

Implementation of Salted MD5 Hash in Ruby: A Security Analysis

Aulia Ichsan Rifkyano
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
rifkyano@gmail.com

Abstract—These days passwords are a kind of requirement almost everyone need. Passwords are like house keys, they can be used to unlock doors so that we are able to go somewhere else in the house. Even so, without using the key, the house can be broken into in many ways. It can be the fault of the key, the door, or the owner of the house itself. Same goes with passwords. If the owner of the password is not careful, people with bad intention might steal that password and use them to do some crimes. To increase security, one might do something to their password to make it difficult for others to steal or use them. One of them is called Hashing. By hashing a password, others might have to take more time than usual to guess or predict the original password. One might also add salt to the hash so that the password thief might not know the original password from the hashed one without the salt.

Keywords—Hash, salt, password, MD5, computer

I. INTRODUCTION

In this modern age where almost everything needs a computer, people realize that computers aren't as reliable without proper configurations. Computers can be used to help human calculate many things that can't be done in human time. But other than that, many people use computers for many different purpose, such as social media, job opportunities, promoting products, and many more. While computers seem to help human, some others believe that through computers too, other bad human might do bad stuff to others.

One example is that by using someone's account, with a password that has been compromised, they can do whatever they want with the identity of the stolen account's owner. This might seem harmless, but not when the owner has their account identities or other accounts such as bank accounts in their computers. This one is a huge problem since someone might lose what they have saved from many years in just a few minutes.

To tackle such problem, people have come up with many different ways to upgrade the security for their computers. One way is to fix and upgrade the passwords system. People tend to use the same password in many different computers and sometimes, that password might be a mediocre and mass used password, such as 'password' for the password. This might not be a problem if the world is a safe place and free from criminals, but the world isn't.

So to improve that password system, many people have come up with many different ideas. One being called hashing. Hashing is basically a function that takes a password and return it in another form and maybe length so

that other people might have to guess or even try harder to find the original password. Hashing is also popular in modern technology and many people have been researching over such topic, because even though it seems secure, some smart geniuses might also find holes in them so that they are breachable.

One hashing function that is popular today is MD5 hash. MD5 hash takes a password, do something to them in the middle of the process and return them in the form of a password that is 128 bits in length, or 16 bytes.

In some cases, people use hashing method to compare passwords whether they are the correct passwords or not in a database. If they are the same, meaning they are correct, then the person accessing the account will be granted access. If not, then the person might have to try again.

Other function that adds to security in passwords is salt. Salt is like another key to the same door that if you lose it, you will be locked forever. Having salt to many hashes seems to be a great idea since it is practical and easily implemented in everywhere. One problem that comes with salting a hash is having to maintain that salt or else the original password might always be unknown.

II. MD5 HASH

The MD5 Hash Algorithm has been in the cryptographic world for a while. Designed by Ron Rivest in 1991, it was originally made to replace its predecessor, MD4 Hash Algorithm. MD5 itself produces a hash of 128 bits long that is represented with a 32 digits of hexadecimal.

MD5 itself was popular until 2005 where it was found to be not resistant to collision. From here, people have tried to come up with a better alternatives to MD5. In 2012, the hash function was declared "cryptographically broken and unsuitable for further use".

To be a proper hashing function, the hash function must be collision resistant, meaning that at least two text should not make the same result. MD5 failed that and many people found a way to 'collide' the passwords just by using computers in their home. Since this exploit, people have been moving on to other hashing function such as SHA function families or others.

For now, MD5 is not usually found in cryptographic needs. It might be found in some websites, but not many. It still can be used to compare original files and the one being spread too. It is now mainly used to store passwords in websites with minimum specifications.

A. The Algorithm

1. Append padding bits, so that the length of the plaintext is congruent to 448 modulo 512. Even though the plaintext is already congruent to 448 modulo 512, padding should always be performed.

To append padding bits, a single '1' bit is appended then followed by '0' bits so that the length of the plaintext is congruent to 448 modulo 512.

2. Append length of the plaintext to the result of the previous step. From here, the result should be an exact multiple of 512 bits.
3. Initialize an MD buffer by making a four word buffer. For example, a four word (A,B,C,D) is used to compute the message digest. Each word is a 32-bit representation which then will be initialized to the following values in hex:

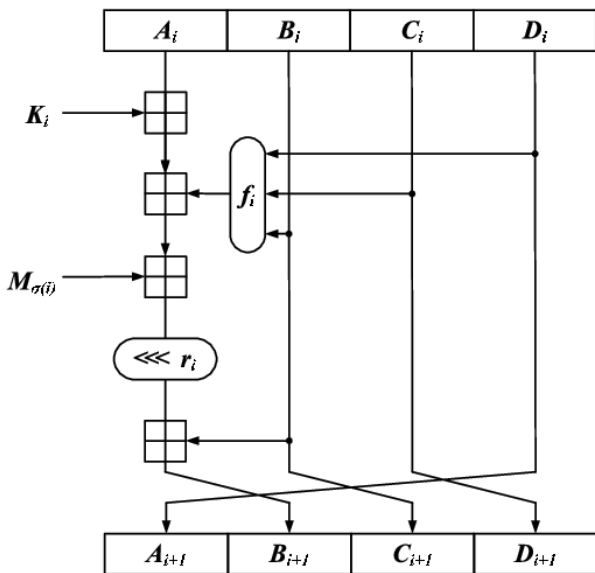
Word A: 01 23 45 67

Word B: 89 ab cd ef

Word C: fe dc ba 98

Word D: 76 54 32 10

4. Process result in a block of 16 words or 512 bits. First divide each result consisting of L blocks until the length becomes 512 bits. They are processed in such way like in this schema:



Drawing 1: Schema of MD5 Hash

B. MD5 Collision

The first MD5 collision was found in 1996. It was found in a lite version of MD5 function. Since then people have been figuring out whether MD5 contains another breach or not.

Then in 2005, Ron Rivest announced that the function was 'broken' and not to be used anymore after some cryptanalyst found them in a collision state. Not long after that, in 2007, another researcher named Anton Kuznetov

found out a breach in MD5 by using what is called as Prefix Collision, by appending two new value that has been hashed to two different documents. The two documents then produced the same result.

The infamous MD5 collision is:

d131dd02c5e6eec4 693d9a0698aff95c 2fcab58712467eab
4004583eb8fb7f89

55ad340609f4b302 83e488832571415a 085125e8f7cdc99f
d91dbdf280373c5b

d8823e3156348f5b ae6dacd436c919c6 dd53e2b487da03fd
02396306d248cda0

e99f33420f577ee8 ce54b67080a80d1e c69821bcb6a88393
96f9652b6ff72a70

And

d131dd02c5e6eec4 693d9a0698aff95c 2fcab50712467eab
4004583eb8fb7f89

55ad340609f4b302 83e4888325f1415a 085125e8f7cdc99f
d91dbd7280373c5b

d8823e3156348f5b ae6dacd436c919c6 dd53e23487da03fd
02396306d248cda0

e99f33420f577ee8 ce54b67080280d1e c69821bcb6a88393
96f965ab6ff72a70

Both producing the same MD5 hash:

79054025255fb1a26e4bc422aef54eb4

With that being one of the classic MD5 hash, now people can create their own collision using HashClash. With the cost of \$0.65, you can make your own collision of MD5 hash.

III. BREAKING THE DOOR

There are many ways to gain access to someone's account without getting the original password or by finding the original password. These are some famous ways of cracking passwords.

A. Brute Force

Brute force is the easiest and simplest way of cracking passwords, but takes the longest time to finish. To brute force, someone would use all combination of character available with the help of some hints available. For example, if someone's password is longer than 5 characters, then the cracker would have to use all combination of at least 5 characters to brute force the password.

Brute force will always be successful in predicting and finding all kind of password, it just takes almost forever and a lot of resource needed until the program finish brute forcing.

Some people already got a measurement for brute force, which is to limit the amount of password tries within a given amount of time. If someone fails more than the threshold given, then the cracker will be banned according to the amount of time specified.

B. Dictionary Attack

Dictionary Attack is basically brute force attack, but using more organized way. The cracker would need to set up

a list or dictionary of words that has the possibility to match with the password. Then the computer will try one by one until the match is found (or not).

Dictionary attack will not always be successful like brute force, but with a set of knowledge and some hints, someone might get a lucky charm with the password cracking.

C. Precomputed Table

Precomputed Table is just a database full of hashed text. Then if the cracker gains access to the database needed, the cracker would then compare the database passwords to his own precomputed table of hashes and find which one is the same.

The problem with precomputed table is that one has to have a large memory to store all the data of the precomputed hashes if they want to succeed. If someone's password is 10 letter long and consists of alphanumeric, then the cracker would need 36^{10} combination in which they are the combination of all the precomputed hashes.

IV. IMPROVING MD5

There are two ways that I am proposing to improve this hash function:

A. Repeated Hashing

By hashing many time, the result of the hash should be a random combination of letters and numbers. Then it would be another chaotic combination of letters and numbers if hashed again. Many websites have used this method by repeating MD5 three times. This would make people that has precomputed table need to have more effort on making another table.

The downside is that by calculating many times, a computer needs more CPU power to do the repeating. Many server would choose not to do that, but other servers would do the triple calculations in the client-side.

B. Salt

Salt is basically another plaintext that is added to the process of hashing. By adding salt, the original password is more obfuscated than before. The result of the hash would be much different than before by adding salt in the middle of the process. Salt is mainly aimed to reduce the efficiency of precomputed table, but then has proven to be effective in many cases of hashing. To crack the original password with salt, firstly the cracker would need to know the salt. Without the salt, the cracker have no way to find the original password.

V. IMPLEMENTATION

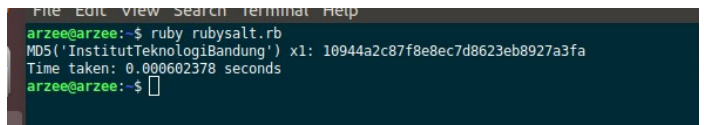
The implementation is Ruby based and follows the specifications specified by RSA. Multiple times hash would be as follows:

`HASH(HASH(HASH(HASH(PLAINTEXT))))`

While salt is just a random combination of alphanumeric word that is appended to the end of the original plaintext

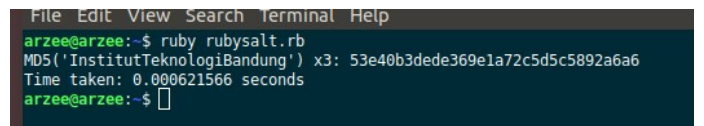
`HASH(PLAINTEXT+SALT)`

Here are some output from the program:



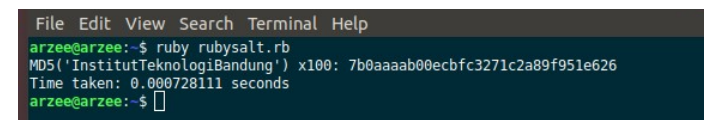
```
File Edit View Search Terminal Help
arzee@arzee:~$ ruby rubysalt.rb
MD5('InstitutTeknologiBandung') x1: 10944a2c87f8e8ec7d8623eb8927a3fa
Time taken: 0.000602378 seconds
arzee@arzee:~$
```

Drawing 2: MD5 Hash with no repeat



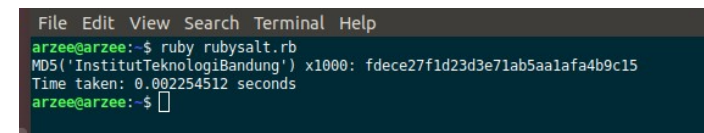
```
File Edit View Search Terminal Help
arzee@arzee:~$ ruby rubysalt.rb
MD5('InstitutTeknologiBandung') x3: 53e40b3dede369e1a72c5d5c5892a6a6
Time taken: 0.000621566 seconds
arzee@arzee:~$
```

Drawing 3: MD5 repeated 3 times of the same input



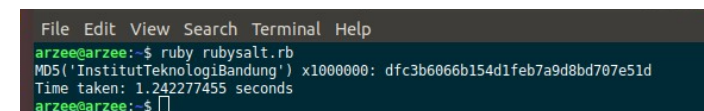
```
File Edit View Search Terminal Help
arzee@arzee:~$ ruby rubysalt.rb
MD5('InstitutTeknologiBandung') x100: 7b0aaaab00ecbfc3271c2a89f951e626
Time taken: 0.000728111 seconds
arzee@arzee:~$
```

Drawing 4: MD5 repeated 100 times of the same input



```
File Edit View Search Terminal Help
arzee@arzee:~$ ruby rubysalt.rb
MD5('InstitutTeknologiBandung') x1000: fdece27f1d23d3e71ab5aa1afa4b9c15
Time taken: 0.002254512 seconds
arzee@arzee:~$
```

Drawing 5: MD5 repeated 1,000 times of the same input



```
File Edit View Search Terminal Help
arzee@arzee:~$ ruby rubysalt.rb
MD5('InstitutTeknologiBandung') x1000000: dfc3b6066b154d1feb7a9d8bd707e51d
Time taken: 1.242277455 seconds
arzee@arzee:~$
```

Drawing 6: MD5 repeated 1 million times of the same input

```
File Edit View Search Terminal Help
arzee@arzee:~$ ruby rubysalt.rb
MD5('Apple') x1: 9f6290f4436e5a2351f12e03b6433c3c
Time taken: 0.000647727 seconds
arzee@arzee:~$
```

Drawing 7: A common word hashed 1 time

```
File Edit View Search Terminal Help
arzee@arzee:~$ ruby rubysalt.rb
MD5('Apple') x1000: 79721599d824ec8c189b77d5c6ee8eb6
Time taken: 0.002258516 seconds
arzee@arzee:~$
```

Drawing 8: The same common word hashed one thousand times

DISCLAIMER

With this I declare that my paper is my own writing, not a copy, not a translation, nor a plagiarism from other's work.

Bandung, 10 Mei 2019



Aulia Ichsan Rifkyano
13515100

Then they are compared in <https://crackstation.net/> and can be seen in the attachment section.

```
File Edit View Search Terminal Help
arzee@arzee:~$ ruby rubysalt.rb
--WITHOUT SALT--
MD5('Ganesha') x100: 3acebadb53a5430ffbd992e92414e129
Time taken: 0.00078956 seconds
---WITH SALT---
Salt: 16d14fa89
MD5('Ganesha16d14fa89') x100: 8afcc93f406b30a1f134467e17ce8d58
Time taken: 0.000158294 seconds
arzee@arzee:~$
```

Drawing 9: MD5 hash with salt

By adding salt, the execution time is also lower because longer plaintext means that the program doesn't have to process the padding byte many times, saving some processor power. The implementation of ruby MD5 hash with salt can be found in <http://github.com/arzeeee/rubysalt>

VI. SUMMARY

The MD5 Hash Algorithm has been in the cryptographic world for a while. Proven to be quick once doesn't mean it will be in the cryptographic world forever. Many breaches made the hash function unusable for many purpose, but yet it still can be used for some reason.

By using salt and repeated hashing, we can improve the security of passwords by a lot. Even though MD5 is popular with collision nowadays, we might still find them in some websites since they are pretty quick and use less computing power than many other hashing functions.

Attachments

https://crackstation.net

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
9f6290f4436e5a2351f12e03b6433c3c
79721599d824ec8c189b77d5c6ee8eb6
```

I'm not a robot reCAPTCHA

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
9f6290f4436e5a2351f12e03b6433c3c	md5	Apple
79721599d824ec8c189b77d5c6ee8eb6	Unknown	Not found.

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

Attachment 1: Result of repeated and non-repeated hashing of a common word