Enhanced Dictionary Based Rainbow Table

Vrizlynn L.L. Thing and Hwei-Ming Ying

Institute for Infocomm Research, Singapore
{vriz,hmying}@i2r.a-star.edu.sg

Abstract. As users become increasingly aware of the need to adopt strong password, it brings challenges to digital forensics investigators due to the password protection of potential evidentiary data. On the other hand, due to human nature and their tendency to select memorable passwords, which compromises security for convenience, users may select strong passwords by considering a permutation of dictionary words. In this paper, we discuss the existing password recovery methods and briefly present our previous work on the design of a time-memory tradeoff pre-computed table (Enhanced Rainbow Table) for efficient random password recovery. We then propose the design of an Enhanced Dictionary Based Rainbow Table to integrate the construction of dictionary based permutated passwords and common passwords within the Enhanced Rainbow Table, to incorporate the two promising password recovery approaches. We then present the analysis of the proposed method.

Keywords: digital forensics, password recovery, rainbow table, cryptanalysis.

1 Introduction

Being the most common authentication method, passwords are widely used to protect valuable data and to ensure a secured access to systems/machines. However, the use of password protection presents a challenge for investigators while conducting digital forensics examinations.

In some cases, compelling a suspect to surrender his password would force him to produce evidence that could be used to incriminate him, thereby violating his right against self-incrimination. Therefore, this presents a problem for the authorities. It is then necessary to have the capability to access a suspects data without expecting his assistance.

While there exist methods to decode hashes to reveal passwords used to protect potential evidence, lengthier passwords with larger characters sets have been encouraged to thwart password recovery. Awareness of the need to use stronger passwords and active adoption have also rendered many existing password recovery tools inefficient or even ineffective.

The more common methods of password recovery techniques are based on brute force, dictionary attack, breaking hashing algorithms and rainbow tables.

D. Gritzalis, S. Furnell, and M. Theoharidou (Eds.): SEC 2012, IFIP AICT 376, pp. 513–524, 2012.

In the brute force attack, every possible combination of the password characters in the password space is attempted for a match search. It is an extremely time consuming process. However, due to its exhaustive generation and search, the password will be recovered eventually if sufficient time is given. Cain and Abel (Cain and Abel, 2011), John the Ripper (John The Ripper, 2011) and LCP (LCPSoft, 2011) are popular tools that support brute force attacks.

The dictionary attack method involves loading a file of dictionary words (and performing permutation optionally) into a password recovery tool to search for a match of their hash values with the stored one. If the password is not a dictionary or permutated dictionary word, the recovery would fail.

Research attempting to discover and identify the weaknesses of hashing algorithms have also been useful in passwords or encryption keys recovery. This method is based on the collision of hashes in specific hashing algorithms (Contini, 2006; Fouque, 2007; Sasaki, 2007; Sasaki, 2008). However, they are highly complex and time consuming for use during time-critical forensics investigations. The methods are only applicable to specific hashing algorithms.

The time-memory tradeoff method (Hellman, 1980) is a hybrid of brute force attack and precomputed tables. A large number of passwords are repeatedly hashed and reduced to form password chains. Only the head and tail of these chains are stored. During recovery, the password hash goes through a series of reduction and hashing until a match with one of the stored tails is found. Passwords encrypted with hashing algorithms such as LM or NTLM used for Windows login (Todorov, 2007), MD5 (Rivest, 1992), SHA-2 (NIST, 2002) and RIPEMD-160 (Dobbertin, 1996) are susceptible to this recovery method.

The rainbow table method (Oechslin, 2003; Thing, 2009; Weir 2009; Ying, 2011) is similar to, and falls under the class of time-memory tradeoff method. The difference is that different reduction functions are used at each step of the chain generation, so as to minimise the collision of merging chains. Therefore, the success rate of password recovery can be higher.

In this paper, we first present our time-memory tradeoff rainbow table method, the Enhanced Rainbow Table, which shows promising performance and results (that is, in terms of success rate and recovery speed) during password recovery compared to existing work. We then propose the design of a novel Enhanced Dictionary Based Rainbow Table by including the generation of permutated dictionary words in the algorithm. We then analyse the proposed method.

2 Background

The idea of a general time-memory tradeoff was first proposed by Hellman in 1980 (Hellman, 1980). In the context of password recovery, we describe the Hellman algorithm as follows.

We let X be the plaintext password and Y be the corresponding stored hash value of X. Given Y, we need to find X, which satisfies h(X) = Y, where h is a known hash function. However, finding X = h^{-1}(Y) is feasibly impossible since hashes are computed using one-way functions, where the reversal function,

h^{-1}, is unknown. Hellman suggested taking the plaintext values and applying alternate hashing and reducing, to generate a pre-computed table.

For example, the corresponding 128-bit hash value for a 7-character password (composed from a character set of English alphabets), is obtained by performing the password hashing function on the password. With a reduction function such as $H \ mod \ 26^7$, where H is the hash value converted to its decimal form, the resulting values are distributed in a best-effort uniform manner. For example, if we start with the initial plaintext value of “abcdefg” and upon hashing, we get a binary output of 0000000....000010000000....01, which is 64 ‘0’s and a ‘1’ followed by 62 ‘0’s and a ‘1’. $H = 2^{63} + 1 = 9223372036854775809$. The reduction function will then convert this value to “3665127553”, which corresponds to a plaintext representation “lwmkgij”, computed from $(11(26^6) + 22(26^5) + 12(26^4) + 10(25^3) + 6(26^2) + 8(26^1) + 9(26^0)$.

After a pre-defined number of rounds of hashing and reducing, only the initial and final plaintext values (i.e. “head” and “tail” of the chains) are stored. Using different initial plaintexts, the hashing and reducing operations are repeated, to generate a larger table (of increasing rows/chains). A larger table will theoretically contain more pre-computed values (i.e., disregarding hash collisions), thereby increasing the success rate of password recovery, while taking up more storage space. The pre-defined number of rounds of hashing and reducing will also increase the success rate by increasing the length of the “virtual” chain, while bringing about a higher computational overhead.

To recover a plaintext from a given hash, a reduction operation is performed on the hash and a search for the computed plaintext among the final values in the table is conducted. If a match is not found, the hashing, reducing and searching operations are repeated. The maximum possible rounds of operations is determined by the chain length. If the hash value is found in a particular chain, the values in the chain are then worked out by performing the hashing and reducing functions to arrive at the plaintext giving the specific hash value.

Unfortunately, there is a likelihood that chains with different initial values may merge due to collisions. These merges will reduce the number of distinct hash values in the chains and diminish the recovery success rate. The success rate can be increased by using multiple tables with each table using a different reduction function. If we let P(t) be the success rate of using t tables, then $P(t) = 1 - (1 - P(1))^t$, which is an increasing function of t since P(1) is between 0 and 1. Hence, introducing more tables increase the success rate but also cause an increase in the computational complexity and storage space.

In (Denning, 1982), Rivest suggested a method of using distinguished points as end points for chains. Distinguished points are keys, which satisfy a given criteria, e.g., the first or last q bits are all 0. The chains are not generated with a fixed length but they terminate upon reaching pre-defined distinguished points. This method decreases the number of memory lookups compared to Hellman’s method and is capable of loop detection. If a distinguished point is not obtained after a finite number of operations, the chain is suspected to contain a loop and is discarded. Therefore, the generated chains are free of loops. One limitation is that the chains will merge if there is a collision within the same table. The variable lengths of the chains will also result in an increase in false alarms. Additional computations are incurred to detect false alarm occurrences.

(Oechslin, 2003) introduced a new table structure to reduce the probability of merging occurrences. The rainbow chains use multiple reduction functions so merges occur only if collisions happen at the same positions in different chains. Oechslin showed that the coverage in a single rainbow table is 78.8% compared to 75.8% in the classical tables of Hellman (with distinguished points).

(Weir, 2009) integrated dictionary attacks with the original rainbow table (Oechslin, 2003) to generate virtual chains of passwords consisting of dictionary words. This method improved the efficiency of dictionary attacks by utilizing the rainbow table. However, the table does not contain randomly generated passwords and can only be used for dictionary password recovery.

3 Enhanced Dictionary Based Rainbow Table

The key objectives in enhancing password recovery is to meet the increasing challenges of strong password-protected evidentiary data. Utilizing the hybrid approach of the brute force technique and precomputed table approach proves to be a cost-efficient way to recover password. Therefore, to improve password recovery performance, further research to increase the success rate and reducing the recovery time by rainbow tables is needed while taking into consideration stronger passwords adopted by common users.

On the other hand, it has been shown that humans are tempted to choose passwords which are easy for them to remember (Google News, 2009; Narayanan, 2005). Such passwords could be based on a combination of common key sequences on the keyboard layout, dictionary words, and a combination of dictionary words. Another common approach to strengthen the passwords while maintain their memorability is to include numbers and special characters in the passwords. Therefore, by taking into consideration the human nature and their tendency in password selections, and incorporating such knowledge into the design of a new password recovery method would improve performance significantly.

In the following sub-sections, we briefly describe our recent work on the enhancement of rainbow tables (Thing, 2009; Ying, 2011). Next, we propose a new approach of integrating memorable passwords with the Enhanced Rainbow Tables by considering the unique features of these new tables. We then present the analysis of this new Enhanced Dictionary Based Rainbow Table method.

3.1 Enhanced Rainbow Table

In (Thing, 2009; Ying, 2011), we proposed an Enhanced Rainbow Table design with a novel sorting algorithm. The first novelty lies in the chains generation technique. Instead of taking a large set of plaintexts as the initial values, we systematically choose a much smaller unique set. We choose a plaintext and compute its corresponding hash value. We let the resulting hash value be H.

Following that, we compute *(H+1) mod* 2^j*, (H+2) mod* 2^j*,......, (H+k) mod* 2^j for a variable k, where j is the number of bits of the hash output value (e.g. in MD5 hash, j = 128). These hash values are the branches of the above chosen initial plaintext. We then apply alternate hashing and reducing operations to all these branches. The resulting extended chain of branches is a block. Only the final values of the plaintexts in each block are stored with one initial plaintext value, instead of storing all the final values with the corresponding initial values in the original rainbow table, resulting in significant storage space conservation (or success rate improvement if the same storage space is provided).

As the "tail" passwords cannot be sorted now, since in doing so, the information of its corresponding initial hash value (which is not stored) will be lost, a novel sorting algorithm (Ying, 2011) was proposed so that the password lookup in the stored tables can be optimized. The use of special characters which are the non-printable ASCII characters, was proposed for the Enhanced Rainbow Table sorting. There are a total of 161 such characters and we assume that these non-printable ASCII characters do not form any of the character set of the passwords since they are not found on the keyboard. We insert a number of these special characters into the stored "tail" passwords. The way in which these special characters are inserted provides information on the original position of the passwords after the table has been sorted. The consequence is that these inserted special characters will incur storage space. We illustrated in (Ying, 2011) that the increase in storage space is minimal and is also significantly lesser than the original rainbow tables storage requirement. The advantage of this sorting algorithm is that the passwords in the table can now be sorted and thus a password lookup can be optimized. The sufficiency of the available special characters for use in sorting, the storage requirements, and the success rate of password recovery were also evaluated in (Ying, 2011). Maintaining the same storage space requirements for both methods, the Enhanced Rainbow Table is able to achieve an improvement of up to 26.13% and 23.60%, for the recovery of alpha-numeric passwords and passwords containing any of the printable ASCII characters, respectively, over the original rainbow table.

Next, we propose the integration of permutated dictionary attack within the Enhanced Rainbow Table to take into consideration the human tendency to select memorable passwords.

3.2 Design of Dictionary Based Enhanced Rainbow Table

In the Enhanced Rainbow Table, passwords were generated based on the hashing and reduction functions. Therefore, the generation is very random and a large percentage of passwords may contain special characters at random places and non-dictionary words. Such passwords have a very low memorability level and users who chose such passwords usually create the passwords using strong password creation tools. Most users also need to note down the passwords and store them separately to prevent them from forgetting their own passwords for subsequent accesses. However, users tend to want to avoid the trouble of choosing a password of such high complexity and worry about forgetting it later. Therefore, they usually try to choose passwords which are memorable. These passwords are usually dictionary words, common keyboard layout key sequences or information related to themselves (for example, their name, spouse's name or birthdays).

A simple approach is to conduct a dictionary attack first and then the rainbow table password recovery if the dictionary attack fails. However, both the computational overhead and storage requirement will be high. Instead, in this paper, we propose a novel approach to incorporate common passwords into the Enhanced Rainbow Table so that the percentage of common passwords can be higher resulting in a higher success rate even in the scenario whereby the length of the passwords is large and the storage capacity is limited. The aim of the integration of permuted dictionary and the Enhanced Rainbow Table is to construct the table where by it can contain as many permutated dictionary words as possible (in both stored elements and the virtual chains).

The simplest method to create the Enhanced Dictionary Based Rainbow Table is to generate permutated dictionary words as the initial column of passwords. However, this is only applicable for the case of the original rainbow table as the initial column is discarded in the Enhanced Rainbow Table.

Instead, we propose to have the first reduction function generate permutated dictionary words in the first virtual column, which in the Enhanced Rainbow Table, will be recoverable. However, in this case, the possible number of "initially generated" permutated dictionary words is limited by the number of chains in the table. The recovery speed may also suffer for such passwords as they fall under the first virtual column. Therefore, next, we additionally propose constructing some chains where all the entries are common passwords.

Suppose we identify x number of common passwords used. We want to ensure that these x passwords can be generated from a rainbow chain. One way to do this is starting with any password, hash and choose an appropriate reduction function R_1 which reduces to one of the x number of words in the list. Continuing with the chain generation, hash this resultant and choose an appropriate reduction function R_2 which again reduces to another one of the words in the list. Continue doing this until all the words in the list is generated in the chain as required. As long as x is not too large relative to the keyspace, we will always almost certainly be able to choose such R_i and thus, generate such a chain which contains all the words in the list. As for the remaining chains, there is no certainty that dictionary words can be generated accordingly in the same manner since the outputs of hash functions tend to be random. The advantage is that this method will be able to recover both common and random passwords.

For example, suppose we want to consider 7-character passwords (consisting of lower case alphabets, hashed by MD5). Given that "letmein", "abcdefg" and "testing" are three common 7-character passwords, the goal is to include them in the rainbow chains.

Starting with the password "testing", upon hashing, its hashed value is H_1 = ae2b1fca515949e5d54fb22b8ed95575. This value is then converted to its decimal representation and the reduction function is applied where $r_1(H_1) = H_1 + 4938209469 \bmod 26^7$. Converting $r_1(H_1)$ back to its password representation results in the password "letmein". The next step is to hash "letmein". This results in H_2 = 0d107d09f5bbe40cade3de5c71e9e9b7. Then, apply a reduction function r_2 to H_2 where $r_2(H_2) = H_2 + 3129034064 \bmod 26^7$. Converting $r_2(H_2)$ back to the password representation will result in the password "abcdefg".

Hence, this initial rainbow chain consists of the above three passwords.

Proposition 1 : Any given 3 passwords can be recoverd regardless of the size of keyspace and the hash applied.

Proof : Let size of keyspace = n. Let the hash fnction be denoted by h. Then, for simplification, let = to mean ≡ mod n for the subsequent parts of the proof. To prove the proposition, we show that there exists at least one arrangement to insert these 3 passwords such their corresponding reduction functions are distinct. Let the 3 passwords be p_1, p_2 and p_3. Let h(p_1) = a_1, h(p_2) = a_2 and h(p_3) = a_3. Suppose for all 6 arrangements, each arrangement results in having identical reduction functions. Thus, we obtain, the following set of equations:

(1) p_2 - a_1 = p_3 - a_2
(2) p_3 - a_1 = p_2 - a_3
(3) p_1 - a_2 = p_3 - a_1
(4) p_3 - a_2 = p_1 - a_3
(5) p_1 - a_3 = p_2 - a_1
(6) p_2 - a_3 = p_1 - a_2

Comparing equations 1 and 3, we get $p_1 + p_2 = 2p_3$
Comparing equations 2 and 5, we get $p_1 + p_3 = 2p_2$
Comparing equations 4 and 6, we get $p_3 + p_2 = 2p_1$

Solving these 3 new equations, we obtain $p_1 = p_2 = p_3$. This is a contradiction since p_1, p_2, p_3 are distinct modulo n; thus proving Proposition 1.

Proposition 2 : Any given 4 passwords can be recovered regardless of the size of keyspace and the hash applied.

Proof : Let size of keyspace = n. Let the hash function be denoted by h. Again, for simplification, let = to mean ≡ mod n for the subsequent parts of the proof. Let the 4 passwords be p_1, p_2, p_3 and p_4 and let h(p_1) = a_1, h(p_2) = a_2, h(p_3) = a_3 and h(p_4) = a_4.

Consider the 3 passwords p_1, p_2, p_3 which are placed in the first 3 entries of the chain. Applying Proposition 1, there exists an arrangement which will result in distinct reduction functions. Without loss of generality, assume that the first 3 passwords in the chain are p_1, p_2, p_3 in that order such that the corresponding reduction functions are distinct. Then, p_4 will be in the 4^{th} entry of the chain. Suppose p_4 - $a_3 \neq p_2$ - a_1 and p_4 - $a_3 \neq p_3$ - a_2. Then $p_1p_2p_3p_4$ is the desired order to place the passwords which ensures distinct reduction functions.

Suppose either p_4 - $a_3 = p_2$ - a_1 or p_4 - $a_3 = p_3$ - a_2.

Case 1 : p_4 - $a_3 = p_2$ - a_1
Consider the arrangement $p_4p_1p_2p_3$. If p_1 - $a_4 \neq p_2$ - a_1 and p_1 - $a_4 \neq p_3$ - a_2, we are done. Otherwise, either p_1 - $a_4 = p_2$ - a_1 or p_1 - $a_4 = p_3$ - a_2.

Case 1(a) : p_1 - $a_4 = p_2$ - $a_1 = p_4$ - a_3.
Consider the arrangement $p_1p_3p_4p_2$. Suppose not all the reduction functions are distinct, then p_3 - $a_1 = p_2$ - a_4. Next, consider the arrangement $p_4p_1p_3p_2$. If not all the reduction functions are distinct, then p_3 - $a_1 = p_2$ - a_3. This implies a_4 = a_3 and thus $p_1 = p_4$ which is a contradiction.

Case 1(b) : p_1 - $a_4 = p_3$ - a_2 and p_4 - $a_3 = p_2$ - a_1
If p_1 - $a_4 = p_1$ - a_3, then $a_3 = a_4$. Thus, $p_4p_3p_1p_2$ will be the desired arrangement. If $a_3 \neq a_4$, consider the arrangements $p_4p_2p_3p_1$, $p_1p_3p_4p_2$, $p_4p_1p_3p_2$, $p_2p_4p_1p_3$ and $p_1p_4p_2p_3$ in the order as stated. Suppose none of these arrangements result in distinct reduction functions.
By considering $p_4p_2p_3p_1$, we get p_2 - $a_4 = p_1$ - a_3.
By considering $p_1p_3p_4p_2$, we get p_1 - $a_3 = p_2$ - $a_4 = p_3$ - a_1.
By considering $p_4p_1p_3p_2$, we get p_1 - $a_4 = p_2$ - a_3.
By considering $p_2p_4p_1p_3$, we get p_4 - $a_2 = p_3$ - a_1.
By considering $p_1p_4p_2p_3$, we get p_4 - $a_1 = p_3$ - a_2.
From the above arrangement $p_2p_4p_1p_3$, we obtain p_3 - $a_1 = p_4$ - a_2 and from arrangement $p_1p_4p_2p_3$, we obtain p_4 - $a_1 = p_3$ - a_2. Hence, $p_3 = p_4$, which is a contradiction.

Case 2 : p_4 - $a_3 = p_3$ - a_2 and p_4 - $a_3 \neq p_2$ - a_1
Consider the arrangement $p_3p_4p_1p_2$. If p_4 - $a_3 \neq p_1$ - a_4 and p_2 - $a_1 \neq p_1$ - a_4, we are done. Otherwise, either p_4 - $a_3 = p_1$ - a_4 or p_1 - $a_4 = p_2$ - a_1.

Case 2(a) : p_4 - $a_3 = p_3$ - $a_2 = p_1$ - a_4
Consider the arrangement $p_3p_4p_2p_1$. If this arrangement results in distinct reduction functions, then p_2 - $a_4 = p_1$ - a_2. Next, consider arrangement $p_4p_2p_3p_1$. If this arrangement does not result in distinct reduction functions again, then we must have p_2 - $a_4 = p_1$ - a_3. Hence, $a_2 = a_3$ which implies $p_3 = p_4$, which is a contradiction.

Case 2(b) : p_1 - $a_4 = p_2$ - a_1 and p_4 - $a_3 = p_3$ - a_2
If $a_1 = a_3$, consider the arrangement $p_3p_2p_1p_4$. If this is not the desired arrangement, then $a_2 = a_4$. Hence, $p_2p_1p_3p_4$ will be the desired arrangement.
If $a_1 \neq a_3$, consider the arrangements $p_4p_1p_3p_2$, $p_2p_1p_3p_4$, $p_3p_2p_4p_1$ and $p_2p_3p_1p_4$ in this order. Suppose none of these arrangements result in distinct reduction functions, then we obtain the following set of equations: p_3 - $\mathrm{a}_1 = p_2$ - $\mathrm{a}_3 = p_1$ - a_2, p_4 - $\mathrm{a}_2 = p_1$ - $a_4 = p_2$ - a_1 and p_1 - $\mathrm{a}_3 = p_4$ - a_1. Simplifying, we obtain p_1 - $p_4 = p_3$ - p_2 and p_2 - $p_1 = p_3$ - p_4. Thus, $p_2 = p_3$, which is a contradiction.

This proves Proposition 2.

3.3 Methods of Constructing Chains

We propose 2 methods of constructing chains such that they include the desired passwords. We then provide an analysis of both methods in terms of feasilibity and the expected computational attempts required.

Method 1 : Compute all the possible values of p_i - a_j. Then, consider all possible chains that can be formed. For each chain, test if the chain results in distinct reduction functions. If so, we have found the required chain; otherwise continue testing the remaining ones until we find such a chain.

Method 2 : Compute all the possible values of p_i - a_j. Take the one which has a lowest occurrence frequency. That link will be the part of the generated chain. For the subsequent links, we choose the ones which are distinct from all previous ones in the chain and occur at a lower frequency. This step is repeated until we have the desired chain or we reach a point where we are unable to add any more links. In the latter case, we backtrack to the previous process and select another link instead, till the desired chain is obtained.

Example

Suppose we want to include 3 passwords p_1, p_2 and p_3 with the following password (plaintext) and hash values in the chain:
$\mathrm{p}_1 = 10$, $\mathrm{p}_2 = 20$, $\mathrm{p}_3 = 30$, $\mathrm{h}(\mathrm{p}_1) = \mathrm{a}_1 = 29$, $\mathrm{h}(\mathrm{p}_2) = \mathrm{a}_2 = 9$, $\mathrm{h}(\mathrm{p}_3) = a_3 = 19$

Then, p_1 - a_2 = 1, p_3 - a_2 = 21, p_2 - a_1 = -9, p_3 - a_1 = 1, p_1 - a_3 = -9, p_2 - a_3 = 1
Hence, only the chains $p_1p_2p_3$ and $p_2p_3p_1$ are the desired ones.

Applying Method 1,
Probability of getting a desired chain in 1 attempt = 1/3
Probability of getting the chain in 2 attempts = 4/6 x 2/5 = 4/15
Probability of getting the chain in 3 attempts = 4/6 x 3/5 x 2/4 = 1/5
Probability of getting the chain in 4 attempts = 4/6 x 3/5 x 2/4 x 2/3 = 2/15
Probability of getting the chain in 5 attempts = 4/6 x 3/5 x 2/4 x 1/3 = 1/15

Therefore, expected number of attempts to get a desired chain
= 1 x 1/3 + 2 x 4/15 + 3 x 1/5 + 4 x 2/15 + 5 x 1/15
= 7/3

Applying Method 2, since p_3 - a_2 occurs with the least frequency, we start the construction of the chain with p_2p_3. We are left with 2 possible links; either p_1 - a_3 or p_2 - a_1. Since both are distinct from the previous one, we select a link according to its occurrence frequency. In this case, both have the same frequency. Therefore, we can select either; e.g. p_1 - a_3. Thus, the generated chain is $p_2p_3p_1$.

Consider a general case of inserting n recoverable passwords. Suppose after computing all n(n-1) values of p_i - a_j, we have the following relations :

1) p_1 - a_2 = p_2 - a_3 = p_3 - a_4 = ... = p_{n-1} - a_n = p_n - a_1
2) the other n(n-2) expressions of p_i - a_j are all mutually distinct.

First, we need to compute the number of chains such that not all its reduction functions are distinct.
Number of chains such that some part of the chain contain $p_{i+2}p_{i+1}p_i$ or $p_2p_1p_n$ or $p_1p_np_{n-1}$
= n(n-2)!

Number of chains such that some part of the chain contain $p_{i+1}p_i$ and $p_{j+1}p_j$
= (n-2)![$\binom{n}{2}$ - n]
= (n-2)!$\frac{n(n-3)}{2}$

Total number of chains such that not all its reduction functions are distinct
= n(n-2)! + (n-2)!$\frac{n(n-3)}{2}$
= $\frac{n!}{2}$

Then, total number of chains such that all of its reduction functions are distinct
= n! - $\frac{n!}{2}$
= $\frac{n!}{2}$

Probability of getting a valid chain after k attempts
= Probability of getting the invalid chains for the first k-1 attempts and getting a valid chain on the kth attempt
= $\frac{n!/2}{n!}$ x $\frac{n!/2-1}{n!-1}$ x $\frac{n!/2-3}{n!-3}$ x .... x $\frac{n!/2-(k-2)}{n!-(k-2)}$ x $\frac{n!/2}{n!-(k-1)}$
= $[\frac{n!-(k-1)}{n!/2-(k-1)}$ / $\frac{n!}{n!/2}]$ x $\frac{n!/2}{n!-(k-1)}$

Expected number of attempts required
= $[\frac{n!}{2}$ / $\binom{n!}{n!/2}]\sum_{k=1}^{n!/2+1}$ $\binom{n!-(k-1)}{n!/2}\frac{k}{n!-(k-1)}$
= $\sum_{k=1}^{n!/2+1}$ $\binom{n!-k}{n!/2-1}$k / $\binom{n!}{n!/2}$

We consider the asymptotic value of n. Let z = $\frac{n!}{2}$
Then, the expected number of attempts required can be rewritten as
= $\sum_{k=1}^{z+1}$ $\binom{2z-k}{z-1}$k / $\binom{2z}{z}$
= $\frac{z!}{2(2z-1)!}\sum_{k=1}^{z+1}$k(2z-k)(2z-k-1).......(z-k+2)

As n increases, z increases. $\frac{z!}{2(2z-1)!}\sum_{k=1}^{z+1}$k(2z-k)(2z-k-1).......(z-k+2) approaches $\sum_{k=1}^{\infty}\frac{k}{2^k}$. However, this is expected of the geometric distribution with parameter $\frac{1}{2}$. Hence, $\sum_{k=1}^{\infty}\frac{k}{2^k}$ = 2 and $\frac{z!}{2(2z-1)!}\sum_{k=1}^{z+1}$k(2z-k)(2z-k-1).......(z-k+2) approaches 2 as z gets large. This in turn implies that $\sum_{k=1}^{n!/2+1}$ $\binom{n!-k}{n!/2-1}$k / $\binom{n!}{n!/2}$ approaches

2 as n gets large. Hence, for Method 1, we can deduce that for large n, the expected number of attempts required is close to 2.

For Method 2, we first select a link that occurs with the least frequency, e.g. p_2 - a_1. Then, we build the chain from this initial link and we get $p_1 p_2 p_3$ and so on until we arrive at $p_1 p_2 p_3$..........p_n, which is one of the desired chains.

4 Conclusion

In this paper, we presented the novel design of an Enhanced Dictionary Based Rainbow Table. We then proposed two new methods of chains construction. We analysed and proved the feasiblity of the proposed methods. We also analysed the probability of generating the desired chains in specific scenarios of different password space sizes and in the generic case of n password space, and expected computational attempts required using each method. The analysis results showed that the proposed Enhanced Dictionary Based Rainbow Table method is a promising new approach to efficiently recover passwords by taking into consideration both the use of common passwords (human memorable) and randomly generated passwords at the same time.

References

1. Cain and Abel: Password recovery tool (2011), http://www.oxid.it (retrieved December 2011)
2. Contini, S., Yin, Y.L.: Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer, Heidelberg (2006)
3. Denning, D.E.R.: Cryptography and Data Security. Addison-Wesley Publication (1982)
4. Dobbertin, H., Bosselaers, A., Preneel, B.: Ripemd-160: A Strengthened version of RIPEMD. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 71–82. Springer, Heidelberg (1996)
5. Fouque, P.-A., Leurent, G., Nguyen, P.Q.: Full Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 13–30. Springer, Heidelberg (2007)
6. Google News (2009), Favorite passwords: '1234' and 'password', http://www.google.com/hostednews/afp/article/ALeqM5jeUc6Bblnd0M19WVQWvjS6D2puvw (retrieved December, 2011)
7. Hellman, M.E.: A cryptanalytic time-memory trade-off. IEEE Transactions on Information Theory IT-26(4), 401–406 (1980)
8. John The Ripper, Password cracker (2011), http://www.openwall.com (retrieved December 2011)
9. LCPSoft, Lcpsoft programs (2011), http://www.lcpsoft.com (retrieved December 2011)
10. Narayanan, A., Shmatikov, V.: Fast dictionary attacks on passwords using time-space tradeoff. In: ACM Conference on Computer and Communications Security, pp. 364–372 (2005)

11. National Institute of Standards and Technology, NIST (2002), Secure hash standard. Federal Information Processing Standards Publication 180(2)
12. Oechslin, P.: Making a Faster Cryptanalytic Time-Memory Trade-Off. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 617–630. Springer, Heidelberg (2003)
13. Rivest, R.: The MD5 message-digest algorithm. IETF RFC 1321 (1992)
14. Sasaki, Y., Yamamoto, G., Aoki, K.: Practical password recovery on an MD5 challenge and response. Cryptology ePrint Archive, Report 2007/101 (2008)
15. Sasaki, Y., Wang, L., Ohta, K., Kunihiro, N.: Security of MD5 Challenge and Response: Extension of APOP Password Recovery Attack. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 1–18. Springer, Heidelberg (2008)
16. Smyth, S.M.: Searches of computers and computer data at the United States border: The need for a new framework following United States V. Arnold. Journal of Law, Technology and Policy (1), 69–105 (2009)
17. Thing, V.L.L., Ying, H.M.: A novel time-memory trade-off method for password recovery. Digital Investigation, International Journal of Digital Forensics and Incident Response 6(suppl.), S114–S120 (2009)
18. Todorov, D.: Mechanics of user identification and authentication: Fundamentals of identity management. Auerbach Publications, Taylor and Francis Group (2007)
19. Ying, H.M., Thing, V.L.L.: A novel rainbow table sorting method. In: International Conference on Technical and Legal Aspects of the e-Society (CYBERLAWS) (2011)
20. Weir, M.: Enough with the Insanity: Dictionary Based Rainbow Tabls. ShmooCon (2009)