

Attacking MD5: Tunneling & Multi- Message Modification

Team Short Bus:

Daniel Liu
John Floren
Tim Sperr

- Introduction
- The MD5 Message Digest
 - Description of Algorithm
 - Our Implementation
- Description of Attacks:
 - Brute-Force/Birthday Attacks
 - Tunneling Attacks
 - Our Attacks vs. Reference Attacks
- Results
- Future Work
- Conclusion

Agenda

- Developed by Ron Rivest in 1991
- Described in Internet Standard RFC 1321 [1]
- Input: Any arbitrarily long message
- Output: 128-bit message digest
- Common uses:
 - Password storage
 - File integrity verification
 - Online certificates
- Now considered broken, and unsuitable for use

MD5: Message Digest Algorithm

MD5 processes its input in a series of steps: [1]

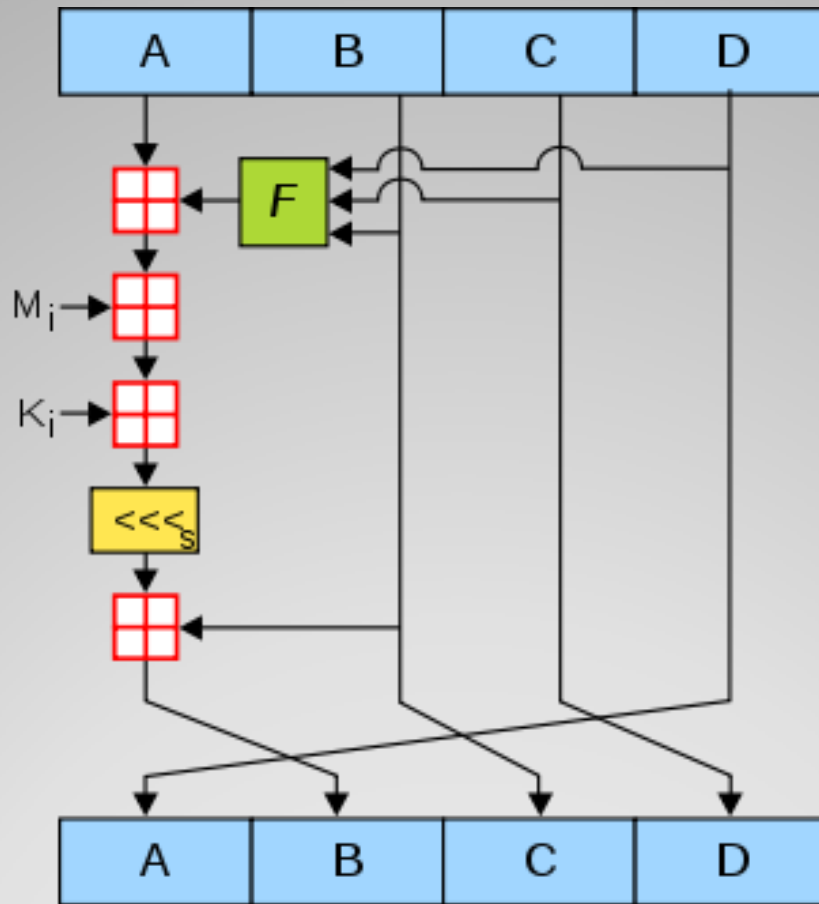
1. Pad the input
2. Length Extension
3. Split the input into 512-bit blocks
4. Process each block using four round functions
5. Mix the result of the rounds with the previous block's result (or with the IV)

MD5: How It Works

MD5 Round Operations:

- 4 rounds, 16 operations per round
- 64 total operations per message block
- All operations are performed on a buffer, (A, B, C, D) that stores intermediate results
- (A, B, C, D) is initialized using some IV

MD5: How It Works



Source: <http://en.wikipedia.org/wiki/Md5>

MD5: How It Works

- Written in the C programming language
- Used in the brute-force attacks
- Can perform a reduced number of rounds
 - Reducing word width was not an option with chosen attack
- Could not be integrated with the tunneling attack code due to differences in implementation

MD5: Our Implementation

- A brute-force attack on a hash function uses the birthday principle to obtain a collision in $2^{N/2}$ hashes, on average
- N is the number of bits of hash output
- For MD5, this number is 128, so 2^{64} hashes are necessary
- Assuming we can do 1 million hashes per second, this would still take almost 600,000 years...
- Attacking fewer rounds doesn't help, either.

Brute-Force Attack

Our Brute-Force attack:

- Tries to find two 512-bit messages that collide.
- Generates one 512-bit message randomly
- Uses a pseudorandom sequence to search other possible messages until a collision is found
- Takes waaaaaay too long.

Brute-Force Attack

- Sophisticated attack on MD5 (basis of Tunneling)
- Relies on choosing a set of values Q based on a set of "sufficient conditions."
- Q is a set of 64 values – one for each round operation of MD5
 - $Q[1]=IV[1]+RL(F(IV[1],IV[2],IV[3])+IV[0]+x[0]+0xd76aa478, 7)$
 - \dots
 - $Q[64]=Q[63]+RL(I(Q[63],Q[62],Q[61])+Q[60]+x[9]+0xeb86d391,21);$
 - Where:

$F(X,Y,Z) = X Y \text{ or } (\text{not}(X) Z)$	(round 1 function)
$G(X,Y,Z) = X Z \text{ or } (Y \text{ not}(Z))$	(round 2 function)
$H(X,Y,Z) = X \text{ xor } Y \text{ xor } Z$	(round 3 function)
$I(X,Y,Z) = Y \text{ xor } (X \text{ or } \text{not}(Z))$	(round 4 function)
- Sufficient conditions are designed such that
(where M and M* are single block (512-bit) messages, C1 and C3 are constants):
 - If $M - M^* = C1$ then
 - $MD5(M) - MD5(M^*) = C3$

Multi-Message Modification

```

bit position /   3332 2222 2222 2111 1111 111
                \   2109 8765 4321 0987 6543 2109 8765 4321

Q[ 1]           = .... ..
Q[ 2]           = .... ..
Q[ 3]           = .... ..vvv 0vvv vvvv 0vvv v0.. ..
Q[ 4]           = 1... .. 0^^^ 1^^^ ^^ ^^ 1^^^ ^011 ..
Q[ 5]           = 1000 10vv 0100 0000 0000 0000 0010 v1v1
Q[ 6]           = 0000 001^ 0111 1111 1011 1100 0100 ^0^1
Q[ 7]           = 0000 0011 1111 1110 1111 1000 0010 0000
Q[ 8]           = 0000 0001 1..1 0001 0.0v 0101 0100 0000
Q[ 9]           = 1111 1011 ...1 0000 0.1^ 1111 0011 1101
Q[10]           = 0111 ... 0001 1111 1v01 ...0 01.. ..00
Q[11]           = 0010 .v 111. 0001 1^00 . 0 11.. ..10
Q[12]           = 000. ..^ .. 1000 0001 ...1 0... ..
Q[13]           = 01.. ..01 ... 1111 111. ...0 0... 1...
Q[14]           = 000. ...00 ... 1011 111. ...1 1... 1...
Q[15]           = v110 0001 ..V. .... 10.. .... 0000 0000
Q[16]           = ^010 00.. ..A. .... v... .... 000 v000
Q[17]           = ^1v. .... ..0. ^... .. ^...
Q[18]           = ^.^ .. .. .1. .... ..
Q[19]           = ^... .. ..0. .... ..
Q[20]           = ^... .. ..v. .... ..
Q[21]           = ^... .. ..^ .. ..
Q[22]           = ^... .. .. .. ..
Q[23]           = 0... .. .. .. ..
Q[24]           = 1... .. .. .. ..

```

Note: ^ = a bit, equal to the bit above
v = a bit, equal to the bit below
V = a bit, equal to the negation of the bit below
A = a bit, equal to the negation of the bit above
. = an arbitrary bit
0/1 = a bit, which is fixed to 0 or 1

Extra conditions are highlighted:

Tunnel Q4 = **by this color**
Tunnel Q9 = **by this color**
Tunnel Q10 = **by this color**
Tunnel Q14 = **by this color**

Fig.2: The sufficient conditions and the extra conditions for the first block

[3] Vlastimil Klima, "Tunnels in Hash Functions: MD5 Collisions Within a Minute," Cryptography ePrint Archive, Report 2006/105, 2006.

Subset of sufficient conditions

- For 16 word (512-bits) messages if $C1[4] = 0x80000000$, $C[11] = 0x00008000$, $C[14] = 0x80000000$
- Differential Path: [3]
- $Q_M[1] - Q_{M^*}[1] = 0x00000000$
- $Q_M[2] - Q_{M^*}[2] = 0x00000000$
- $Q_M[3] - Q_{M^*}[3] = 0x00000000$
- $Q_M[4] - Q_{M^*}[4] = 0x00000000$
- $Q_M[5] - Q_{M^*}[5] = 0xFFFFFC0$
- $Q_M[6] - Q_{M^*}[6] = 0x807FFFC0$
- $Q_M[7] - Q_{M^*}[7] = 0xF87FFFBF$
- ...
- $Q_M[61] - Q_{M^*}[61] = 0x80000000 = IV_M[0] - IV_{M^*}[0]$
- $Q_M[62] - Q_{M^*}[62] = 0x82000000 = IV_M[1] - IV_{M^*}[1]$
- $Q_M[63] - Q_{M^*}[63] = 0x82000000 = IV_M[2] - IV_{M^*}[2]$
- $Q_M[64] - Q_{M^*}[64] = 0x82000000 = IV_M[3] - IV_{M^*}[3]$

Differential Path

- Now message's hash differ by a constant difference, but the goal is to make two messages with the same hash
- Add a second block N and N* to M and M*
- $X - X^* = (M, N) - (M^*, N^*) = (C1, C2)$
- Design sufficient conditions for N given an IV difference of:
 - $IV_M[0] - IV_{M^*}[0] = 0x80000000$
 - $IV_M[1] - IV_{M^*}[1] = 0x82000000$
 - $IV_M[2] - IV_{M^*}[2] = 0x82000000$
 - $IV_M[3] - IV_{M^*}[3] = 0x82000000$ ([3],[5])
- Such that the final difference in hashes is:
- $MD5(N) - MD5(N^*) = 0$ and thus $MD5(X) - MD5(X^*) = 0$
- The initial message difference constants (C1,C2) and all "sufficient conditions" were chosen based strong statistical and dependency analysis.

Differential Path

- $Q[1] - Q[24]$ could be found deterministically to satisfy all conditions
- Remaining $Q[25] - Q[64]$ can only be determined by probabilistic trial and error due to extremely complex relationships

1. Find arbitrary values for $Q[1] - Q[17]$ that satisfy conditions
2. Use inverse Q functions to find $M[0,1,2,4,5,6,10,11,15]$
3. Find remaining $Q[18] - Q[24]$
4. Calculate remaining message M and $Q[25] - Q[64]$ based on past values
5. If conditions don't match, restart with new arbitrary values

} Deterministic Conditions

- Instead, "tunnels" can be utilized to add more fixed conditions (reduce search size) or change the probability of the results of certain Q values (increase likelihood of meeting conditions) [2]
 - Take advantage of dependencies between all Q equations
 - **Challenge:** Hashes are designed such that these dependencies are complex

Tunneling

- Calculations for Q[9-12]

As given in [3]:

$Q[9] = Q[8] + \text{RL}(F(Q[8], Q[7], Q[6],) + Q[5] + x[8] + 0x698098d8, 7);$

$Q[10] = Q[9] + \text{RL}(F(Q[9], Q[8], Q[7],) + Q[6] + x[9] + 0x8b44f7af, 12);$

$Q[11] = Q[10] + \text{RL}(F(Q[10], Q[9], Q[8],) + Q[7] + x[10] + 0xffff5bb1, 17);$

$Q[12] = Q[11] + \text{RL}(F(Q[11], Q[10], Q[9],) + Q[8] + x[11] + 0x895cd7be, 22);$

$F(X, Y, Z) = (X \text{ and } Y) \text{ or } ((\text{not } X) \text{ and } Z)$

Tunneling: The Q9 Tunnel

- Q[10] depends on Q[9]; adjust x[9] to fix
- Q[13] will also be changed, adjust x[12]
- The changes to Q[9], x[8], x[9], and x[12] do not affect anything else before the Q[24]
 - Subsequent Q's are probabilistic
 - Rearrange Q[9]'s tunnel bits for different POV's
- In actuality, conditions on Q[9-11] mean that only 3 bits can be changed this way
- 2^3 different combinations

Tunneling: The Q9 Tunnel

- Klima found tunnels on $Q[4]$, $Q[9]$, $Q[10]$, $Q[13]$, $Q[14]$, and $Q[20]$
- These can be applied simultaneously to significantly reduce computation times

Multiple Tunnels

- Although simple in concept, implementing multi-message modification and tunneling was difficult
 - Could not find specifications of all conditions
 - Actual order in which to check and apply conditions unclear
 - Only one existing public implementation of full tunneling attack (Klima)

Tunneling: Implementation

- Attempted to design independent implementation of tunneling
 - Used published conditions and attempted to reverse-engineer other conditions from Klima's code
 - Complex inter-tunnel dependencies
 - New dependencies that are not documented
 - Poorly-formatted code base
 - Independent implementation was not successful.

Tunneling: Independent Attempt

- Eventually forced to use Klima's code directly to get some results
 - Code was cleaned up where possible
 - Modified to run for 2 rounds or the full 4 rounds
- Was not possible to run for 1 or 3 rounds due to design of differential path

Klima's Attack

- 2 round reduction was possible due to properties of differential path: [3]
 - $Q_M[29] - Q_{M^*}[29] = 0 = IV_M[0] - IV_{M^*}[0]$
 - $Q_M[30] - Q_{M^*}[30] = 0 = IV_M[1] - IV_{M^*}[1]$
 - $Q_M[31] - Q_{M^*}[31] = 0 = IV_M[2] - IV_{M^*}[2]$
 - $Q_M[32] - Q_{M^*}[32] = 0 = IV_M[3] - IV_{M^*}[3]$
- Which means the hashes of M and M* at step 32 are the same
- For 1 and 3 round reduction the hash difference of M and M* are new differences, so in order to cancel the hash value the second set of conditions for N and N* must be rediscovered.

Klima's Attack

- main() seeds prng, calls function to find block 1
- Block 1 function sets up deterministic values of Q and finds message block x
 - It then uses nested for() loops to iteratively check through every possible combination of tunnels
 - If the probabilistic conditions are met, it calls the block 2 function
- Block 2 function behaves much like block 1 function but with new conditions and initialization vectors
- If colliding messages are found, they are printed along with the elapsed time

Klima's Attack

Brute-Force Attack:

- No collisions found after ~ 24 hours, for a reduced number of rounds.
- Full MD5 brute-force would take the same amount of time if not longer.
- We would all most likely be dead before a single collision is found.

Results

Tunneling Attacks:

Hash Type	Run 1 time	Run 2 time	Run 3 time	Average time
Two-round	< 0.01 s	< 0.01 s	< 0.01 s	< 0.01 s
Full (four-round)	37 s	34 s	75 s	48.6 s

- It takes less than a minute to find a random collision for full MD5.
- Compare this to the ~ 8 hours taken using a standard multi-message modification attack. [5]
- Reduced-round attacks for 1 and 3 rounds could not be performed, because that would require changing the differential scheme.

Results

As of now, MD5 is officially considered cracked.

- “Rainbow tables” used to crack common passwords
- Programs exist that can find MD5 collisions for self-extracting archives

Tunneling, however, can be extended to other hash functions, such as SHA-1, SHA-2...

Future Work

- Tunneling is an effective modification of a sophisticated attack on MD5, and allows collisions to be found very quickly
- Additional work can be done to locate tunnels for MD5 and publishing the details of such attacks
- Tunneling with Multi-Message Modification can be implemented on SHA-0, SHA-1, and SHA-2 hashes, if the dependencies between steps can be analyzed to generate sufficient conditions and new tunnels

Conclusion

- [1] The MD5 Message-Digest Algorithm, International Engineering Task Force, RFC1321, April 1992, <http://www.ietf.org/rfc/rfc1321.txt>
- [2] Vlastimil Klima, "Finding MD5 Collisions – a Toy For a Notebook," Cryptography ePrint Archive, Report 2005/075, 2005.
- [3] Vlastimil Klima, "Tunnels in Hash Functions: MD5 Collisions Within a Minute," Cryptography ePrint Archive, Report 2006/105, 2006.
- [4] Wang Yu, Chen Jianhua, He Debiao, "A New Collision Attack on MD5," *Networks Security, Wireless Communications and Trusted Computing, International Conference on*, vol. 2, pp. 767-770, 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing, 2009.
- [5] Xiaoyun Wang and Hongbo Yu, "How to Break MD5 and Other Hash Functions," In *Advances in Cryptography - EUROCRYPT 2005*, pp.19-35, Springer-Verlag, May 2005.
- [6] Vlastimil Klima, "Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications," Cryptography ePrint Archive, Report 2005/102, 2005.

References

Questions?