Rainbow Tables Explained

By: Warpboy

Table of Contents

Total Pages: 11

Defining Rainbow Tables

So what exactly are rainbow tables? Rainbow tables are the new generation of cracking, using advanced developed methods for cracking passwords encrypted with algorithms such as the Message Digest 5 (MD5) or LanManager (LM). Rainbow tables have become more popular and more widely known for the speed at which passwords encrypted with these algorithms can be cracked. It is vital to stay up-to-date with technology; therefore, this paper will teach you all about rainbow tables and how they are being used and applied in todays world.

A rainbow table is a special type of lookup table offering a time-memory tradeoff used in recovering the plaintext password from a ciphertext generated by a one-way hash. Translation? A rainbow table is a lookup table, such as you lookup a word in a text file, but a little more complicated. Rainbow tables use time-memory trade-off (explained later) to decrease the amount of time to crack a ciphertext (encrypted word) into a plaintext. The algorithms md5 and lm are one way, meaning they cannot be decrypted, but they can be looked up.

Time-Memory Trade-Off

The traditional way to crack passwords is brute forcing, which would simply just try all the plaintexts one by one. This was and still is a time consuming method of cracking passwords. The implementation of Philippe Oechslin's time-memory trade-off method of decreasing time of cryptanalysis by using precalculated data stored in memory, is being used in rainbow tables. The idea of time-memory trade-off is to do all cracking time computation in advance and store the result in files (rainbow tables).

The process of pre- computation of rainbow-tables does take a decent amount of time. However, once all the tables are created, the time-memory trade-off cracker is hundreds of times faster than a traditional brute force cracker. Time-memory trade-off has revolutionized password cracking by decreasing the time of cryptanalysis. To summerize, time-memory trade-off is used in rainbow tables to speed up the time it takes to crack a ciphertext.

Setting Up Software

There is a variety of rainbow table generation software and time-memory trade-off crackers out there. For the sake of this tutorial and by personal preference I am going to use RainbowCrack which is available (FREE) for download at http://www.antsight.com/zsl/rainbowcrack/. If you dislike RainbowCrack and are looking for a rainbow table generator and a time-memory trade-off cracker check out Winrtgen and Cain&Abel, which are both available at http://www.oxid.it/projects.html (FREE).

Let's download and setup RainbowCrack. Extract its contents to your preferred directory; "C:\" for the tutorials sake, and rename the folder to something like "rainbowcrack". After extracting it should look something like in Figure 1.1, below.

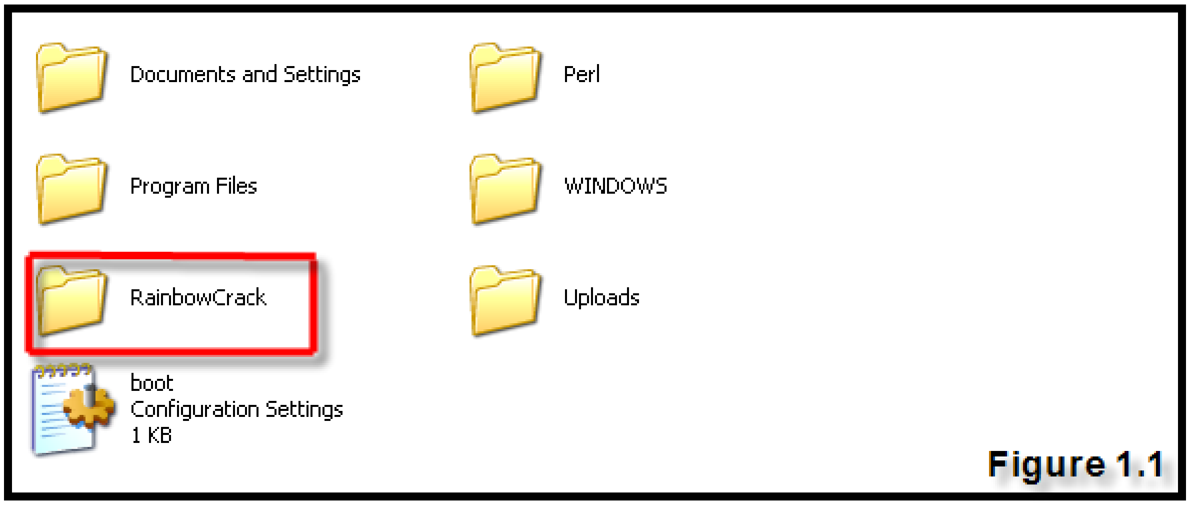


Figure 1.1

If it helps, you may want to take a look at the ReadMe, but this really isn't necessary. If there were no extraction errors and the files contents match the ones in the zip archive you downloaded from the Project RainbowCrack homepage, then you have successfully installed RainbowCrack to your computer.

Generating Rainbow Tables

0x04

Before you can begin to crack hashes you need to have a good amount of rainbow tables available. Therefore, during this chapter the generation process of rainbow tables is covered. The amount of space you are willing to set aside for your rainbow tables completely depends on your personal preference. If you have under 100gb I wouldn't recommend generating to many rainbow tables.

The program used to generate rainbow tables is "rtgen", located in the RainbowCrack directory we created earlier. To generate rainbow tables you need to specify specific arguements, these arguements consist of:

hash algorithm \ character set \ plaintext length minimum \ plaintext length maximum \ rainbow table index \ rainbow chain legth \ rainbow chain count \ file title suffix

Hash algorithm options: algorithm of choice (exp: md5)

Character set: located in the RainbowCrack directory as charset.txt, can be modified (exp: alpha-numeric [123456789ABCDEFGHIJKLMNOPQRSTUV])

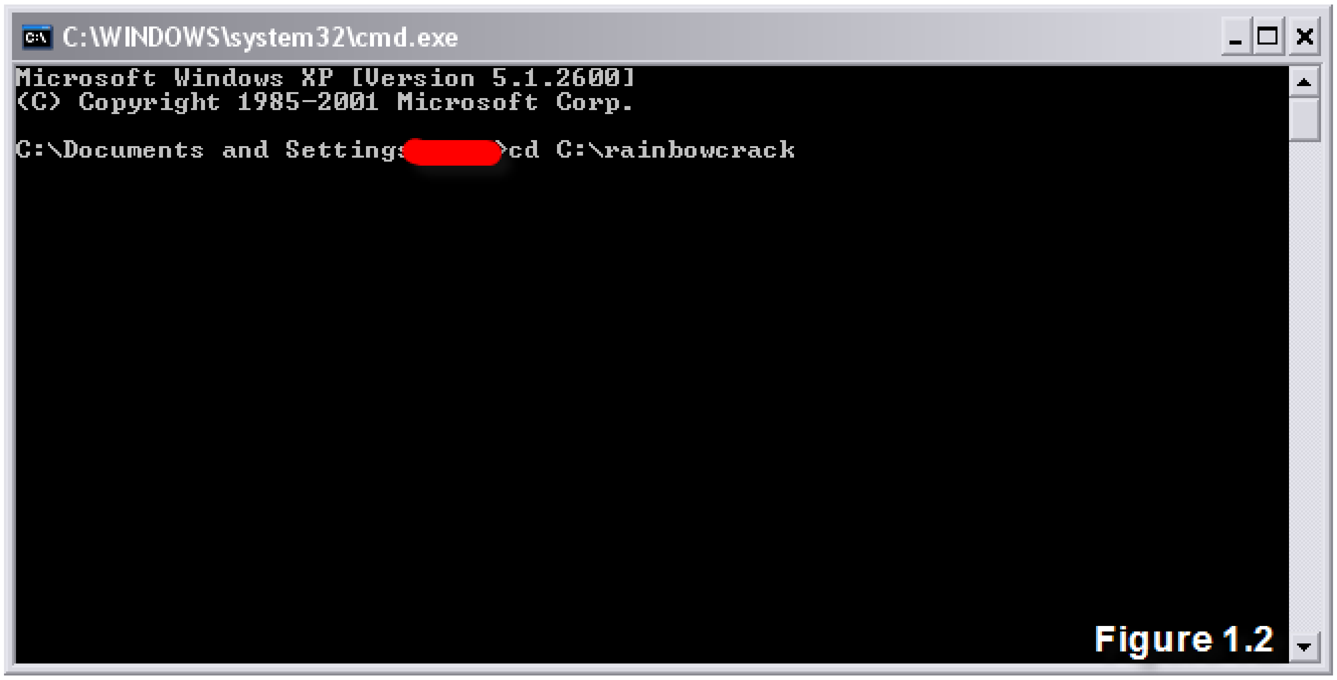
Plaintext length minimum: Minimum characters in each word (exp: 1)

Plaintext length maximum: Maximum amount of characters in each word(exp:8)

Rainbow Table Index: The index of the rainbow table (exp: 0)

Rainbow Chain Length: The length of the rainbow chain (exp: 11300)
Rainbow Chain Count: The amount of rainbow chains to generate (exp: 6000)
File Title Suffix: This is used for rainbow tables which are to be linked with eachother to prevent duplicating (exp: 0)

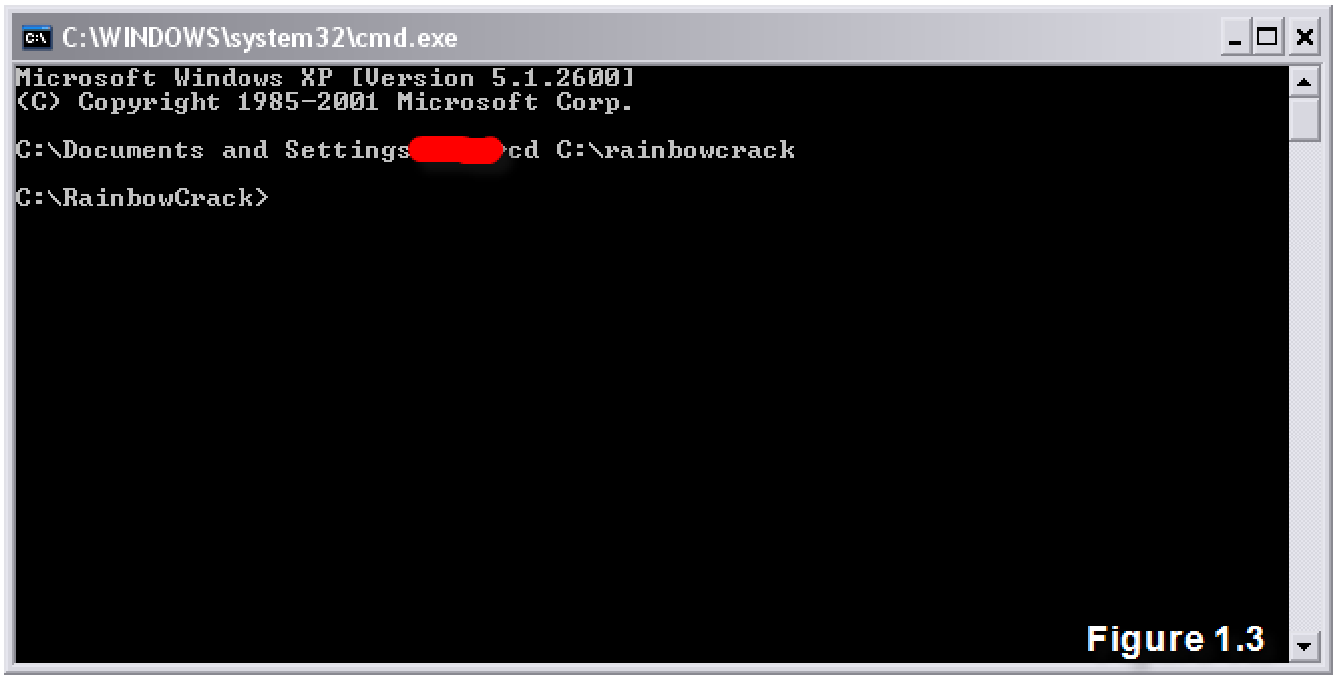
Before generating a table it is a good idea to check and see an approximantation of the amount of time the table is going to take to generate. To do this you use the "-bench" arguement at the end of the arguements (excluding the rainbow chain count/length and file title suffix arguements). To do this first open the cmd prompt (because rtgen is a console application) then cd to the rainbow crack directory (where you installed rainbow crack). See Figure 1.2 below.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings[redacted]>cd C:\rainbowcrack
```

Figure 1.2

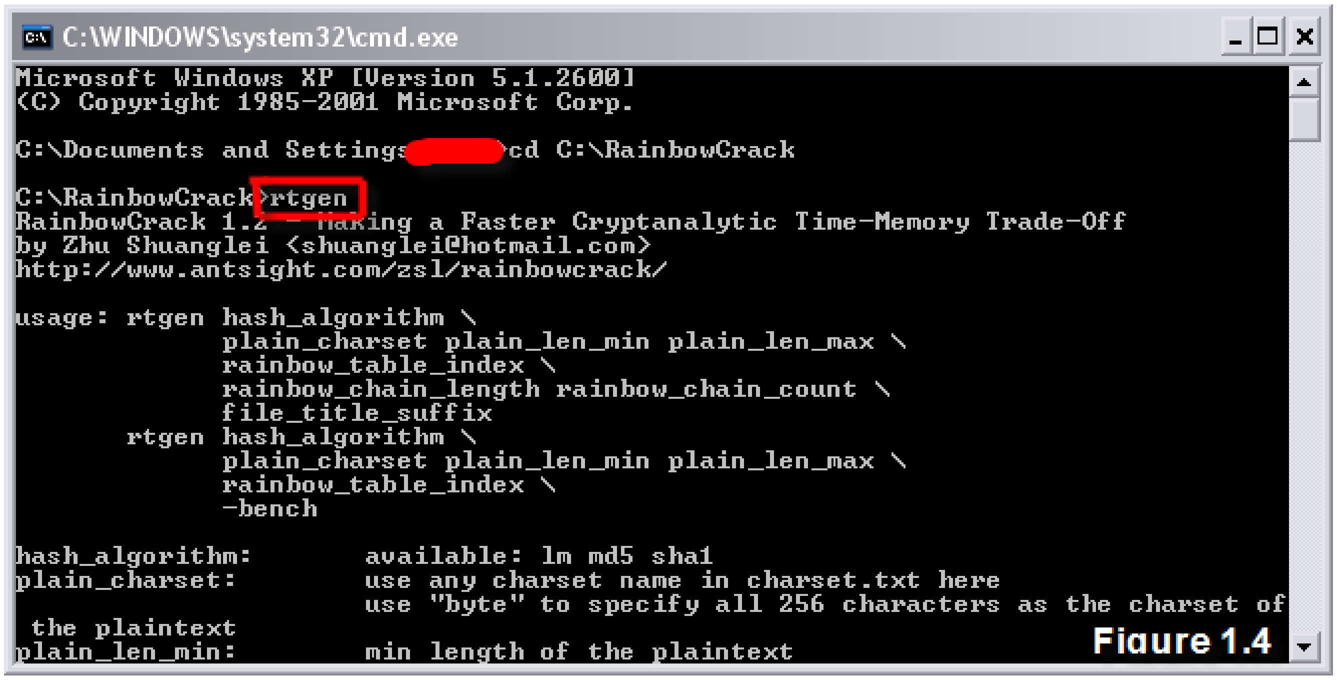
Now you should be in the rainbow crack directory, like below in Figure 1.3.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings[redacted]cd C:\rainbowcrack

C:\RainbowCrack>
```

Figure 1.3

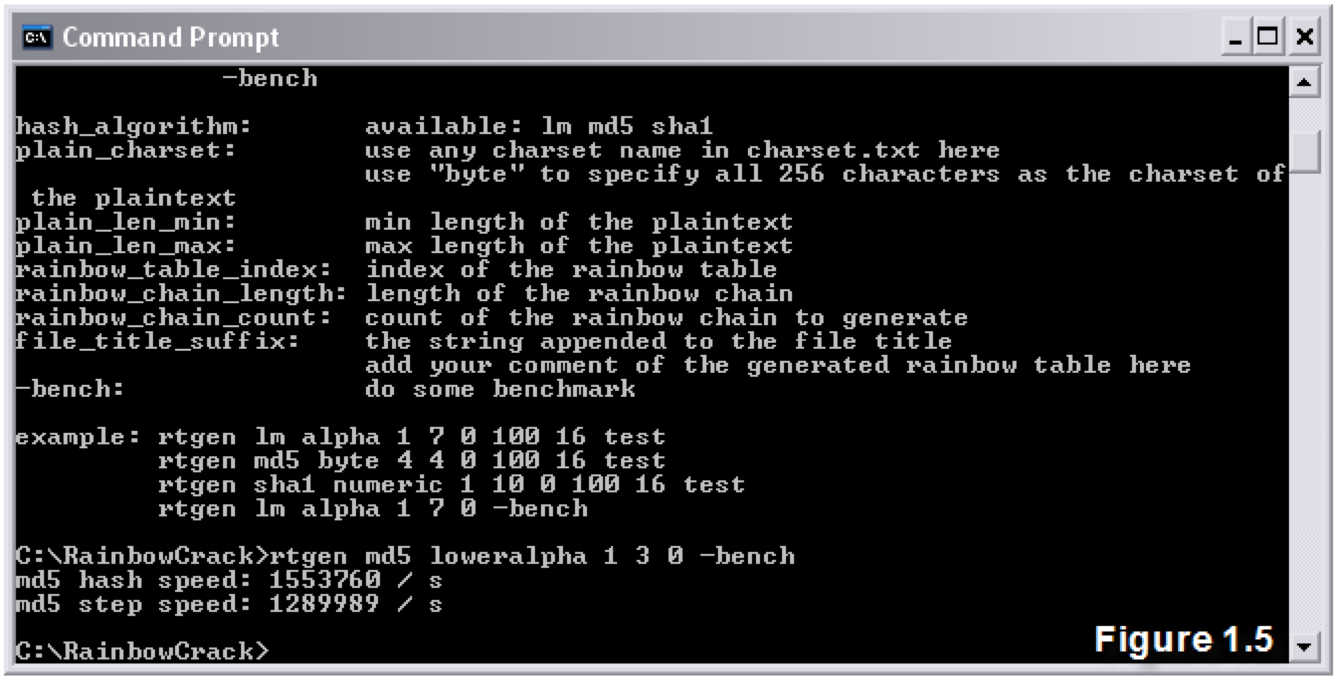
Now that you are in the RainbowCrack directory you need to start 'rtgen.exe' or the rainbow table generator and then you can start to generate the rainbow tables. Let's start rtgen and see what happens, see Figure 1.4 below.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings[redacted]>cd C:\RainbowCrack

C:\RainbowCrack>rtgen
RainbowCrack 1.2 - Making a Faster Cryptanalytic Time-Memory Trade-Off
by Zhu Shuanglei <shuanglei@hotmail.com>
http://www.antsight.com/zsl/rainbowcrack/

usage: rtgen hash_algorithm \
             plain_charset plain_len_min plain_len_max \
             rainbow_table_index \
             rainbow_chain_length rainbow_chain_count \
             file_title_suffix
       rtgen hash_algorithm \
             plain_charset plain_len_min plain_len_max \
             rainbow_table_index \
             -bench

hash_algorithm:        available: lm md5 sha1
plain_charset:         use any charset name in charset.txt here
                       use "byte" to specify all 256 characters as the charset of
 the plaintext
plain_len_min:         min length of the plaintext
```

Figure 1.4

As you can see it's just the arguments (covered earlier) that rtgen requires to generate a table based on the user inputted arguments. Now that we know how the program works, let's generate our first rainbow table. Before you can generate, it's a good idea to benchmark or figure out how long the table is going to take to be created. To do this you use the -bench argument, see Figure 1.5 below.



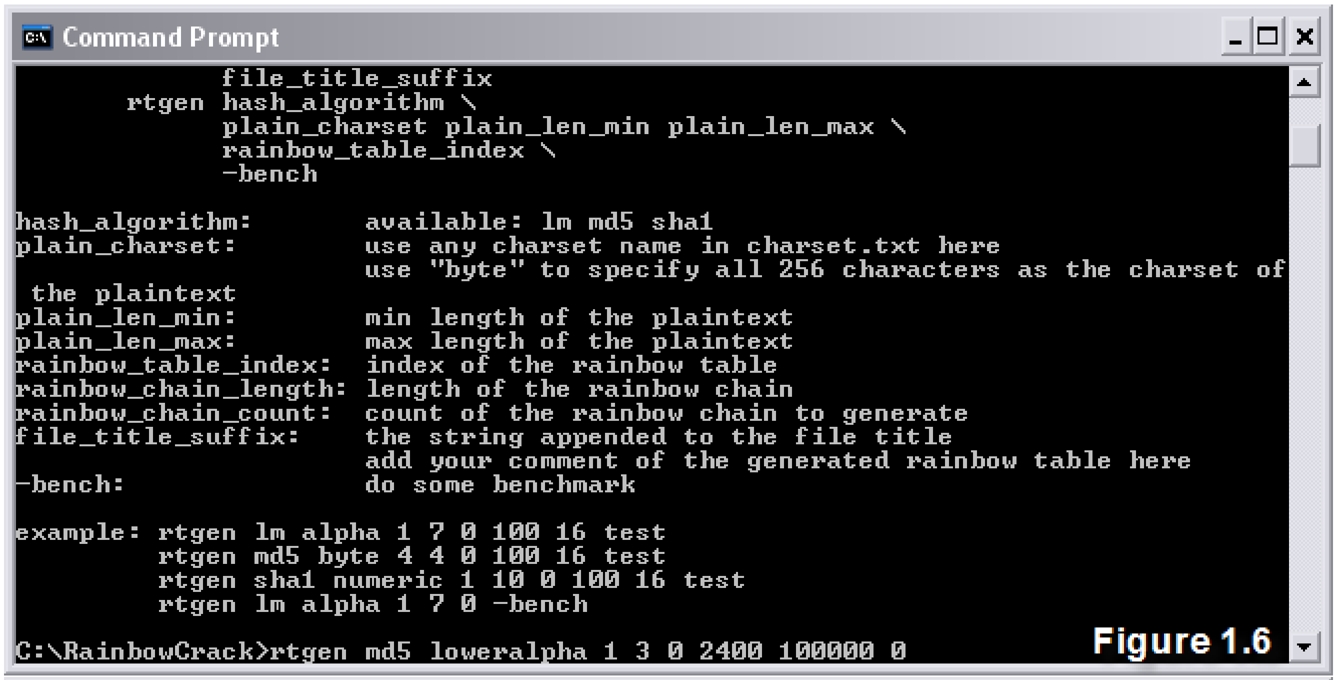
```
Command Prompt
             -bench

hash_algorithm:        available: lm md5 sha1
plain_charset:         use any charset name in charset.txt here
                       use "byte" to specify all 256 characters as the charset of
 the plaintext
plain_len_min:         min length of the plaintext
plain_len_max:         max length of the plaintext
rainbow_table_index:   index of the rainbow table
rainbow_chain_length:  length of the rainbow chain
rainbow_chain_count:   count of the rainbow chain to generate
file_title_suffix:     the string appended to the file title
                       add your comment of the generated rainbow table here
-bench:                do some benchmark

example: rtgen lm alpha 1 7 0 100 16 test
         rtgen md5 byte 4 4 0 100 16 test
         rtgen sha1 numeric 1 10 0 100 16 test
         rtgen lm alpha 1 7 0 -bench

C:\RainbowCrack>rtgen md5 loweralpha 1 3 0 -bench
md5 hash speed: 1553760 / s
md5 step speed: 1289989 / s

C:\RainbowCrack>
```

Figure 1.5

After you have tested to see how long the generation process is going to take, you're ready to start generating. The disheartening part about the generation process is that it usually takes an incredibly long time. If you have multiple computers, then generating sets of rainbow tables would obviously not take as long.

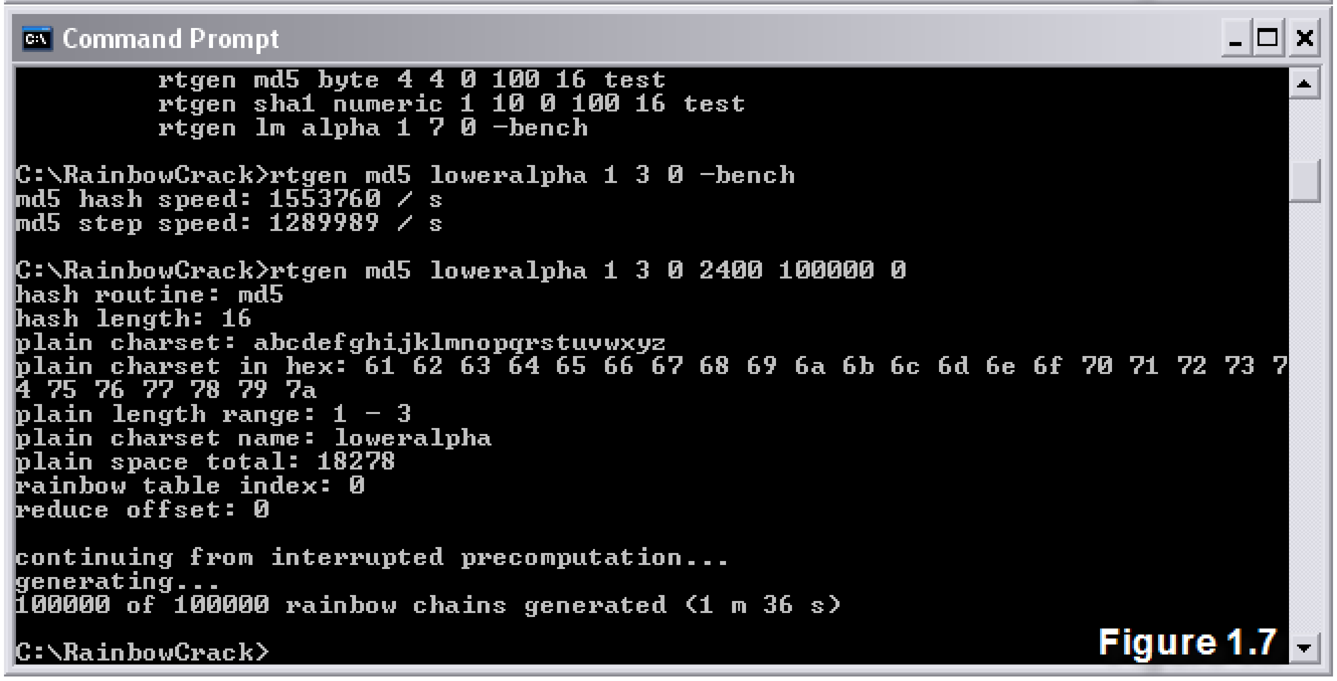
Sets of rainbow tables are just rainbow tables based on each other, for example you can't have a 39 GB rainbow table because the computer can't handle this large size. However, you can have 39 by 1 GB tables; you can do this by having indexes on your tables. Rtgen knows that if the index is 1 and there is a 0 indexed table then it won't re-generate the same contents that were in the first table(0 table). Basically, rainbow table sets allow you to have multiple tables that really are one big table, just split up into separate ones.

In Figure 1.6 below, there is a basic 1-3 character lower-alpha char. set rainbow table generating. After its generated it should look something like Figure 1.7.



```
             file_title_suffix
       rtgen hash_algorithm \
             plain_charset plain_len_min plain_len_max \
             rainbow_table_index \
             -bench

hash_algorithm:        available: lm md5 sha1
plain_charset:         use any charset name in charset.txt here
                       use "byte" to specify all 256 characters as the charset of
 the plaintext
plain_len_min:         min length of the plaintext
plain_len_max:         max length of the plaintext
rainbow_table_index:   index of the rainbow table
rainbow_chain_length:  length of the rainbow chain
rainbow_chain_count:   count of the rainbow chain to generate
file_title_suffix:     the string appended to the file title
                       add your comment of the generated rainbow table here
-bench:                do some benchmark

example: rtgen lm alpha 1 7 0 100 16 test
         rtgen md5 byte 4 4 0 100 16 test
         rtgen sha1 numeric 1 10 0 100 16 test
         rtgen lm alpha 1 7 0 -bench

C:\RainbowCrack>rtgen md5 loweralpha 1 3 0 2400 100000 0
```

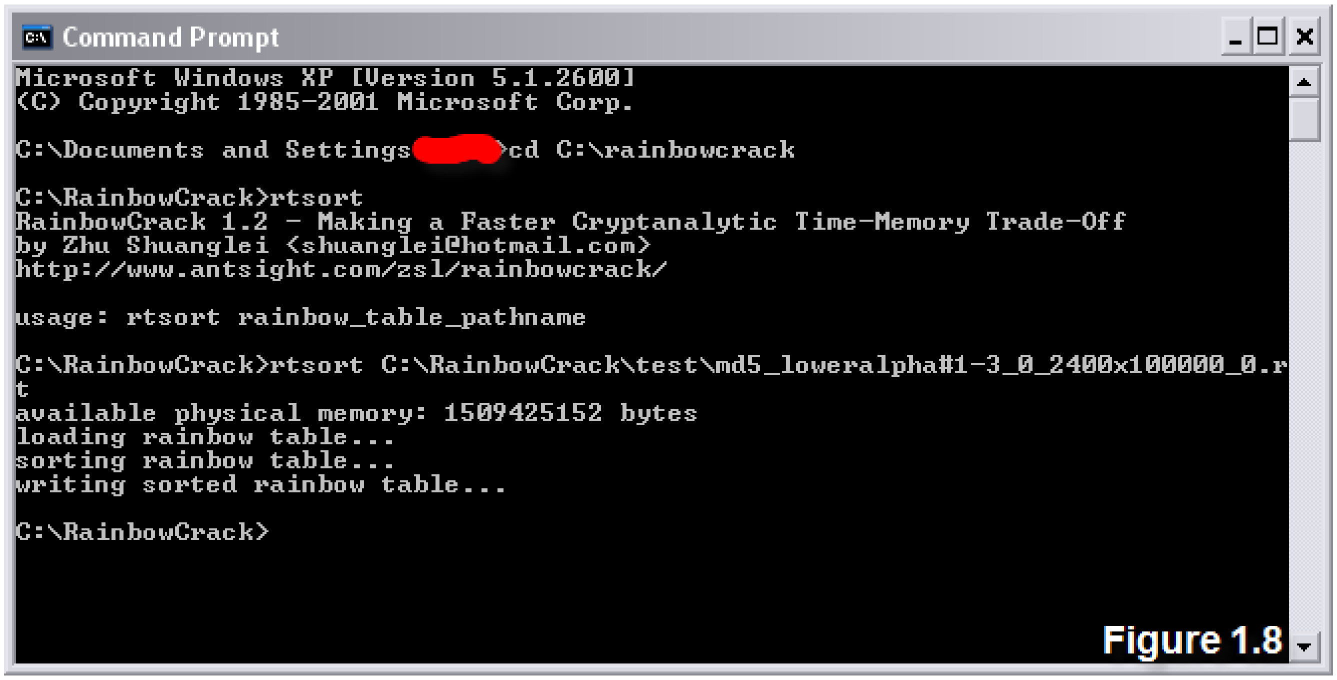
Figure 1.6



```
         rtgen md5 byte 4 4 0 100 16 test
         rtgen sha1 numeric 1 10 0 100 16 test
         rtgen lm alpha 1 7 0 -bench

C:\RainbowCrack>rtgen md5 loweralpha 1 3 0 -bench
md5 hash speed: 1553760 / s
md5 step speed: 1289989 / s

C:\RainbowCrack>rtgen md5 loweralpha 1 3 0 2400 100000 0
hash routine: md5
hash length: 16
plain charset: abcdefghijklmnopqrstuvwxyz
plain charset in hex: 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 7
4 75 76 77 78 79 7a
plain length range: 1 - 3
plain charset name: loweralpha
plain space total: 18278
rainbow table index: 0
reduce offset: 0

continuing from interrupted precomputation...
generating...
100000 of 100000 rainbow chains generated (1 m 36 s)

C:\RainbowCrack>
```

Figure 1.7

After the generation process is complete you have to sort the rainbow table using rtsort, all this program does is organize the newly created rainbow table. See Figure 1.8 below, where the previous rainbow table that was generated in Figure 1.7 is being sorted.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings[redacted]>cd C:\rainbowcrack

C:\RainbowCrack>rtsort
RainbowCrack 1.2 - Making a Faster Cryptanalytic Time-Memory Trade-Off
by Zhu Shuanglei <shuanglei@hotmail.com>
http://www.antsight.com/zsl/rainbowcrack/

usage: rtsort rainbow_table_pathname

C:\RainbowCrack>rtsort C:\RainbowCrack\test\md5_loweralpha#1-3_0_2400x100000_0.r
t
available physical memory: 1509425152 bytes
loading rainbow table...
sorting rainbow table...
writing sorted rainbow table...

C:\RainbowCrack>
```

Figure 1.8

After generating and sorting is complete it's time to test the rainbow table out.

0x05

Cracking Hashes

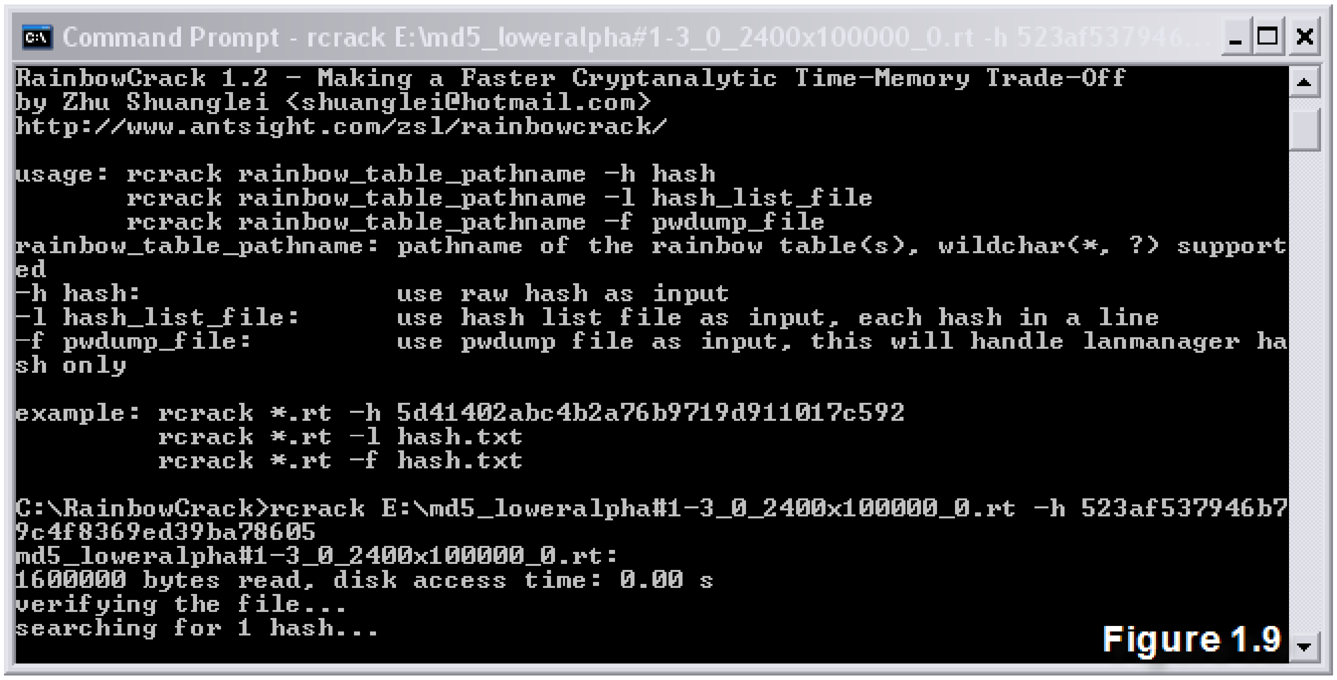
Since the rainbow table's algorithm that we just generated is MD5 lets go encrypt a simple plain-text with the alogrithm. There are tons of online MD5 encryptors on the net, to make it easy you can go to www.securitydb.org/cracker/ or you can google "MD5 encryptor".

Remember when submitting the plain text, make sure it matches the character set used in the previously generated rainbow table. For the sake of this paper, im going to encrypt 'ad' (without quotes) with the md5 alogrithm. Then I am going to attempt to crack the hash with the rainbow table. This is really only to test the rainbow tables ability.

ad:523af537946b79c4f8369ed39ba78605

Now the rainbow table should be able to crack the hash and give the plaintext 'ad'. The program used is rcrack, the arguements for it are easy, you can actually load all the rainbow tables in a directory with it, but in this tutorial we are only using one rainbow table.

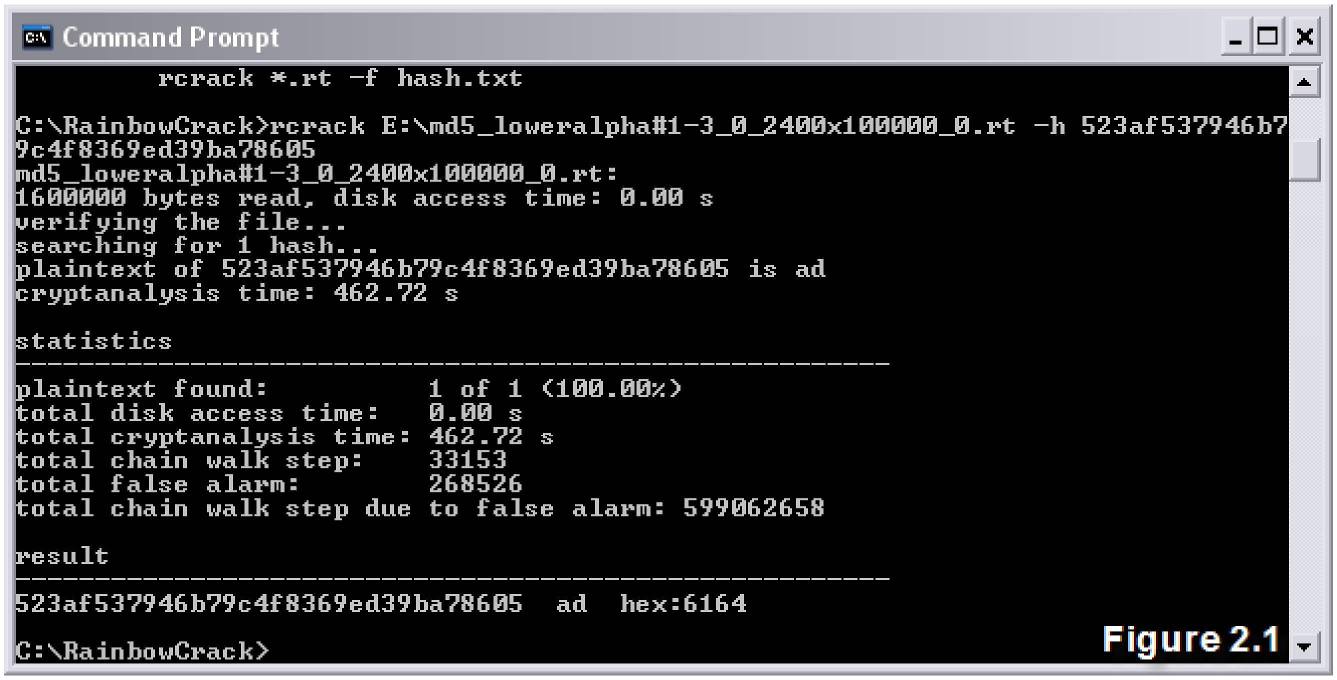
In Figure 1.9 (below) 'rcrack' is searching for the hash in the rainbow table.


Command Prompt - rcrack E:\md5_loweralpha#1-3_0_2400x100000_0.rt -h 523af537946 ...

```
RainbowCrack 1.2 - Making a Faster Cryptanalytic Time-Memory Trade-Off
by Zhu Shuanglei <shuanglei@hotmail.com>
http://www.antsight.com/zsl/rainbowcrack/

usage: rcrack rainbow_table_pathname -h hash
       rcrack rainbow_table_pathname -l hash_list_file
       rcrack rainbow_table_pathname -f pwdump_file
rainbow_table_pathname: pathname of the rainbow table(s), wildchar(*, ?) support
ed
-h hash:                use raw hash as input
-l hash_list_file:      use hash list file as input, each hash in a line
-f pwdump_file:         use pwdump file as input, this will handle lanmanager ha
sh only

example: rcrack *.rt -h 5d41402abc4b2a76b9719d911017c592
         rcrack *.rt -l hash.txt
         rcrack *.rt -f hash.txt

C:\RainbowCrack>rcrack E:\md5_loweralpha#1-3_0_2400x100000_0.rt -h 523af537946b7
9c4f8369ed39ba78605
md5_loweralpha#1-3_0_2400x100000_0.rt:
1600000 bytes read, disk access time: 0.00 s
verifying the file...
searching for 1 hash...
```

Figure 1.9

In Figure 2.1 (below) you can see the rainbow table worked.



```
         rcrack *.rt -f hash.txt

C:\RainbowCrack>rcrack E:\md5_loweralpha#1-3_0_2400x100000_0.rt -h 523af537946b7
9c4f8369ed39ba78605
md5_loweralpha#1-3_0_2400x100000_0.rt:
1600000 bytes read, disk access time: 0.00 s
verifying the file...
searching for 1 hash...
plaintext of 523af537946b79c4f8369ed39ba78605 is ad
cryptanalysis time: 462.72 s

statistics
-------------------------------------------------------
plaintext found:          1 of 1 (100.00%)
total disk access time:   0.00 s
total cryptanalysis time: 462.72 s
total chain walk step:    33153
total false alarm:        268526
total chain walk step due to false alarm: 599062658

result
-------------------------------------------------------
523af537946b79c4f8369ed39ba78605  ad  hex:6164

C:\RainbowCrack>
```

Figure 2.1

Congrats! You have successfully generated and sorted a rainbow table and used it to crack a simple hash. With 'rcrack' there are 2 other arguments besides (-h). You have -p which handles a password dump file, only with the lan manager algorithm though. Then there is -l, which handles a text file with hashes. So you can crack multiple hashes in 1 session.

0x06

The End

During this paper, the word 'crack' or 'cracking' is used; however, the hashes are never really cracked. The MD5 and LM algorithms have not yet been cracked, they are said to be one-way algorithms so they can only be looked up. So really when you say "Im cracking this hash with my rainbow tables", you are incorrect. However, with modern slang that statement could be treated as if it were valid.

Some resources for furthering your knowledge on rainbow tables:

www.antsight.com/zsl/rainbowcrack/
http://en.wikipedia.org/wiki/Rainbow_table
http://lasecwww.epfl.ch/~oechslin/publications/crypto03.pdf

Some online time-memory trade-off crackers:

www.milw0rm.com/cracker/
www.plain-text.info

0x07

Shoutz / Contact Info

ShoutZ: **TimQ, Z666, Ice_Dragon, kAoTiX, Ethernet, PunkerX, The-Maggot, theNerd, Archangel, Phrankeh, Grim, Splinter, Gammarayz, Maverick, Wolv, NinjaOptix, and SDB memberz + all my otha friends not mentioned**

Contact Info

E-mail: Warpboy1@yahoo.com
MSNM: Warpboy1@yahoo.com
Website: www.securitydb.org