UNICRYPT: A CONSTRUCTIVE APPROACH TOWARDS RAINBOW TABLE VULNERABILITY

Mohit Dagar[1], Nandit Saini[2], Himanshu Naresh[3], Ashish Sankla[4]

[1]Student, Computer Science Department, G.B Pant Govt. Engineering College, New Delhi, India
[2]Student, Computer Science Department, G.B Pant Govt. Engineering College, New Delhi, India
[2]Student, Computer Science Department, G.B Pant Govt. Engineering College, New Delhi, India
[4]Assistant Professor, Computer Science Department, G.B Pant Govt. Engineering College, New Delhi, India

Abstract

This project shows how we can eliminate the threat of password cracking by rainbow table. In this project we had made an encipher which encrypts our message or password in such a way that it becomes impossible to make a complete rainbow table for it which thus protects us from rainbow table attack used by professional hackers to crack the password. The encipher encrypts our message such that a different hash value is created every time, even if you encrypt the same message more than one time.

Key Words: *Cryptography, unicrypt, brute force attack, LM, NTLM, rainbow tables, dictionary attack.*

--***--

1. INTRODUCTION

The current authentication system which asks for the password and user name for authentication while logging into a system is our current front line of defence against any intruder who has the physical access to the device. A lot of work is done in this field about how the system database should store the login credentials of different user so that an intruder will not be able to determine the passwords even if he get access to the file of database in which the login credentials are stored.

Today's login authentication system suffer from a major vulnerability of the possibility of **rainbow table attack** because for a unique password the hash created every time is same, this gives the hacker the possibility of making a rainbow table and checking any given hash against it.

1.1. EXISTING TECHNIQUES

1.1.1. LM Hash

LAN Manager [1], or LM, is an authentication protocol designed (at its time) to maximize password security in a Windows-based environment. The LM protocol was first used in Microsoft's LAN Manager Product a very long time ago and is still the authentication protocol of choice for older operating systems, such as Windows 95 and Windows NT 3.51 and earlier.

1.1.2. NTLM Hash

Later, when Windows NT was introduced, LM was enhanced and renamed the NTLM [2] authentication protocol. NT LAN Manager (NTLM) is the Microsoft authentication protocol that was created to be the successor of LM. NTLM was accepted as the new authentication method of choice and implemented with Windows NT 4.

The creation of an NTLM hash (henceforth referred to as the NT hash) is actually a much simpler process in terms of what the operating system actually does, and relies on the MD4 hashing algorithm to create the hash based upon a series of mathematical calculations. After converting the password to Unicode, the MD4 algorithm is used to produce the NT hash. In practice, the password “PassWord123”, once converted, would be represented as “67A54E1C9058FCA16498061B96863248”.

MD4 is considered to be significantly stronger than DES[6] as it allows for longer password lengths, it allows for distinction between uppercase and lowercase letters and it does not split the password into smaller, easier to crack chunks.

1.2. EXISTING ATTACKS

There are various attacks used by hackers to break into Windows. Some of them are described below:-

1.2.1. Guessing Attack

Any attacker attempting to find credentials by guessing likely passwords can be considered a guessing attacker. As the name suggest in guessing attack, the guessing attacker uses a series of password guesses to break into the user's PC. Attackers research the targeted user to enhance their guessing strategy. This attack usually consist the name of user's family member, date of birth, phone number and other general information.

A well-designed verifier can easily limit the Guessing Attack. A well-designed verifier will employ rate-limiting techniques to limit the number of guesses which can be made, for example by limiting the number of authentication attempts[3] in a given time period, forcing a user to reset his or her password after too many failed attempts. Android phones password lock is a good example of well-designed verifier.

Windows doesn't employ well-designed verifier. Thus in Windows system the guessing attacker can try as many passwords as he want. Dictionary attack is an advance version of the Guessing Attack.

SUCCESS RATE	:	LOW
PROCESSING TIME	:	LOW
POPULARITY	:	LOW

1.2.2. Brute Force Attack

Brute Force consists of systematically checking all possible keys or words until the correct one is found. In the worst case, this would involve traversing the entire search space. This method is very fast when used to check all short passwords, but for longer passwords other methods such as the dictionary attack are used because of the time a brute-force search takes.

Cane and Abel [7] is a well-known Brute force tool for Windows that requires the hash value from the SAM. When targeting multiple users at same time the Hacker usually use reverse brute-force attack. In a reverse brute-force attack, a single (usually common) password is tested against multiple user-names or encrypted files. The process may be repeated for a select few passwords. In such a strategy, the attacker is generally targeting a large number of non-specific users. It then generates and compares the hash value of all the possible keys or password until the correct one is found out.
Today two emerging technologies have proven their capability in the brute-force attack of certain ciphers. One is modern graphics processing unit (GPU) technology; the other is the field-programmable gate array (FPGA) technology.

SUCCESS RATE: HIGH
PROCESSING TIME: ULTRA HIGH (YEARS FOR LARGE PASSWORD)
POPULARITY: MEDIUM

1.2.3. Dictionary Attack

A dictionary attack uses a targeted technique of successively trying all the words in an exhaustive list called a dictionary (from a pre-arranged list of values).In contrast with a brute force attack, where a large proportion key space is searched systematically, a dictionary attack tries only those possibilities which are most likely to succeed, typically derived from a list of words for example a dictionary (hence the phrase dictionary attack). Generally, dictionary attacks succeed because many people have a tendency to choose passwords which are short (7 characters or fewer), such as single words found in dictionaries or simple, easily predicted variations on words, such as appending a digit. However these are easy to defeat. Adding a single random character in the middle can make dictionary attacks untenable. Unlike Brute-force attacks, Dictionary attacks are not guaranteed to succeed.

Popular tools that are used for carrying out Brute-Force Attacks are *Brutus, Cain and Abel, Crack, Aircrack-ng, John the Ripper.*

SUCCESS RATE	:	HIGHLY DEPENDANT ON DICTIONARY
PROCESSING TIME	:	MEDIUM OR HIGH DEPENDANT ON DICTIONARY
POPULARITY	:	HIGH

1.2.4. Rainbow Tables

A rainbow table is a pre-computed table for reversing cryptographic hash functions, usually for cracking password hashes. Tables are usually used in recovering a plaintext password up to a certain length consisting of a limited set of characters. It is a practical example of a space/time trade-off, using less computer processing time and more storage than a brute-force attack which calculates a hash on every attempt, but more processing time and less storage than a simple lookup table with one entry per hash.

Tools which are used to carry out Rainbow Table attack are *Ophcrack and Herin's Boot*.

SUCCESS RATE	:	HIGHLY DEPEND ON PRE-COMPUTED TABLE
PROCESSING TIME	:	LOW
POPULARITY	:	AMAZINGLY HIGH IN WINDOWS

2. DRAWBACKS OF EXISTING TECHNOIQUES

There are various major weaknesses in the NTLM [4] hashes currently used by Microsoft in Windows 7, XP, Vista which are given as follows:-

- *NTLM* can use a maximum of 14 characters to create its stored hash. These 14 characters are split into two seven-character strings. Crypto-graphically, it is reasonably easy to brute force attack [5] two seven-character strings with modern computers.

- NTLM cannot use lowercase letters. It converts all lowercase letters to uppercase before creating the hash. This reduces the character set for the password, making brute force attacks far more likely to succeed.

- The hash algorithm used to store passwords became well known. That allowed attackers to guess users' passwords by running password guesses through the hash until the result matched the result stored in the SAM. Because the algorithm remained constant, large libraries of hashed passwords could be stored and used to quickly attack a SAM.

- The created hash for a unique password is unique and not many hashes for a single unique password are created.

With this being the case, it is possible for an attacker to generate what are called rainbow tables. Rainbow tables are actually tables containing every single hash value for every possible password possibility up to a certain number of characters. Using a rainbow table, you can simply take the hash value you have extracted from the target computer and search for it. Once it is found in the table, you will have the password. As you can imagine, a rainbow table for even a small number of characters can grow to be very large, meaning that their generation, storage, and indexing but once they're out there, every attacking computer can leverage those tables to make their attacks on hashed passwords that much more potent. The smallest rainbow table available is the basic alphanumeric one, and even it is 388 megabytes. Even that smallish table is remarkably effective.

3. Proposed Technique

3.1. Project Design

The design of our project is very simple. It consists of two text boxes; first textbox is for giving an input text which can either be a plaintext or a cipher text and in the second textbox the output is generated which is a cipher text during Encryption Phase and a plaintext during Decryption Phase.

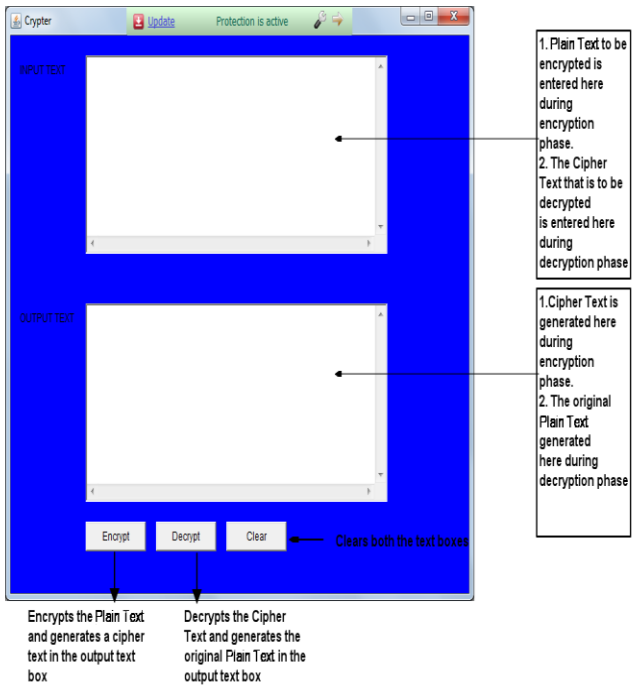

Crypter
Update
Protection is active
INPUT TEXT
OUTPUT TEXT
Encrypt
Decrypt
Clear
1. Plain Text to be encrypted is entered here during encryption phase.
2. The Cipher Text that is to be decrypted is entered here during decryption phase
1.Cipher Text is generated here during encryption phase.
2. The original Plain Text generated here during decryption phase
Clears both the text boxes
Encrypts the Plain Text and generates a cipher text in the output text box
Decrypts the Cipher Text and generates the original Plain Text in the output text box

Figure 1 *Design and Components of the Project*

3.1.1 Working of the Project as a whole

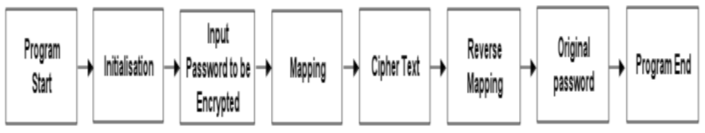


Figure 2 *Block Diagram for the working of the Project*

From the above block diagram we can see that our project consists of various phases which includes Initialisation phase, input plaintext to be encrypted, then mapping is done which creates a cipher text.
Then during decryption phase the cipher text is converted into Original plaintext which was entered by the user by reverse mapping.

3.1.2. Initialisation Phase

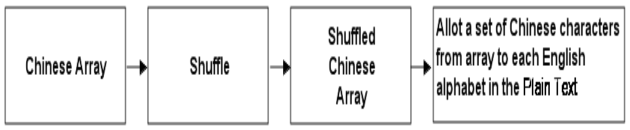

Chinese Array
Shuffle
Shuffled Chinese Array
Allot a set of Chinese characters from array to each English alphabet in the Plain Text

Figure 3 *Initialisation Phase*

From the block diagram given above it is clear that when the program is started and goes in its initialisation phase it shuffles the sequential (alphabetical) Chinese character Array to create a shuffled Chinese character Array according to which the one to many mapping is done during the encryption phase. This shuffling of the array is done by treating each character in the array as a separate entity just like a shuffling a deck of cards. After the shuffling of the Chinese array, a Chinese character set or block of fixed size is extracted from this shuffled array and is allotted to each English alphabet character.

3.3. Mapping

A mapping in the scope of this project is a mathematical relation such that each ASCII character is associated with a Chinese character. This mathematical relation is **one to many** in nature that is for every ASCII character a Chinese Character set of fixed size is allotted at the time of initialisation and then from this character set, only one Chinese character is chosen at random. After that the ASCII character in the password is mapped to this chosen Chinese character. This random choosing of Chinese character is done for each ASCII character in the password and the set for each ASCII character in the password is different. This mapping algorithm ensures that an ASCII character is mapped to a different Chinese Character even if that same letter is encrypted again because it is picking up random character from the fixed set allotted at the time of initialisation. That means unique hash or cipher text is created even if the same password is encrypted twice.

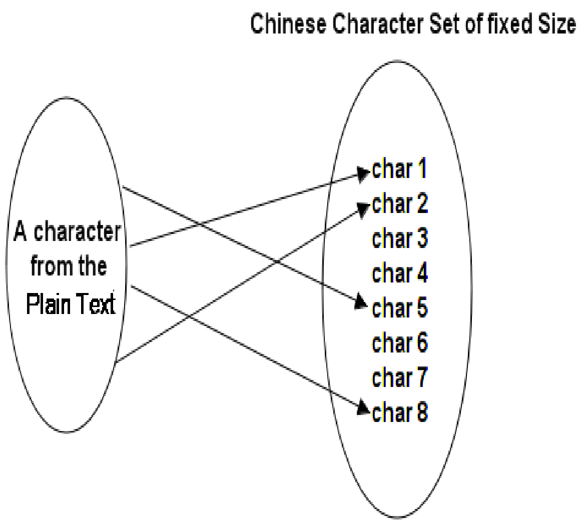

Chinese Character Set of fixed Size
A character from the Plain Text
char 1
char 2
char 3
char 4
char 5
char 6
char 7
char 8

Figure 4 *One To Many Mapping of characters*

3.4. Encryption Method

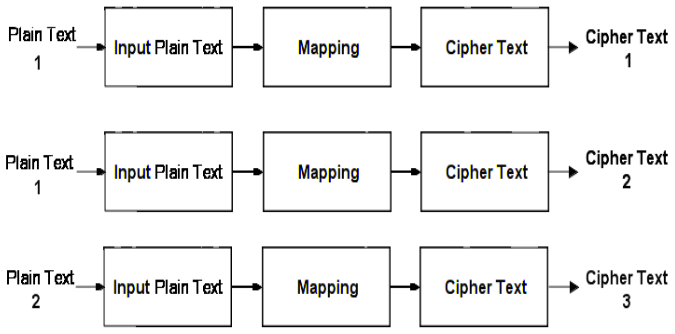
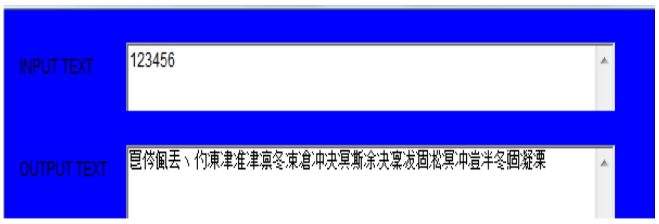


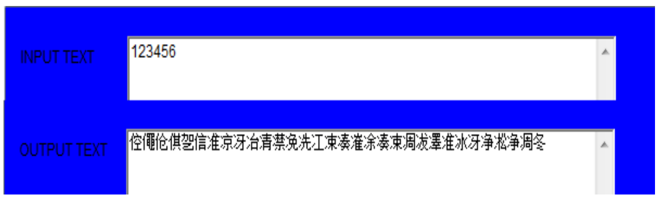
Figure 5 *Encryption Phase*

In this phase the password that is to be encrypted is fetched from the input text box and then according to the shuffled Chinese Character Array the mapping is done to generate a unique cipher text in the output text box even if the same password is encrypted again. The speciality of this algorithm is that for the same password different hash or cipher text is created even if it is encrypted more twice and not a unique hash which shows one to one relationship between the plain text character and cipher text.


INPUT TEXT 123456

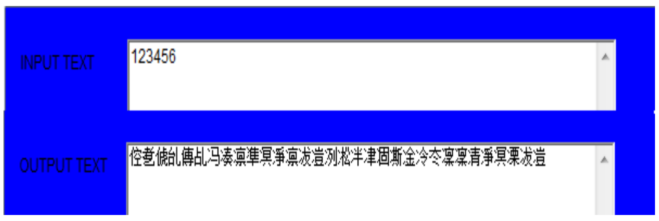
OUTPUT TEXT 叵侉颪丟丶仢東津准津凛冬東滄沖決冥斯涂決凜泼固凇冥沖豈半冬涸凝凓

Figure 6 *Plaintext "123456" encrypted 1st time*


INPUT TEXT 123456

OUTPUT TEXT 倥僶伧倛垽信准京冴冶青禁浼冼冮涷凑凗凃凑凍凋泼罿准冰冴净凇净凋冬

Figure 7 *Plaintext "123456" encrypted 2nd time*


INPUT TEXT 123456

OUTPUT TEXT 倥巻傀乱傳乩冯凑凛凖冥净凛泼豈冽凇泮津涸斯凎冷冭凜凜青净冥凓泼豈

Figure 8 *Plaintext "123456" encrypted 3rd time*

3.5. Decryption Method

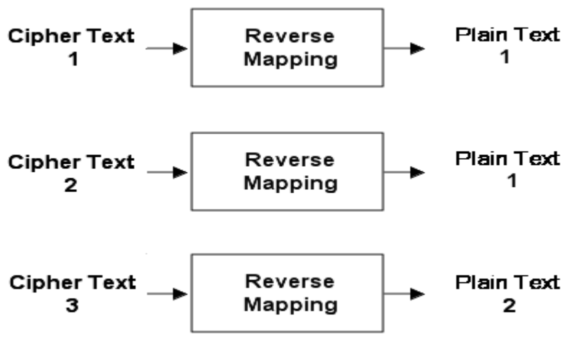
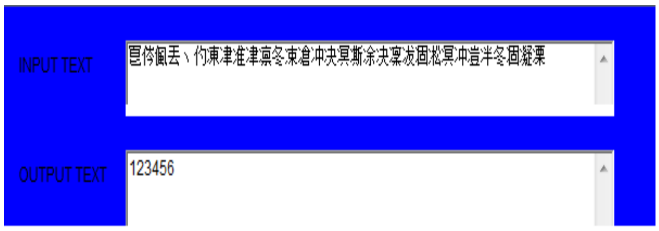


Figure 9 *Decryption Phase*

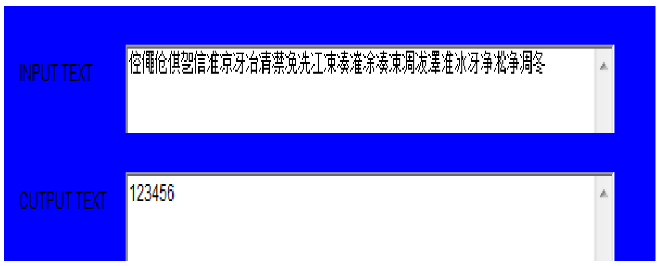
In this phase the cipher text is fetched from the input text box and then the original password is generated in the output text box of the application. This original password is generated by doing the reverse mapping of the cipher text. When the program is started again the initialisation phase is performed and then the Chinese Character Array is shuffled again. This shuffled Chinese Character Array generated now is different from the one created before. That means the allotted Chinese character set of fixed size is also changed and then from that fixed size character set a Chinese character is chosen at random. Now the mapping of the alphabet character in the password is done to that selected Chinese character.



INPUT TEXT 叵佟佩丟丶仢凍凁准凁凛冬凍凔冲决冥凘凃决凛冹凅凇冥冲凒冸冬凅凝凓

OUTPUT TEXT 123456

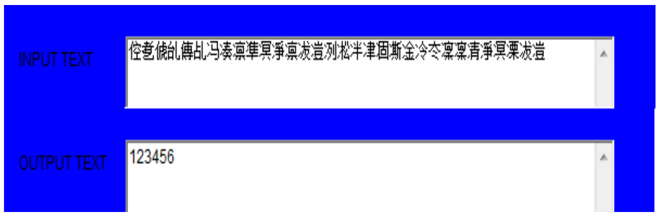
Figure 10 *Decryption of Cipher text generated 1st time*



INPUT TEXT 倥僶伦倛塑信准凉冴冶凊禁凂冼冮凍凑凗凃凑凍凋冹凙准冰冴净凇净凋冬

OUTPUT TEXT 123456

Figure 11 *Decryption of Cipher text generated 2nd time*



INPUT TEXT 倥耋僥乿傳乩冯凑凛凖冥净凛冹凒冽凇冸凁凅凘凎冷冬凛凛凊净冥凓冹凒

OUTPUT TEXT 123456

Figure 12 *Decryption of Cipher text generated 3rd time*

4. IMPLEMENTATION

All the phases of our poject are implemented in Java Programming using the NetBeans IDE. All the Chinese Characters are represented by using **Unicode**, which is a computing industry standard for the consistent encoding, representation and handling of text expressed in most of the world's writing systems that provides 110,000 characters which is far more than the ASCII encoding which is limited to only 128 characters. In this project we are using Chinese characters which range from 3000-4000 in numbers. So by such a large set of characters we can create a unique hash

Now phase wise details of the implementation are given below.

4.1. Initialisation

In the initialisation phase two things happen:

- Shuffling
- Allotment of Chinese character set

4.1.1. Shuffling

The shuffling algorithm randomly permutes the specified list using a default source of randomness. What it does is that it traverses the list backwards, from the last element up to the second, repeatedly swapping a randomly selected element into the "current position". Elements are randomly selected from the portion of the list that runs from the first element to the current position, inclusive.

4.1.2. Allotment of the Chinese Character Set

From the shuffled Chinese array we allot the first N characters to an ASCII character “a” then the next N characters to the alphabet “b” and so on up till “z” and capital letters and numbers. This thing is implemented with the help of indexing of the array and let’s say for example N=5. The indexing starts from 0 to access the first element in an array so we take the Chinese characters between 0 to 4th index from the shuffled Chinese Character array and then allot it to “a” ,then the Chinese characters from 5th to 9th index of the shuffled array is allotted “b” and so on. This allotting is done for capital letters and numbers as well.

4.2. Mapping

In mapping we take a random Chinese Unicode character from the set that was allotted to each ASCII character in the initialisation phase and then replace the encountered ASCII character in the password with that randomly selected Chinese Unicode character. This means that if multiple a's are present in the password to be encrypted it may or may not map to the same Chinese Unicode character as the previous "a" in the password. This ensures one to many relations of the characters.

4.3. Encryption

After the random character that is chosen from the set allotted to a alphabet or number has been calculated then that alphabet or character is replaced with the random Chinese character. Then at last the output is generated in the output text box. As the Chinese characters are there in the cipher text hence Unicode is used in order to implement it programmatically as told earlier.

Algo -1: UNICRYPT- Encryption

----

1) Choose a Plaintext password $P(P_1P_2\ldots..P_n)$ of length n.
2) If n is less than 32 characters in length then pad it with spaces to make the total length of 32 else if n is equal or greater than 32 no padding is needed.
3) Now our string password to be encrypted is P + padding.
4) If padding is there then replace (32-n) padded spaces ,where n is the length of the password provided by the user with (32-n) random Chinese characters from a predefined set of 50 Chinese characters $C_1,C_2,\ldots\ldots C_{50}$.
5) Now the padding portion of our string has been encrypted.
6) Now for each P_i in P we assign 1 random Chinese character C_i from a set of 10 random Chinese Characters.
7) Now our whole password has been encrypted.

The number of possible permutation of cipher texts generated for a password of n length is given below

For n<32

$({}^{50}C_{32-n}).(10^n)$ *possibilities of cipher text combinations*

For n>=32

10^n possibilities of cipher text combinations

4.4. Decryption

In this phase the reverse mapping of the cipher text is done to get the original password. The reverse mapping is also implemented by the indexing. What we do can be understood by an example that if the index of the cipher text Chinese character is between the 0 – 4th index of the shuffled array or the index range of the set which was allocated for "a" or any alphabet or number then that cipher text will be replaced by "a" or that other alphabet or number and this is done for each cipher text character otherwise nothing is replaced. Hence we get the original password.

5. RESULTS

5.1. Encryption Phase Results

Table -1: Output Cipher Text of some most common passwords during Encryption Phase

Input Text	**Output Text**
123456	俳儇儁傕儠书浼冰凝凍澿澿津況冻凍凓泮夵涷凃凓淒夵凝夵准洌湮湮湮汥
123456	共伂俊内予亣減泯涑凜浼淦凋津凇决冶凗凗凙津泮准凓澌冲凋冯净冻澿津
123456	丽傚儽严侃儸泮決洗滄江凑凝冲泯淨湮洽淒凝冯凙涷洌湮涸滄冰況津津湮
password	亼伣儶丕俻兓丏伋净凇冲夵況夵凉汥淦滄冻准凃凗准浼冰冴減凙溟凙冯滄
password	傛乎儹停倠侟傏伐凋溟凙淦凘凘凊凛涸滄冯凗凘冷淦洽凖凍冻減淦净夵冰
password	件偦岂乳僄儶傦兽凜江冷冻冯凙凍涸凇凑洽凖淒冶冲冰溟凗冬凉津凒洌湮
qwerty	乘儀俫侹偖仉溟凇决凗凒涷冯凖冬凒冯冬凍淦凊洌凋凘准泯凒洗冴淨冱況
qwerty	偓侗傷儆傲儈泮凜冯决决凝冱决凘滄净泯凊凗泯冴減涷淦冯冯浼凜夵淨冰
qwerty	俵伭以儞傲儕滄凒冷净涷准津凘凉准浼冬凖澿泮凌湮溟凝凋冲凗洗冶凉涸

abc123	偺佝傱俳倣儒决凈冽冻凛凚冭冷凑凍凕凂冻凅净冰凍凇凂冽凙冰凐凒凑冲
abc123	伣偎兑亨乾儑冻凙冸凛冯况凂净冼减冹冬冰凗准冬津凇准凉凒冾凚凓凄凂
abc123	偦佝儉亨习儰冲凛冹凂凘冹冬凌凛冬减冯凈准凛凄凚凙冬凛冹凓冷冺冱冴
ilove you	儷偽儁乀儭乇偶丞凘凛津凌凘冼凗凛冮凄凍凊冴凅决凗准冬凝冼凚凂凔冱
ilove you	傉兜儲乽乳低兟偊凓减冴凍冷冹凖冾凂凋凊凊凓津冸决冭凑冯凝冶冻凇凝
ilove you	伡令僧僂僶儈儲倖决准凄冼冰凇凋凍凓冭凎凗冴况冺冻凕凍凘凈冮凋况冾
111111	共丁丁亡亨佲冱凃冶凛冻况凑凃况冴冸凌凗减冭决冸凚冸冯凋冺冼凛凔冸
111111	伶亨伶俳亨共冴凋冬冭减冹冻冶冭凛净凛凂冰冲凅冷冶凄凉凕冱凙津冬冲
111111	丁伶丽佲亨亨凊凊冶凒凎凛凑凃冬凈冯凃减冱冯凑冹凝凓凌冽冷凝凒冮凖

5.2. Decryption Phase Results

Table -2: Original *Plain Text generated from decrypted Cipher Text*

Input Text	**Output Text**
俳僽儒偺儠书凂冰凝凍凚凚津况冻凍凓冸冭凍凃凓凄冭凝冭准冽凐凐凐冹	123456
共伂俊内予亣减冺凍凛凂凎凋津凇决冶凗凗凙津冸准凓凘冲凋冯净冻凚津	123456
丽倣儑严侃僱冸决冼凔冮凑凝冲冺凈凐冾凄凝冯凙凍冽凐凅凔冰况津津凐	123456
亼伣儶丕傄兟丐仭净凇冲冭况冭凉冹凎凔冻准凃凗准凂冰冴减凙凕凙冯凔	password
傛乎儹停倠侟傏伐凋凕凙凎凘凘凊凛凅凔冯凗凘冷凎冾凖凍冻减凎净冭冰	password
件偦岂乳僄儁偦兽凛冮冷冻冯凙凍凅凇凑冾凖凄冶冲冰凕凗冬凉津凒冽凐	password
乘僟倈侹偖仉凕凇决凗凒凍冯凖冬凒冯冬凍凎凊冽凋凘准冺凒冼冴凈冱况	qwerty
偓侗偒傚儆儈冸凛冯决决凝冱决凘凔净冺凊凗冺冴减凍凎冯冯凂凛冭凈冰	qwerty
傫伭以偖儆傡凔凒冷净凍准津凘凉准凂冬凖凚冸凌凐凕凝凋冲凗冼冶凉凅	qwerty

偺佝傱俳倣儒决凈冽冻凛凚冭冷凑凍凕凂冻凅净冰凍凇凂冽凙冰凐凒凑冲	abc123
伣偎兑亨乾儑冻凙冸凛冯况凂净冼减冹冬冰凗准冬津凇准凉凒冾凚凓凄凂	abc123
偦佝儉亨习儰冲凛冹凂凘冹冬凌凛冬减冯凈准凛凄凚凙冬凛冹凓冷冺冱冴	abc123
儷偽儁乀儭乇偶丞凘凛津凌凘冼凗凛冮凄凍凊冴凅决凗准冬凝冼凚凂凔冱	ilove you
傉兜儲乽乳低兟偊凓减冴凍冷冹凖冾凂凋凊凊凓津冸决冭凑冯凝冶冻凇凝	ilove you
伡令僧僂僶儈儲倖决准凄冼冰凇凋凍凓冭凎凗冴况冺冻凕凍凘凈冮凋况冾	ilove you
共丁丁亡亨佲冱凃冶凛冻况凑凃况冴冸凌凗减冭决冸凚冸冯凋冺冼凛凔冸	111111
伶亨伶俳亨共冴凋冬冭减冹冻冶冭凛净凛凂冰冲凅冷冶凄凉凕冱凙津冬冲	111111
丁伶丽佲亨亨凊凊冶凒凎凛凑凃冬凈冯凃减冱冯凑冹凝凓凌冽冷凝凒冮凖	111111

6. CONCLUSION

A new encryption algorithm is developed which encrypts the password in Unicode and eliminates the vulnerability of creating a rainbow table which leaves our system open to rainbow table attack. This project successfully eliminates the rainbow table attack which can be performed even by a novice hacker on Windows Users by producing a hash in such a way that it is unique even if you encrypt the same password more than one time. It can also be used in any other authentication system suffering from the same vulnerability of the possibility of rainbow attack.

7. REFERENCES

[1]. G. Zorn, S. Cobb, "Microsoft LM: Microsoft PPP CHAP Extensions" Microsoft Corporation, October 1998.

[2]. Eric Glass, “The NTLM Authentication Protocol and Security Support Provider", 2006.

[3]. Joseph Bonneau, “Guessing human-chosen secrets” University of Cambridge, Technical Report Number - 819, p. 11,May 2012.

[4]. Meetika Malhotra, Bhushan Dua,” A Review of NTLM Rainbow Table Generation Techniques” Publisher: Global Journals Inc. (USA), Volume 13 Issue 7 Version 1.0 Year 2013, Online ISSN: 0975-4172 & Print ISSN:0975-4350.

[5].William Stallings. 2010. Cryptography and Network Security: Principles and Practice (5th Ed.). Prentice Hall Press, Upper Saddle River, NJ, USA.

[6]. Sombir Singh, Sunil K. Maakar, Dr.Sudesh Kumar ,"Enhancing the Security of DES Algorithm Using Transposition Cryptography Techniques", Volume 3, Issue 6, June 2013 ISSN: 2277 128X.

[7]. Jørgen Blakstad , Rune Walsø Nergård , Martin Gilje Jaatun , Danilo Gligoroski ,"All in a day's work:Password cracking for the rest of us", Presented at the NISK-2009 conference.

[8]. Mudassar Raza, Muhammad Iqbal, Muhammad Sharif and Waqas Haider," A Survey of Password Attacks and Comparative Analysis on Methods for Secure Authentication", World Applied Sciences Journal 19 (4): 439-444, 2012 ISSN 1818-4952.

BIOGRAPHY

Mohit Dagar was born in Issapur village, Delhi in March 1993.He is currently pursuing bachelor of technology (B.Tech.) in Computer Science and Engineering from Govind Ballabh Pant Government Engineering College, Okhla, New Delhi. He has gathered a lot of knowledge about the various cryptography techniques and their strengths and weaknesses by attending many workshops, and reading different research paper related to ethical hacking, windows security, and Linux security.

Nandit Saini was born in New Delhi in July 1993.He is currently pursuing bachelor of technology (B.Tech.) in computer Science and Engineering from Govind Ballabh Pant Government Engineering College, Okhla, and New Delhi. He has read many research papers related to cryptography and security. His current area of interest include Windows security, Java programming and online competitive programming.

Himanshu Naresh was born in New Delhi,in August 1993.He is currently pursuing bachelor of technology (B.Tech.) in Computer Science and Engineering from Govind Ballabh Pant Government Engineering College, Okhla, New Delhi. He is a Cisco Certified Network Associate (CCNA). His current field of interest includes network security, windows security and C Programming.

Ashish Sankla, New Delhi, 31 july 1986. M-tech in *Computer Science Engineering*, from University School of Information and Technology, Guru Gobind Singh Indraprastha University, Delhi. B-Tech in *Computer Science Engineering* from Guru Premsukh Memorial College of Engineering, Guru Gobind Singh Indraprastha University, Delhi. He is working as Assistant Professor, CSE Dept., at G. B Pant Govt Engineering College, from January-2013. He qualified UGC-NET-2012 and scholar of his M-Tech batch..