# How to Break EAP-MD5

Fanbao Liu, Tao Xie



# How to Break EAP-MD5

Fanbao Liu and Tao Xie

School of Computer, National University of Defense Technology, Changsha, 410073, Hunan, P. R. China
liufanbao@gmail.com

**Abstract.** We propose an efficient attack to recover the passwords, used to authenticate the peer by EAP-MD5, in the IEEE 802.1X network. First, we recover the length of the used password through a method called length recovery attack by on-line queries. Second, we crack the known length password using a rainbow table pre-computed with a fixed challenge, which can be done efficiently with great probability through off-line computations. This kind of attack can also be implemented successfully even if the underlying hash function MD5 is replaced with SHA-1 or even SHA-512.

**Keywords:** EAP-MD5, IEEE 802.1X, Challenge and Response, Length Recovery, Password Cracking, Rainbow Table.

## 1 Introduction

IEEE 802.1X [6] is an IEEE Standard for port-based Network Access Control, which provides an authentication mechanism to devices wishing to attach to a Local Area Network (LAN) or Wireless Local Area Network (WLAN). IEEE 802.1X defines the encapsulation of the Extensible Authentication Protocol (EAP) [4] over IEEE 802 known as "EAP over LAN" (EAPoL). IEEE 802.1X authentication involves three parties: a peer, an authenticator and an authentication server. The peer is a client device that wishes to attach to the LAN/WLAN. The authenticator is a network device, such as a Wireless Access Point (WAP), and the authentication server is typically a host running software supporting the Remote Authentication Dial In User Service (RADIUS) and EAP protocols. EAP-MD5 [4], first used as Challenge-Handshake Authentication Protocol (CHAP) [13] in Point-to-Point Protocol (PPP) [12, 9] network for peer authentication, is a hash-based challenge and response protocol. EAP-MD5 applies the secret prefix approach of hashing [14] to generate response $R = \mathrm{MD5}(P||C)$, where $P$ is the shared password and $C$ is the challenge.

In EAP-MD5 based wireless network, the peer provides user name and the shared password, which is further hashed after appending the challenge sent by the authentication server, to the authenticator to get right to access the network. The authenticator forwards the peer's credentials to the authentication server, only if the server determines the peer's credentials are valid, and then the authenticator will allow the peer to access the network.

EAP-MD5 provides the peer authentication but without mutual authentication, it is prone to attacks as man-in-the-middle (MITM). The shared password between the peer and the authentication server is always chosen by the peer, who prefers to choose a simple and memorable one, as a result, EAP-MD5 is also vulnerable to dictionary attack just like other systems with human memorable passwords.

**Our contributions**. We firstly propose an efficient method to break EAP-MD5 by efficiently cracking the shared password.

Based on the length recovery attack to MD4 [15], we apply this kind of attack after minor modification to recover the length of password used in EAP-MD5, with which we can reduce most of the extra costs of length guessing during password cracking.

Further, based on the rainbow table and the chosen challenge attack, we generate a series of rainbow tables using a fixed challenge combined with all password candidates with fixed lengths.

We look up the received response from the corresponding rainbow table to recover the known length password.

This paper is organized as follows. We introduce some preliminaries and background in section two. We present some related work about EAP-MD5 and password cracking in second three. We introduce an efficient password cracking method to EAP-MD5 in detail in section four. We conclude the paper in the last section.

## 2 Preliminaries

### 2.1 Brief Description of MD5

MD5 [11, 17, 7, 8] is a typical Merkle-Damgård structure hash function, which takes a variable-length message $M$ as input and outputs a 128-bit hash value. $M$ is first padded to be multiples of 512 bits, a '1' added at the tail of $M$, followed by '0's until the bit length becomes 448 on modulo 512, finally, the length of the unpadded message $M$ is inserted to the last 64 bits. Let $t = |M'|/512$, the padded $M'$ is further divided into chunks of 512-bit blocks $(M_0, M_1, \ldots, M_{t-1})$.

The MD5 compression function ($CF$) takes $M_i$ and a 128-bit chaining variable $H_i$, initialized to $H_0$, as input, and outputs $H_{i+1}$. For example, $H_1 = CF_{H_0}(M_0)$, and the hash result of MD5 is computed as (1).

$$\mathrm{MD5}(M) = H_t = CF_{H_{t-1}}(M_{t-1}) \tag{1}$$

However, we omit the details of $CF$, since it is irrelevant to this paper, for more details please refer [11].

**Padding rule**. For two arbitrary distinct messages $M$ and $M'$, if their lengths $|M| = |M'|$, then, the padding bits of these two messages are just the same. Since MD5, MD4 and SHA-1 et al. share the same padding procedure, the padding rule is also applicable to them.

**Extension attack**. For an unknown message $M$ and the corresponding hash $R = \text{MD5}(M)$. With the known length of $M$, compute the padding bits $pad$ for $M$, then for arbitrary message $x$, compute the padding bits $pad1$ for $M||pad||x$, then we can generate the hash value for $M||pad||x$ from computing $CF_R(x||pad1)$, without any knowledge of $M$ except its length.

### 2.2 EAP-MD5

Extensible Authentication Protocol (EAP) [4] is an authentication framework frequently used in 802.1X wireless networks and PPP connections. EAP provides some common functions and negotiation of authentication methods called EAP methods, such as EAP-MD5.

EAP-MD5 [4] is analogous to CHAP [13]. EAP-MD5 differs from other EAP methods in that it only provides peer authentication.

The authentication of EAP-MD5 in 802.1X network involves three parts, the peer, the wireless access point (WAP, the authenticator) and the RADIUS server (the authentication server). The peer first sends an "EAPoL-start" packet to the WAP, and after then the WAP replies an "EAP-Request/ID" to ask for the peer's ID. After the WAP and RADIUS server both receive the peer's ID, the WAP forwards the EAP-MD5 random challenge string including a 1-byte $id$ and a variable length $C$, sent by the RADIUS server, to the peer. If the response $R$ sent by the peer is valid, then the peer is authorized to use the network, otherwise, reject. At random interval, the server redoes the authentication, by sending different challenges, to limit the time of exposure to any single attack.

The peer authentication process of EAP-MD5 in IEEE 802.1X is shown in Fig. 1.

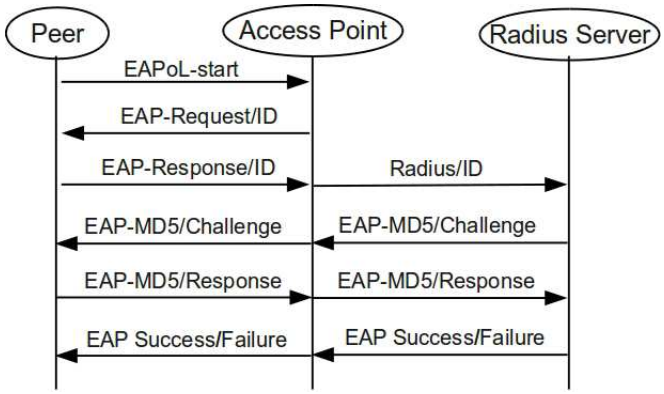


**Fig. 1.** The peer authentication of EAP-MD5

The response $R$ of EAP-MD5 is calculated with MD5 using a 1-byte session identifier $id$ and a challenge $C$ both sent by the authentication server, combined

with the shared password $P$. The calculation is shown in (2), where $||$ means concatenation.

$$R = \text{MD5}(id||P||C) \tag{2}$$

With randomly chosen $id$ and $C$, EAP-MD5 provides protection against playback attack launched by malicious peer.

## 3 Related Work

### 3.1 EAP-MD5-Pass Attack

In 2008, Wright demonstrated that a program utility eapmd5pass [16] could crack the passwords used in EAP-MD5 of the wireless network. The utility includes a eavesdropping and a dictionary attack tools against the EAP-MD5 protocol. However, the successful ratio of eapmd5pass totally depends on the used dictionary and the complexity of the password.

### 3.2 Rainbow Table

Rainbow tables contain a connection between a hash value and its corresponding password $P$, for passwords hashed by $H(P)$, where $H$ is a hash function. By having this connection pre-calculated for all possible hash values, a quick search through the table for a desired hash value can reveal the password.

In 1980, Hellman [5] proposed a time-memory trade-off when applying a cryptanalytic attack. Hellman used hash chains, which is time consuming, to decrease the memory requirements. In the way of hash chain, only the first and the last elements of the chain are stored in the memory, which saves memory at the cost of extra cryptanalysis time. Further, additional memory trade-off can be achieved by reducing the hash values to "keys" with the help of a reduction function. Instead of using a single reduction function $R$, Oechslin [10] suggested using a sequence of related reduction functions $R_1$ up to $R_{k-1}$, in 2003. Oechslin called the new chains rainbow chains, and the tables containing these chains rainbow tables. Since then, this method has been widely used and implemented in many popular password recovery tools [2, 3, 1], to break passwords hashed by the scheme of $h(P)$.

## 4 Breaking EAP-MD5

Since MD5 and MD4 share the same padding procedure, the length recovery attack to MD4($P||C$) [15] is also applicable to EAP-MD5. After the length of the used password is recovered, we search the corresponding rainbow table for the received response, which is generated by using a pre-chosen challenge. So, we divide the password cracking of EAP-MD5 into two parts, the first is to recover the length of the used password, and the second is to crack the password with already known length, using rainbow table technologies, based on fixed length passwords, and pre-chosen challenges.

### 4.1 On-Line Length Recovery Attack to EAP-MD5

In the wireless network, we can easily collect enough information, by eavesdropping, about the access point and the peer, such as a pair of ($C$, $R$), $id$, the MAC addresses of both, the user name and SSID of the access point, et al. We can generate faked packets, through masquerading as the access point, and send them to the peer; the peer will take it for granted that the packets are sent from the very access point, for he doesn't authenticate the access point. We omit the role of the authentication server, since it only interacts with the access point WAP.

We implement the length recovery attack to EAP-MD5 with pre-chosen challenge by masquerading as the access point (the authenticator), in the wireless network shown as follows.

1. Pre-choose a fixed $id$ and a fixed challenge $C$.
2. Eavesdrop from the network and find a communicating pair, a peer and a server (the authenticator, access point). Collect information about them.
3. Masquerade as the server, send the pre-chosen $id$ and $C$ to the peer. Get the corresponding response $R = \text{MD5}(id||P||C)$ from the peer, and reply an affirmative signal.
4. Initiate the length $l$ of the password $P$ with the guessing $l = 1$ character.
5. Generate the padding $pad$ for $id||P||C$ with the guessing length $l$ of the password.
6. Generate $C' = C||pad||r$, where $r$ is a randomly generated string, and send $C'$ with the known $id$ to the peer by masquerading as the access point.
7. Receive $R'$ from the peer, reply an affirmative signal. Generate $pad1$ for $C'$ and check if $R' = \text{CF}_R(r||pad1)$ holds. If so, $l$ is the very length. Otherwise, set $l = l + 1$, go to step 5.

**Packet Injection**. We send the faked packet with the challenge information $C$, by masquerading as the access point. Since the peer doesn't authenticate the access point $S$ in EAP-MD5, it will responds any packet from the pretended server $S'$. Both $S$ and $S'$ will receive response $R'$. Since $S$ doesn't send $C'$, it will silently discard the response $R'$. We let the pretending $S'$ reply an affirmative signal to the peer to end this verification session.

To recover the password length $l$, we need to send $l + 1$ faked packets with different challenges and $l + 1$ packets of affirmation. The time of the length recovery attack to EAP-MD5 is negligible, since the server has full control over the time intervals to send challenge to verify the peer, and the peer will reply any challenge at any time from the claimed server.

### 4.2 Off-Line Fixed Length Password Cracking with Password Pre-computation

**Passwords Pre-computation**. We can utilize the security hole that challenges can be sent by unauthenticated server, to launch a password pre-computation cracking. The overall strategy is shown as follows.

1. Passwords pre-computation.
   (a) We first choose a string, satisfying all the restrictions placed on the EAP-MD5 challenges, as a fixed challenge $C$. And choose a byte as the fixed $id$.
   (b) Set the length of password candidates be $l = 5$.[1]
   (c) Use the fixed challenge $C$ and the fixed $id$, we compute the corresponding hash values of all possible passwords $P$s with fixed length $l$ in the form of MD5($id||P||C$), and resort those results in a rainbow table[2].
   (d) Check if $l = 8$ holds, if so, go to step 2a. Otherwise, set $l = l + 1$, go to step 1c.
2. Break passwords in EAP-MD5.
   (a) Launch an on-line length recovery attack with the fixed challenge $C$ and $id$. We can immediately get the very length of the user's password, which is discussed in last subsection.
   (b) We look up in the corresponding pre-computed rainbow table, generated in the phase of passwords pre-computation, for the first received response $R$ replying from the peer, according to the already recovered password length. If it hits, we get the peer's very password.

Given a simple example of one password pre-computation attack to EAP-MD5, we choose a string "<jantie_lee@mukouren.com>" as a fixed challenge $C$. We also choose "x" as the fixed $id$. There are 95 printable characters to be used as password candidates in total, including digits, punctuations and letters with lower and upper cases. Then for a password with exact six characters, there may be $95^6 \approx 2^{39}$ password candidates. We compute the hash results of $R$=*MD5*(x$||P||$<*jantie_lee@mukouren.com*>), using all $2^{39}$ passwords with six characters in the brute force strategy, and get all corresponding "responses", eventually, then resort these results into a rainbow table.

Finally, we listen to the network and find a communicating pair. We impersonate the server and send the pre-chosen *id* "x" and the *challenge string* "<jantie_lee@mukouren.com>" to the peer, for recovering his password's length.

After the length of the peer's password is recovered, we look up in the corresponding rainbow table or a common table for the first received response. Obviously, if the peer uses a password with six characters, we can immediately get the very answer, for we have pre-computed all possible passwords.

**Response checking**. To check whether a given hash value is in the pre-computed rainbow table, we first generate a chain from this value. For each password that we encounter in the chain, we check to see if it is at the end of the chain. When we find a match, we know that the hash is part of this chain. Thus we generate the complete chain again and get the very password.

**The Rainbow Table**. We use the rainbow tables to pre-compute all hashes of the possible passwords with fixed lengths, based on the pre-chosen challenge

[1] We can directly generate a table mapping all possible candidates whose lengths are less than 5.

[2] We do not "store" their corresponding hash results, since those results are reduced by the reduction functions.

and $id$. Here, we implement the rainbow table of passwords with six characters, which takes about 3 days to be completed on an ordinary PC with Intel 2.G CPU. The details of complexity of such rainbow table are shown in Table 1, where $s$ means second. We also evaluate the overall complexity of generating rainbow tables of passwords with seven and eight characters, respectively. It seems infeasible to complete the rainbow table of passwords with eight characters on an ordinary PC. However, if utilize the parallelism computing with one thousand PCs, this work can be done soon. More important, an attacker of EAP-MD5 may only want to use the network as a legal user, hence, he needs to break a peer's passwords less than eight characters.

**Table 1.** The Complexity of Generating the Rainbow Table of MD5($id||P||C$)

| Char Set | Characters | Passwords Space | Rainbow Table Size | Hit Time |
|---|---|---|---|---|
| 95 | 8 | $95^8 \approx 2^{52}$ | 576 GB | 1108 $s$ |
| 95 | 7 | $95^7 \approx 2^{45}$ | 64 GB | 829 $s$ |
| 95 | 6 | $95^6 \approx 2^{39}$ | 9 GB | 18 $s$ |

**Overall complexity**. The overall complexity of such attack consists of three parts, the off-line pre-computation, the on-line query and the off-line table looking up. The time of the on-line query is negligible, since we have full control over the time intervals to challenge the peer. The time of the off-line table looking up varies from seconds to minutes. The complexity of the rainbow table pre-computation depends on the chosen length of the passwords and the pre-chosen challenge. Since EAP-MD5 employs human-memorable passwords, it means that we can compute all of the passwords within eight characters to cover most of the ordinary using passwords. Moreover, we point out that once the pre-computations are done, thus rainbow tables can always be reused for this kind of attack. In [16], to crack a password, the repeated dictionary attack must be performed each time. Hence, the attack in [16] is not a reusable one. More important, we will not try to crack passwords with more characters than these in our pre-computed rainbow tables, hence, our attack is a clever one.

Nowadays, the computation technology is dramatically improving, which includes the memory and storage enlargement, CPU processing speed increasing, and the network computing strengthening. Parallel computing allows anybody owning a modern personal computer in a network to break such cryptographic systems, which were believed to be secure.

## 5 Conclusion

In this paper, we propose an efficient and universal way to break EAP-MD5. We first recover the length of the used password by on-line queries, and then we apply the advantage of rainbow table to crack the known length password off-line. Though the 802.1X wireless network may not use this authentication

method any more, the operating systems supporting EAP-MD5 are vulnerable to this kind of attack. This attack is also applicable even the underlying hash function of EAP-MD5 is replaced with SHA-1 or even SHA-2, since these hash functions share the same padding rule and have the same weakness.

**Acknowledgement**

We thank the anonymous reviewers for their valuable comments. This work was partially supported by the program "Core Electronic Devices, High-end General Purpose Chips and Basic Software Products" in China (No. 2010ZX01037-001-001), and supported by the 973 program of China under contract 2007CB311202, and by National Science Foundation of China through the 61070228 project.

## References

1. OphCrack. http://ophcrack.sourceforge.net/
2. Rainbow Tables. http://rainbowtables.shmoo.com/
3. RainbowCrack. http://project-rainbowcrack.com/
4. Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., Levkowetz, H.: Extensible Authentication Protocol (EAP). RFC 3748 (Proposed Standard) (Jun 2004), http://www.ietf.org/rfc/rfc3748.txt, updated by RFC 5247
5. Hellman, M.: A cryptanalytic time-memory trade-off. IEEE Transactions on Information Theory 26(4), 401–406 (Jul 1980)
6. IEEE: IEEE 802.1x: IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control (2001)
7. Liu, F.: On the security of digest access authentication. In: Proc. -14th IEEE Int. Conf. on Computational Science and Engineering, CSE 2011 and 11th Int. Symp. on Pervasive Systems, Algorithms, and Networks, I-SPAN 2011 and 10th IEEE Int. Conf. on IUCC 2011. pp. 427–434 (2011)
8. Liu, F., Liu, Y., Xie, T.: Fast Password Recovery Attack: Application to APOP. Cryptology ePrint Archive, Report 2011/248 (2011), http://eprint.iacr.org/
9. Liu, F., Xie, T., Feng, Y., Feng, D.: On the Security of PPPoE Network. Security and Communication Networks pp. n/a–n/a (2012), http://dx.doi.org/10.1002/sec.512
10. Oechslin, P.: Making a faster cryptanalytic time-memory trade-off. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003, Lecture Notes in Computer Science, vol. 2729, pp. 617–630. Springer Berlin / Heidelberg (2003)
11. Rivest, R.: The MD5 Message-Digest Algorithm. RFC 1321 (Informational) (Apr 1992), http://www.ietf.org/rfc/rfc1321.txt, updated by RFC 6151
12. Simpson, W.: The Point-to-Point Protocol (PPP). RFC 1661 (Standard) (Jul 1994), http://www.ietf.org/rfc/rfc1661.txt, updated by RFC 2153
13. Simpson, W.: PPP Challenge Handshake Authentication Protocol (CHAP). RFC 1994 (Draft Standard) (Aug 1996), http://www.ietf.org/rfc/rfc1994.txt, updated by RFC 2484
14. Tsudik, G.: Message authentication with one-way hash functions. SIGCOMM Comput. Commun. Rev. 22, 29–38 (October 1992)

15. Wang, L., Ohta, K., Kunihiro, N.: Password recovery attack on authentication protocol MD4(Password||Challenge). In: Proceedings of the 2008 ACM symposium on Information, computer and communications security. pp. 3–9. ASIACCS '08, ACM, New York, NY, USA (2008)
16. Wright, J.: EAPMD5PASS-AUDITING EAP-MD5. http://www.willhackforsushi.com (2003)
17. Xie, T., Liu, F., Feng, D.: Could The 1-MSB Input Difference Be The Fastest Collision Attack For MD5?. Cryptology ePrint Archive, Report 2008/391 (2008), http://eprint.iacr.org/