the former one, like Fig. 3 shown.

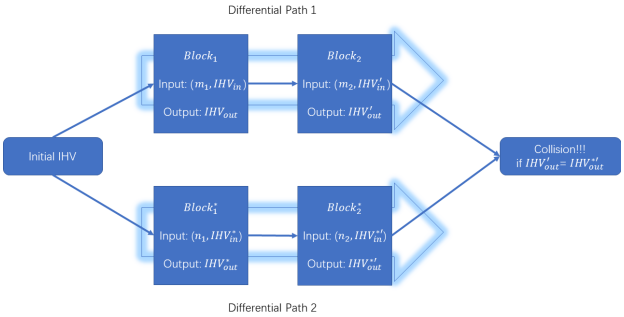

Fig. 3. Differential Paths

In order to use the latter block to cancel the variations caused by the former one, there are some restrictions for the two 1024-bit messages also with some sufficient conditions for the intermediate IVs - the IV represents the temporary output for each step. 1) For the first block two 512-bit messages, $\delta m_4 = 2^{31}$, $\delta m_{11} = 2^{15}$, $\delta m_{14} = 2^{31}$ and other $\delta m_i = 0$. 2) For the second block two 512-bit messages, $\delta m_4 = 2^{31}$, $\delta m_{11} = 2^{-15}$, $\delta m_{14} = 2^{31}$ and other $\delta m_i = 0$. 3) For the output IHV_1 and IHV'_1 of the first two blocks on different paths, it requires the differences $\delta IHV_1 = \{\delta IV_{1A}, \delta IV_{1B}, \delta IV_{1C}, \delta IV_{1D}\}$ that $\delta IV_{1A} = 2^{31}$, $\delta IV_{1B} = 2^{31} + 2^{25}$, $\delta IV_{1C} = 2^{31} + 2^{25}$ and $\delta IV_{1D} = 2^{31} + 2^{25}$. 4*) For the output IHV_2 and IHV'_2 of the second two blocks, $\delta IV_{2A} = 0$, $\delta IV_{2B} = 0$, $\delta IV_{2C} = 0$ and $\delta IV_{2D} = 0$. These four requirements are preliminary to lead the cancellation and find the *differential path* collision at the end. The conditions for each step's intermediate IV has been shown in the appendix.

The conditions mentioned above can increase the probability of finding a collision successfully at the end. Compared with the brute-force attack which result can only be verified at the final stage, the judgement of the sufficient condition during

the steps can determine if it needs to restart searching halfway when any of the conditions are not fulfilled.

identical-prefix attack application

Thereafter, people can apply this method to find as many collisions. However, Wang's method has two shortages. The first one is that the collision messages are randomly generated which are always meaningless ones. The second one is that the colliding messages must be generated simultaneously along differential paths.