Finding Hash Collisions Efficiently in MD5

CPE-579 Final Project

Peter Rauscher and Yu Wang

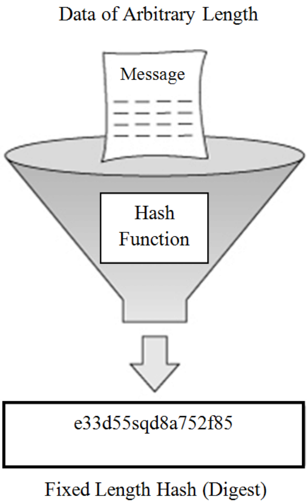
Stevens Institute of Technology

May 9, 2023

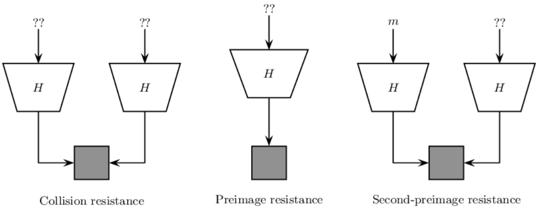
Contents

Overview of Message Digests

- Message digest algorithms (hash functions) are cryptographic procedures that map an arbitrary-length input to a fixed-length output
- The term "map" is important - the same input should always produce the same output
- A small change in the input data should result in a significant difference in the output (avalance effect)
- They have many practical applications:
  - Verifying message integrity
  - Server-side password storage
  - The "hash table" data structure
  - Fast detection of file changes - used in cloud storage solutions and backup managers



Measuring Security Strength



The security levels of different hash functions are evaluated on three of their fundamental properties:

- **Pre-image resistance**: Given a hash value h, it should be impractical to find the message m such that $hash(m) = h$ without a rainbow table
- **Second pre-image resistance**: Given a message m_1 it should be impractical to find a different message m_2 such that $hash(m_1) = hash(m_2)$
- **Collision resistance**: It should be impractical to find ANY two messages m_1 and m_2 such that $hash(m_1) = hash(m_2)$

How MD5 Works

- The input is padded to a bitlength l by adding a "1" bit followed by the least number of "0" bits needed until $l \equiv 448 \pmod{512}$
- The padded data is partitioned into N 512-bit blocks $M_1, M_2, ..., M_N$
- For $i = 1...N$, a compression function is used $md5comp(V_{i-1}, M_i)$. The function takes as input a vector of four 32-bit words $V_i = (A_i, B_i, C_i, D_i)$ and a message block M_i, where V_0 is publicly known in hexadecimal:

$$V_0 = (\text{01234567, 89abcdef, fedcba98, 76543210})$$

- The function $md5comp(V, M)$ goes as follows:
  For each $t = 0...63$, an Addition Constant AC_t, a Rotational Constant RC_t are defined as:

$$AC_t = \lfloor 2^{32} |\sin(t+1)| \rfloor$$

$$(RC_t, RC_{t+1}, RC_{t+2}, RC_{t+3}) = \begin{cases} (7,12,17,22) & \text{for } t = 0,4,8,12 \\ (5,9,14,20) & \text{for } t = 16,20,24,28 \\ (4,11,16,23) & \text{for } t = 32,36,40,44 \\ (6,10,15,21) & \text{for } t = 48,52,56,60 \end{cases}$$

How MD5 Works (Cont.)

- And a non-linear function f_t is defined as:

$$f_t = \begin{cases} F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z) & \text{for } 0 \leq t < 16 \\ G(X,Y,Z) = (X \wedge Y) \vee (Y \wedge \neg Z) & \text{for } 16 \leq t < 32 \\ H(X,Y,Z) = X \oplus Y \oplus Z & \text{for } 32 \leq t < 48 \\ I(X,Y,Z) = Y \oplus (X \vee \neg Z) & \text{for } 48 \leq t < 64 \end{cases}$$

- The message block M is split into sixteen 32-bit words $m_0, m_1 ... m_1 5$ and expanded to 64 words W_t:

$$W_t = \begin{cases} m_t & \text{for } 0 \leq t < 16 \\ m_{(1+5t)} \mod 16 & \text{for } 16 \leq t < 32 \\ m_{(5+3t)} \mod 16 & \text{for } 32 \leq t < 48 \\ m_{(7t)} \mod 16 & \text{for } 48 \leq t < 64 \end{cases}$$

How MD5 Works (Cont.)

- And another vector of four 32-bit words $(Q_t, Q_{t-1}, Q_{t-2}, Q_{t-3})$ is maintained for each step $t = 0..63$, with an initial state $(Q_0, Q_{-1}, Q_{-2}, Q_{-3}) = (b, c, d, a)$, and each new Q_{t+1} being calculated as:

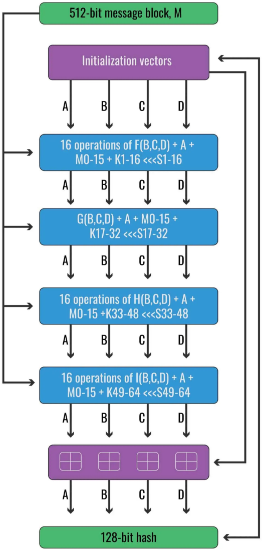
$$\begin{aligned} F_t &= f_t(Q_t, Q_{t-1}, Q_{t-2}) \\ T_t &= F_t + Q_{t-3} + AC_t + W_t \\ R_t &= RL(T_t, RC_t) \\ Q_{t+1} &= Q_t + R_t \end{aligned}$$

- The returned value of $md5comp$ is the addition of the resulting state words to the input vector:

$$md5comp(V, M) = (a + Q_{61}, b + Q_{64}, c + Q_{63}, d + Q_{62})$$

- The output is the concatenation $A_N + B_N + C_N + D_N$ after all runs of the compression function are complete.

MD5 Visualized

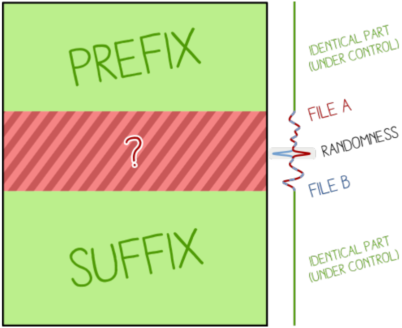


The MD5 Timeline

- In April 1992, MIT professor Ronald Rivest implements MD5 and recommends its adoption over MD2 and MD4 for its additional security measures. It is published in RFC 1321 [5].
- In 2007, security researchers were able to create a collision between a pair of X.509 certificates - used by the Certificate Authority VeriSign - which both appeared legitimate [4]. Shortly after, Rivest declares MD5 "cryptographically broken and unsuitable for further use" [1].
- In 2009, researchers Yu Sasaki and Kazumaro Aoki published a paper theorizing a pre-imaging attack on MD5 that can find a preimage in $2^{123.4}$ hash operations [6]. For reference, one of Hashcat's developers tested his rig on MD5 in 2016, with 8 x Nvidia GTX 1080 GPUs performing 200GH/s [3]. On his machine, the attack would take around 2.25×10^{18} years, about double the number of configurations for a 3x3 Rubik's cube.
- In 2012, another team of researchers published the nail in MD5's coffin: a fully automated program that could perform the *chosen-prefix collision attack*.

Identical Prefix Collision Attack

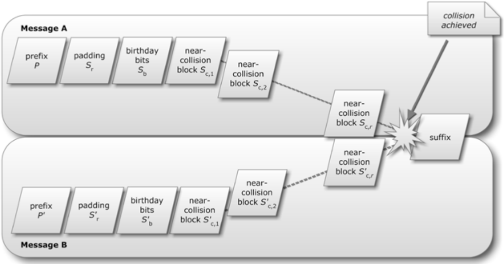
For any given input P, an attacker computes two messages M_1 and M_2 such that $hash(P + M_1) = hash(P + M_2)$ where the $+$ operation denotes concatenation. Any suffix may also be appended to both the files without changing the hash value.



However, this is difficult to exploit for real-world attacks.

The Chosen-Prefix Collision Attack

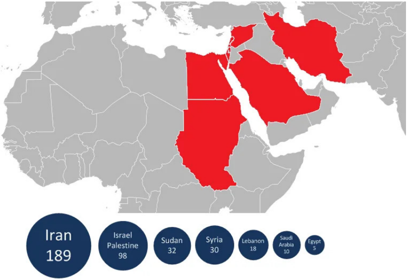
For arbitrary inputs P_1 and P_2, an attacker computes two messages M_1 and M_2 such that $hash(P_1 + M_1) = hash(P_2 + M_2)$ where $+$ is concatenation. Again, any suffix may be appened as well.



Though more costly, these attacks are **much** more exploitable. The difference in contents between files will no longer be completely random, but can be chosen by the attacker.

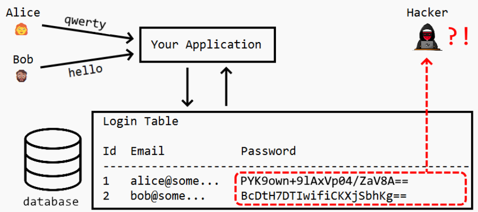
A Real-World Attack Using MD5 Weaknesses

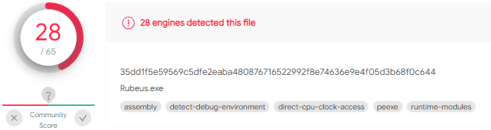
Flame/sKyWIper was a malware and spyware discovered by Iranian researchers in 2012 on over 1,000 Windows systems within Middle Eastern countries. Used for targeted espionage, though the country of origin is unconfirmed. Exploited a Microsoft Licensing Server using the chosen-prefix collision attack on MD5 [7] that allowed itself to share a code signature with legitimate Windows processes. Interestingly, the malware was found to have been in operation since 2010, meaning the malware's creators automated the chosen-prefix vulnerability themselves based solely on the proof-of-concept.



Common Usage of Hash

- To achieve security, database always store password hash instead of clear text passwords.
- Using hash to confirm a file's integrity
- Using hash as a fingerprint to identify a file





Security Considerations

- It is important to remember that **no** hash function can be completely collision-resistant, by definition (the pigeonhole principle). Secure functions will simply be well-designed enough to mitigate the *practicality* of attacks.
- MD5 is no longer considered cryptographically secure. It still has practical applications for error-checking and file-change monitoring, but should not be used beyond this.
- The recommended replacement for MD5 is SHA-256. It is the NIST's officially approved and standardized hashing algorithm.
- Unfortunately, as of 2019 nearly a quarter of SaaS platforms still use MD5 to store user's passwords [2].
- Demands for security audits, a push for frequent system updates, and raising public awareness of the issue is the only way to mitigate the threat legacy systems using MD5 still pose today.

References

[1] Cert/cc vulnerability note vu836068.

[2] Catalin Cimpanu. A quarter of major cmss use outdated md5 as the default password hashing scheme.

[3] epixoip. 8x nvidia gtx 1080 hashcat benchmarks, Jun 2016.

[4] Arjen Lenstra, Xiaoyun Wang, and Benne de Weger. Colliding x.509 certificates, Mar 2005.

[5] Ronald L. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, April 1992.

[6] Yu Sasaki and Kazumaro Aoki. Finding preimages in full md5 faster than exhaustive search. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, pages 134–152, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[7] Alex Sotirov. Analyzing the MD5 Collision in Flame. Trail of Bits, Jun 2012.