Rainbow Table to Crack Password using MD5 Hashing Algorithm

Himanshu Kumar, Sudhanshu Kumar, Remya Joseph, Dhananjay Kumar, Sunil Kumar Shrinarayan Singh, Ajay Kumar, Praveen Kumar.

Himanshu Kumar

IEEE Conference Publishing

School of Information Technology and Engineering (SITE)
VIT University, Vellore, India

Abstract— Rainbow tables are basically huge tables filled with hash values and are used to find required password. Rainbow Table is used by the hackers to find the password by reversing the hashing function. Hashing the plaintext or password is a 1-way function which implies that hash can't be decrypted to find the required password[10]. To authenticate the user a system takes the hash value generated by the hash function on user's computer and it is compared with the hash value stored in the table on the server machine. If the hash matches, then the user is authenticated and can access the system. Rainbow tables are used to crack the password in short amount of time as compared to brute force technique, but it takes a lot of storage to hold rainbow table itself[1]. It is the most efficient methods for cracking passwords. This paper presents the implementation of Rainbow tables for cracking passwords of operating systems such as Windows7 and application which uses Message Digest v5(MD5) and Simple Hash Algorithmv1(SHA1) as their password hashing mechanism. It discusses the functionality of Rainbow Tables and its advantages over conventional brute-force approach and the usage of rainbow table to crack windows password.

Keyword- Time-memory Trade-off, MD5, character set, hash speed, cryptanalysis time, success probability.

I. INTRODUCTION

Cryptographic hashes are one of the major functions in information security. They have the property of being one way and because of this property they have major usage in: (a) providing message integrity. (b) Storing passwords in operating system. Some of the well-known hash algorithms are Message Digest-5(MD5) [7], Secure Hash Algorithm-1(SHA1) [8], and NT LAN Manager (NTLM) [9] etc. Since hashes are one way attack, it follows two approaches. (a) Brute force approach: computes hashes for probable message space one by one and matches with the given hash. This approach however has the following disadvantage: it consumes lot of time and resources as message size increases and the computation is not reusable. (b) Cryptanalytic approach: It uses time memory tradeoff approach. This method recomputes message hash pairs and stores them in a file to be used later for search. But storing it requires a lot of space.

II. MESSAGE DIGEST ALGORITHM (MD5)

Step1: Append padding bits

The input message is "padded" (extended) so that its length (in bits) equals to 448 mod 512. Padding is always performed, even if the length of the message is already 448 mod 512. Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448 mod 512. At least one bit and at most 512 bits are appended.

Step2. Append length

A 64-bit representation of the length of the message is appended to the result of step1. If the length of the message is greater than 2^64, only the low order 64 bits will be used. The resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits. The input message will have a length that is an exact multiple of 16 (32-bit) words.

Step3. Initialize MD buffer

A four-word buffer (A, B, C, D) is used to compute the message digest. Each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal, low-order bytes first):

word A: 01 23 45 67

word B: 89 ab cd ef

word C: fe dc ba 98

word D: 76 54 32 10

978-1-4673-5758-6/13/$31.00 © 2013 IEEE

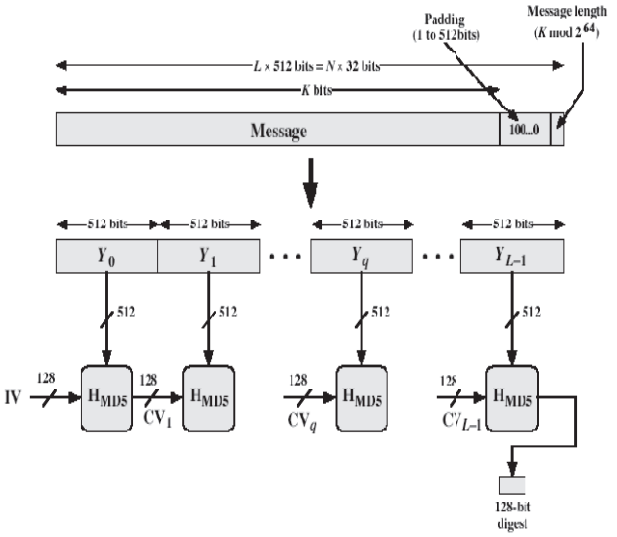

Fig.1. MD5 Block Diagram

MD5 Process:-

It has as many stages as the number of 512-bit blocks in the final padded message. Digest: 4 32-bit words: MD=A|B|C|D. Every message block contains 16 32-bit words: m0|m1|m2…|m15. Every stage consists of 4 passes over the message block and each modifys message digest. Each block contains 4 rounds and each round has 16 steps.

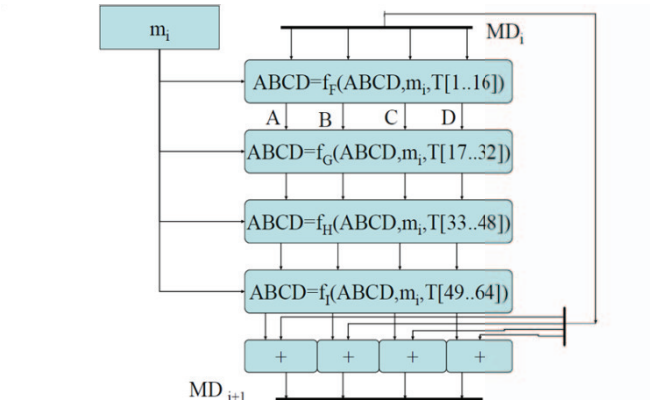

Fig.2. MD5 Process Diagram

III. RELATED WORK

- Many searching algorithms such as knapsack and discrete logarithm algorithms uses time- memory trade-off approach [6]. If there are N possible solutions, T operations and M word of memory then searching in Memory trade off approach takes product of time and memory which is TM. This paper uses the concept of time-memory trade off. It uses block cipher which is one of the major constraints in MD5.
- In [5] Philippe suggest using grand points as the end-points of the chains. When cipher text is given, generate a chain of keys until grand point is found and then look it up in the memory. This reduces the number of memory lookups.
- In [2] Orhun Kara suggest that how to find pre-images on a hash function which uses MD5 as the hashing algorithm. It uses Rainbow Tables for finding pre-images of hash functions. But it can be argued that Rainbow Tables are not effective for large digest size. But all hash functions don't require large digest size. Rainbow tables can be real threat for hash function of smaller digest size to find pre-images with work load within specified time.
- Rainbow tables are one of the efficient methods for cracking passwords [1], it uses hashes on different cryptanalytic algorithms. It's a kind of look-up which offers an almost optimal memory –time trade-off for recovering the plaintext password from a password hash which is generated by hash function like MD5.
- The equations are an exception to the prescribed specifications of this template. You will need to determine whether or not your equation should be typed using either the Times New Roman or the Symbol font (please no other font). To create multi-leveled equations, it may be necessary to treat the equation as a graphic and insert it into the text after your paper is styled.

IV. Proposed System

To authenticate the user a system takes the hash value generated by the hash function on user's computer and it is compared with the hash value stored in the table on the server machine. If the hash matches, then the user is authenticated and can access the system [11].

Example:

If the set of plain text is [0123456789] {6} i.e., to generate rainbow table of all numeric passwords of length 6 and the hashing function is MD5.

A hash of plain text might be MD5 ("493823") □ "222f00dc4b79f131c89cff641d188c50"

If hash function is H and it has a finite set of password i.e., P. Thus a data structure is generated for output hash function h and it locates an element p in P such that H (p) =h. This is the simplest way to compute H (p) for all p in P.

The rainbow table is generated in the following way:

-starting point/ end point

-Rainbow chain length and count.

-hash algorithm.

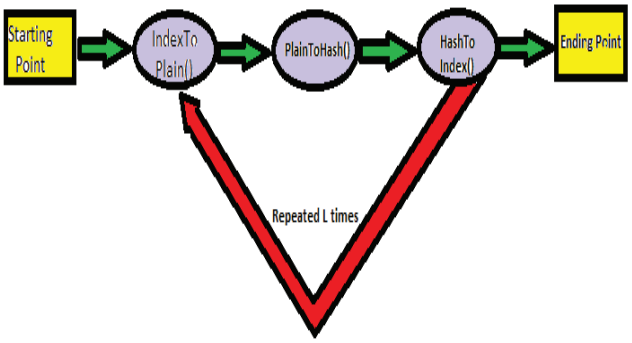

Fig.3. Block Diagram of Rainbow Table

Hash chains [2] are techniques for decreasing the space requirement in disk. It defines a reduced function R to map hash values into values of P again. It is done by an alternate operation of hash function and the reduction function chains of alternate passwords and corresponding hash values are formed.

An example, P is a set of lowercase alphabetic 6- character passwords, and the hash values were 32-bit long[4], then the chain is as below:

aaaaaa →281daf40 → sgfnyd → 920ecf10 →keibgt.

Take a set of passwords from P to generate a table, and then compute a chain of say length K and store the first and the last password of each chain which is processed.

In the chain the 1st password is called starting point and the last one is called ending point.

In next step it finds a password for the given hash value h, then it computes a chain starting with hand then applying R and H alternatively.

An example, if the hash is 920ecf10, its chain is created by applying R:

920ecf10→ kiebgt

Now as "kiegbt" is the endpoint of chain in table, select its password "aaaaaa" and finds out its chain until 920ecf10 is found.

aaaaaa→281DAF40→sgfnyd→920ECF10. Thus according to the table the password is "sgfnyd".

V. Result & Discussion

The following table demonstrates the usefulness of rainbow table in cracking fixed length passwords for different hash algorithms [6]. The parameter used in the table is explained below:

1) Char set: Set of characters (ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789!@#$%^&*()-_+=~`[]{}|\:;"'<>,.?/) and password length is length of password, these two determine total number of keys i.e. key space.

2) Disk Space: It is the size of Rainbow Table.

3) Chain Length and Chain Count are length and number of Rainbow Chains.

4) Hash Speed is hash calculating speed and Precomputation Time is total time to generate Rainbow Table, these two depends on the speed of the hardware.

5) Cryptanalysis Time is time taken to get the password given its hash and Rainbow Table.

6) Success Probability measures amount of success. Since Rainbow Table is probabilistic in nature success is not guaranteed always.

TABLE 1: RESULT TABLE FOR DIFFERENT HASH ALGORITHM.

HASH ALGORITHM PARAMETER	MD5	SHA1	NTLM
Char Set	Alpha numeric	Alphabetic	Allspace
Password Length	1-7	1-8	6-8
Key Space	80603140212 keys	217180147158 keys	463985095954432 keys
Disk Space	610.35 MB	762.93 MB	7.45 GB
Chain Length	240000	30000	250000
Chain Count	400000	50000000	600000000
Hash Speed	4440497 hash/sec	1882530 hash/sec	2857142 hash/sec
Success Probability	97.97%	94.96%	80.95%
Pre-computed Time	4.9 days	13.6 days	23.7 year
Crypt analysis Time	3.5 hours	5.8 minutes	2 days

Snapshots:-

Fig.4. Enter the text to find hash.

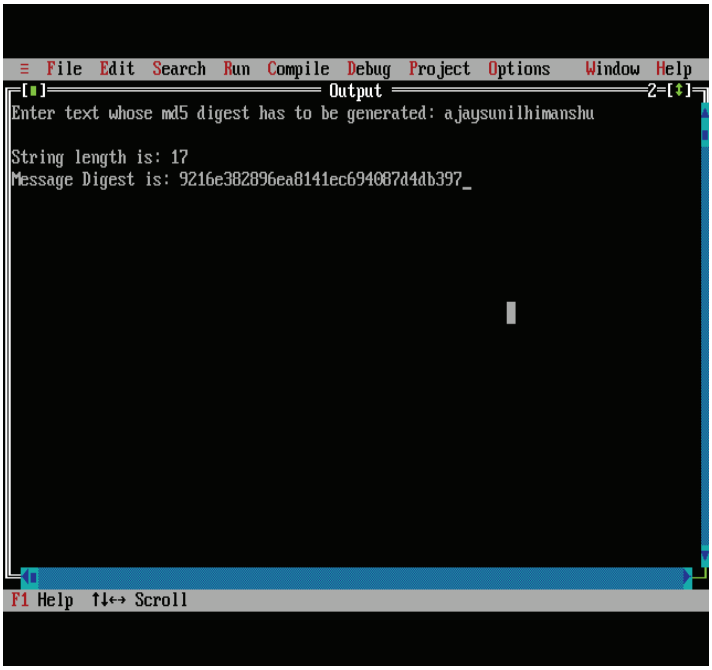

Fig.5. Hash is generated.

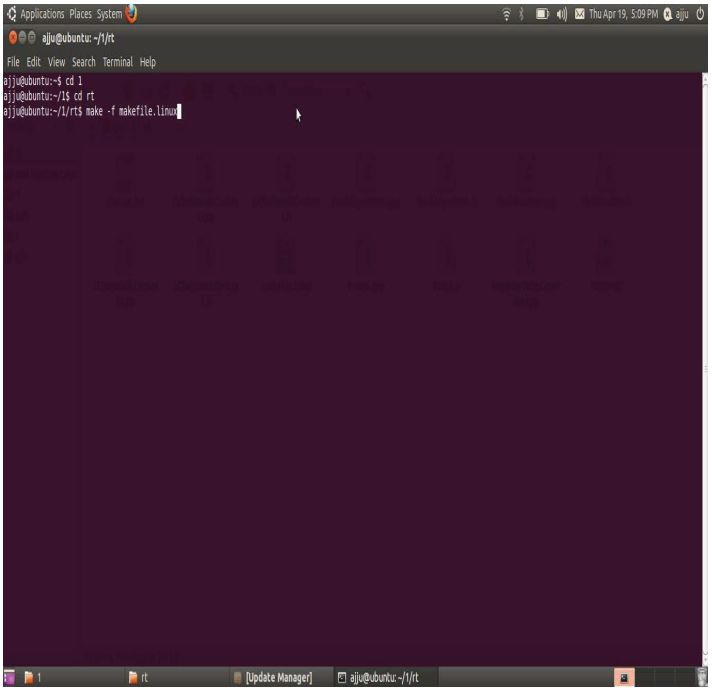

Fig.6. Run makefile.linux file.

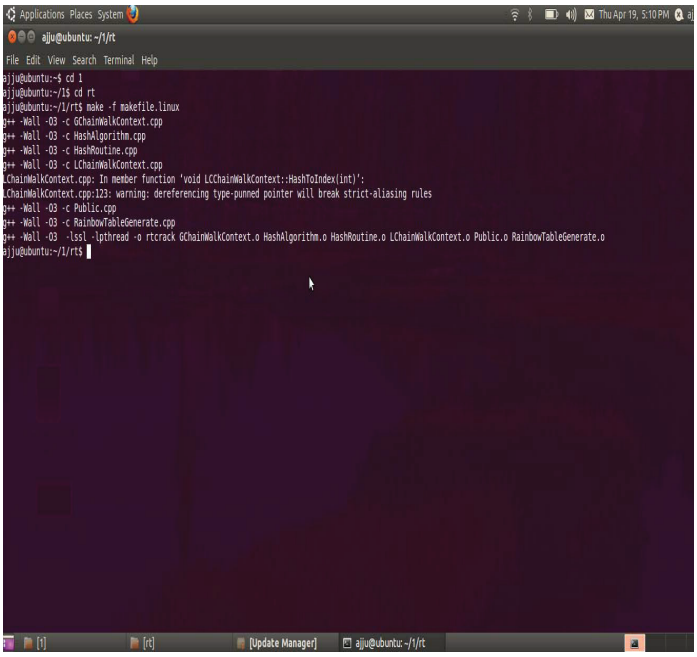

Fig.7. Creating object file for all programs.

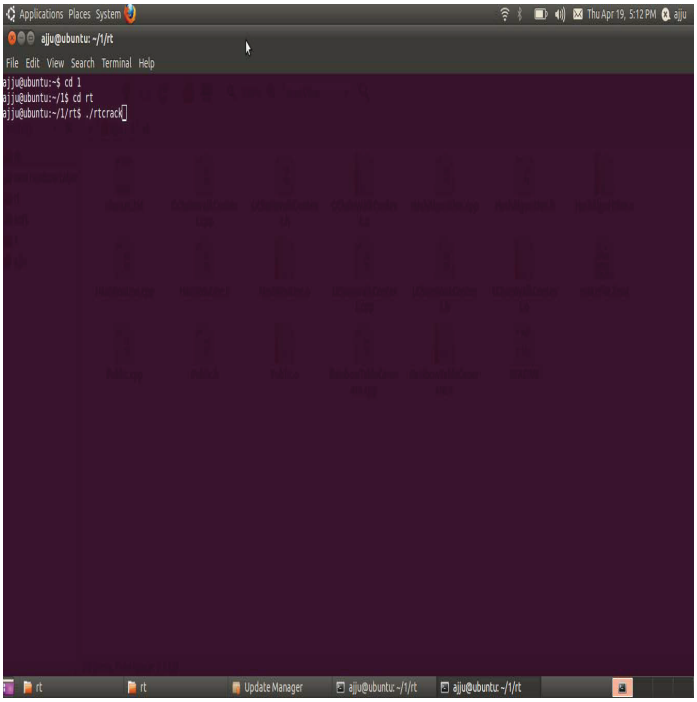

Fig.8. Run all programs.

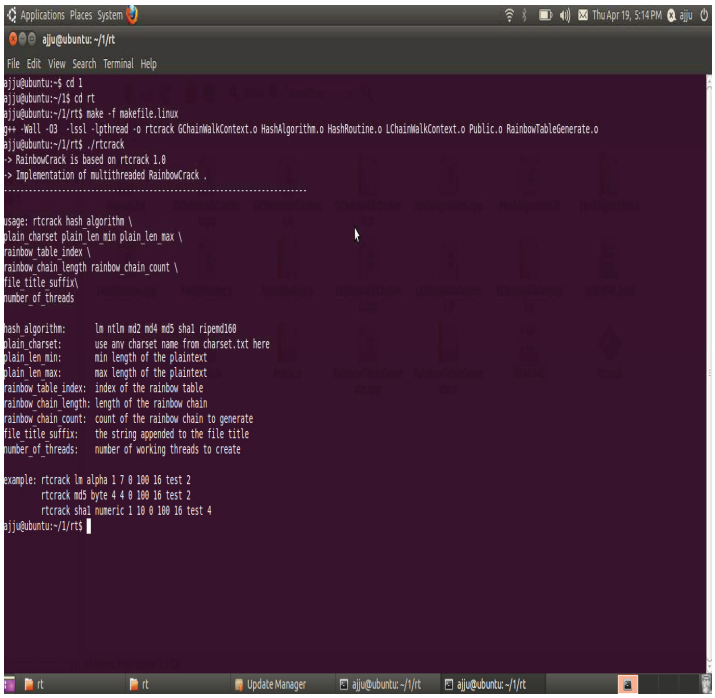

Fig.9. To generate Rainbow Table give some default values.

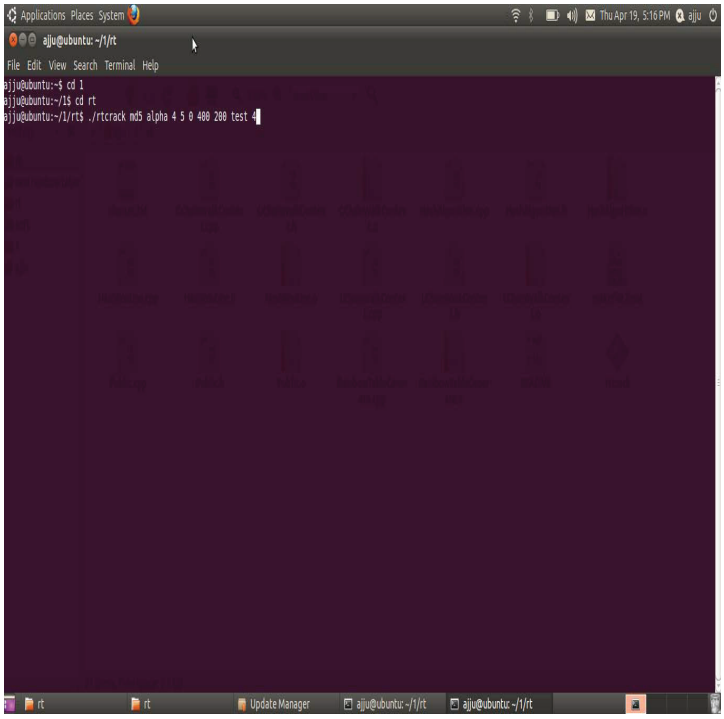

Fig.10. To generating Rainbow Table we select some parameters.

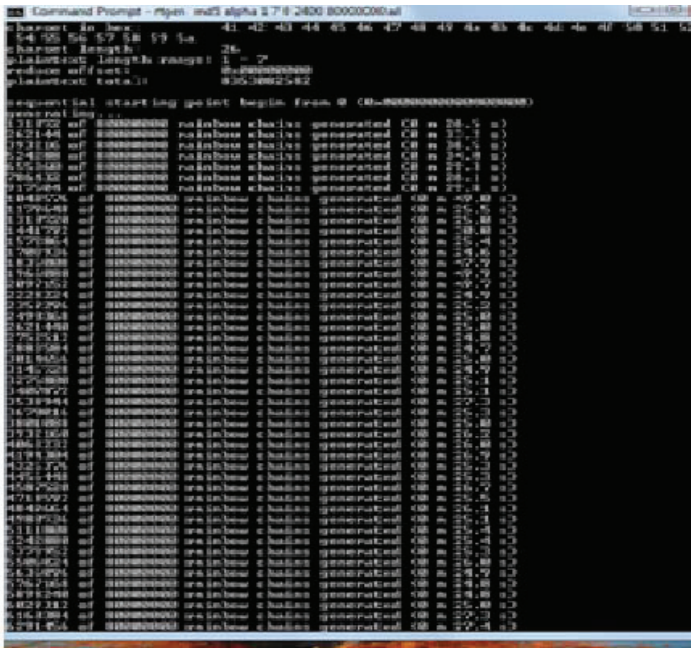

Fig.11.GeneratingRainbow table.

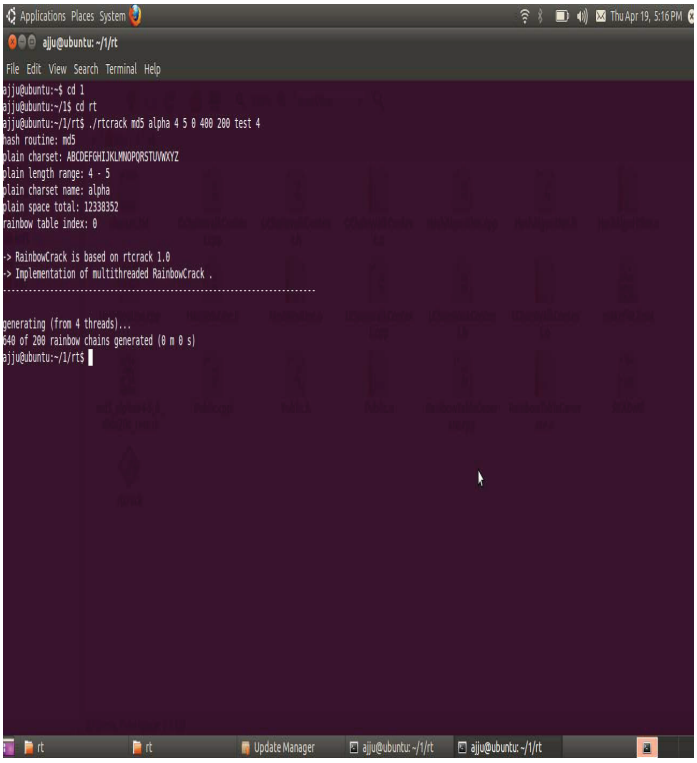

Fig.12. Rainbow table generated.

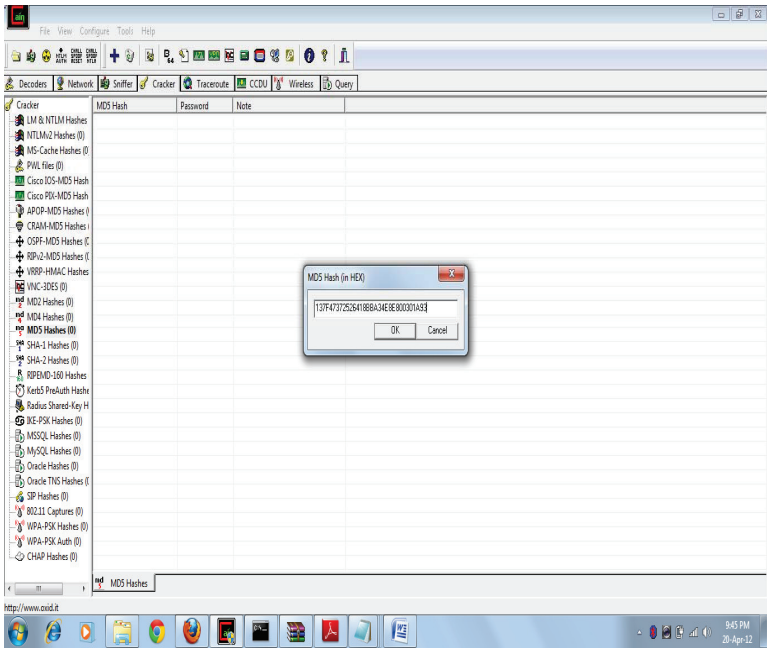

Fig.13. Give hash value.

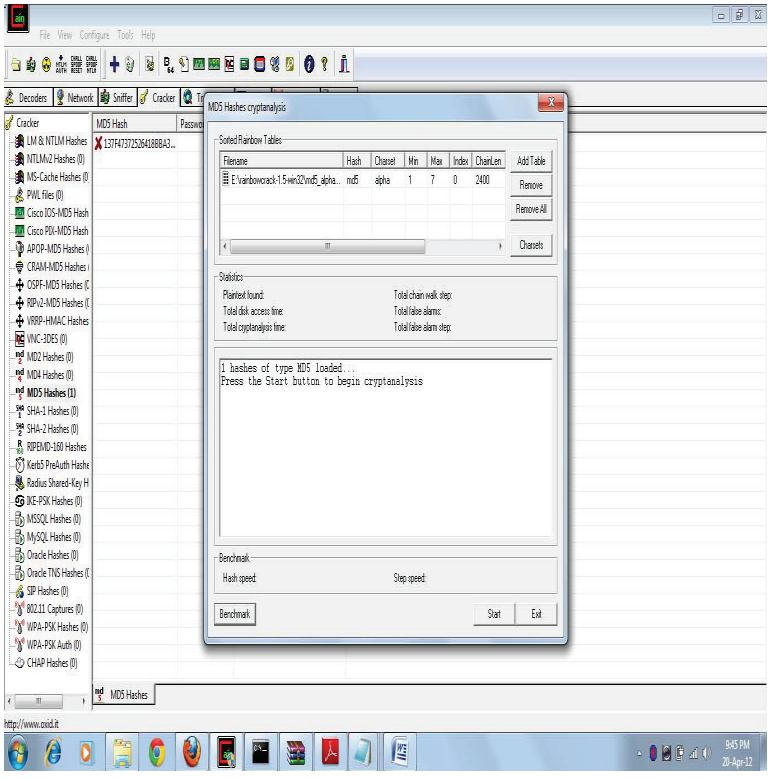

Fig.14. Select the Rainbow Table.

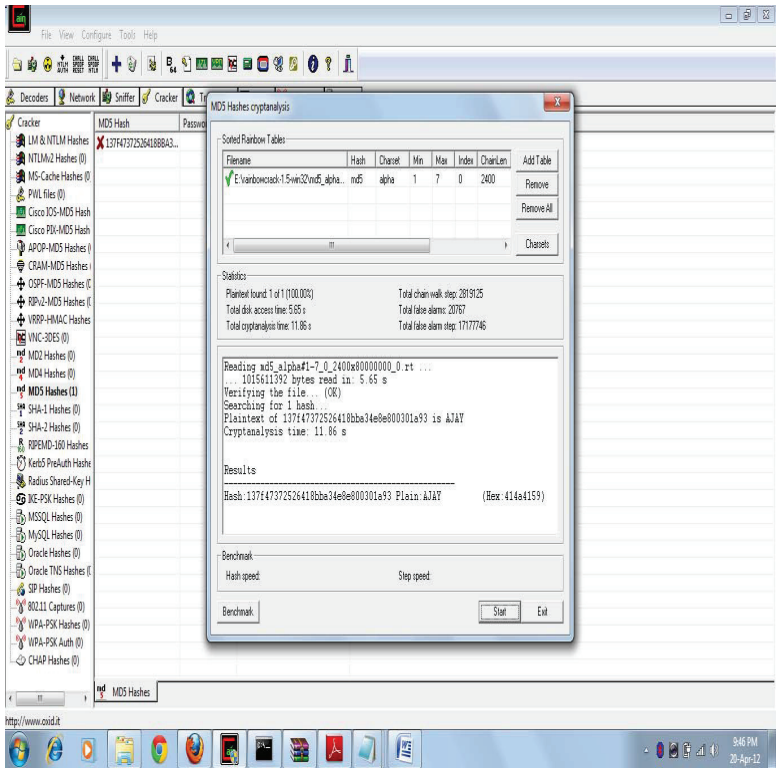

Fig.15. Final Result.

VI. Conclusion

By observing the results table, it is known thatcomputational time and cryptanalysis time increases as the key space increases. Rainbow Tables directly attack on hash algorithm, not on the password length so the passwords can be cracked easily even if the size of the password is more. Rainbow Tables are built once, and are used many times; it is fast and perfect for cracking hashes. One of the limitation of Rainbow Table is it is limited for reverse fixed length messages.Another limitation is table is not generated for all keys in key space i.e. it is probabilistic in nature as a result 100% success is not guaranteed. But higher success rate can be achieved by carefully choosing chain length and count. So it is concluded that Rainbow Table has a lot of scope for improvement and is evolving and many researchers are still working in this domain.

References

[1] Kostas Theoharoulis,IoannisPapaefstathiou, “Implementing Rainbow Tables in high end FPGAs for superfast password Cracking”, International Conference on field programmable Logic and applications(FPL),ISBN:978-1-4244-7842-2, Aug. 2010.

[2] Orhun Kara “Pre-images of Hash Functions Through Rainbow Tables” IEEE ACM Transactions on Information and System Security ISBN 978-1-4244-5023-7/09/September, 2009.

[3] N. Mentens,L Batina,B. Preneel and I. Verbauwhede,”Cracking Unix Passwords using FPGA plateforms”,SHARCS’05, Paris,February 2005.

[4] Ferguson,Neils,Bruce Schneier,”Practical Cryptography”, Indianpolis:John Wiley & Sons ISBN:0471-22357-3 June 2004.

[5] Oechslin Philippe,”Making a Faster Cryptanalytical Time-Memory Trade-Off”,Lecture Notes in Computer Science. Santa Barbara,California,USA;Springer.ISBN:3-540-40674 -3,August 2003.

[6] M.E. Hellman,”A Cryptanalytic Time Memory Trade-Off” IEEE Transactions on Information Theory,vol. 26,July 1980,pp.401-406.

[7] RFC page on MD5: http://tools.ietf.org/html/rfc1321.

[8] RFC page on SHA1: http://tools.ietf.org/html/rfc3174.

[9] RFC page on NTLM: http://tools.ietf.org/html/rfc4559.

[10] www.about.com/internet/networksecurity.

[11] More on Rainbow Tables: http://www.freerainbowtables.com.

[12] http://en.wikipedia.org/wiki/Rainbow_table.

[13] http://www.caihongbiao.org/jaieshao/Rainbowtable.html.

[14] http://www.thefullwiki.org/Rainbow_table.