Information Security Project Report

Group Members

Kanza Latif	19I-0550
Mubrra Asma	19I-0699
M.Haris	19I-0740

Section:

CS-A

Submitted to:

Sir Abdullah Abid

Project Overview:

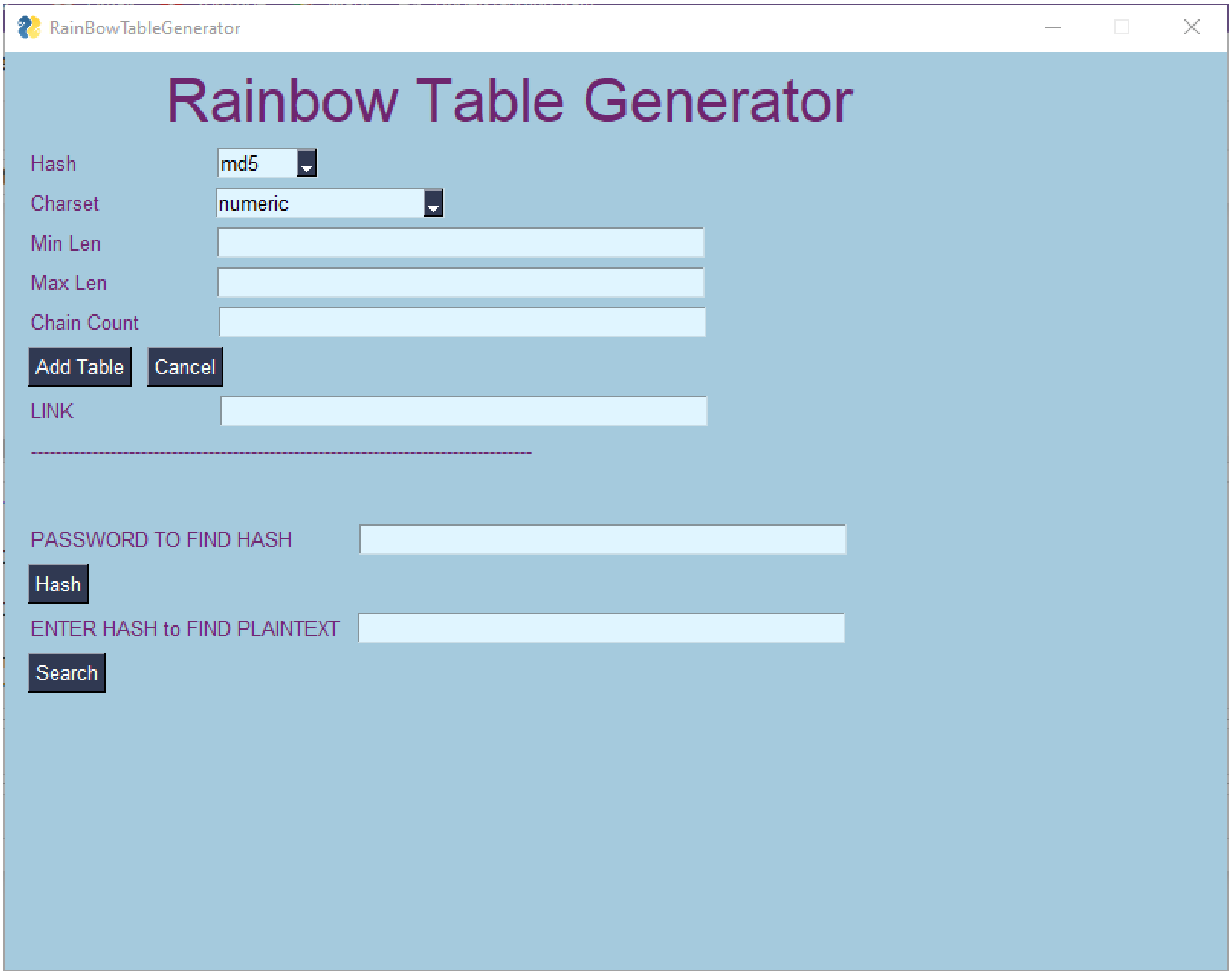
Passwords in the system are not stored as simple plain text whenever a user is signed up, rather hashed using encryption. A rainbow table is a lookup table that stores precomputed hashes of the plaintext of the selected hashing algorithm and charset. These hashes are then stored against their plaintext. For cracking a hash of the plain text is matched with the start of the chain, if matches then the entire chain is searched till the required plain text is found.

Working

Based on the hashing algorithm and charset selected, rainbow table generator starts calculating hashes.A hash function is a 1-way function, which means that it can’t be decrypted once hashed. Whenever a user enters a password, it is converted into a hash value and is compared with the already stored hash value. If the values match, the user is authenticated.Unlike bruteforce attack, which works by calculating the hash function of every string present with them, calculating their hash value and then comparing it with the one in the computer, at every step. A rainbow table attack eliminates this need by already computing hashes of the large set of available strings. For example, hash(1234) = “81dc9bdb52d04dc20036dbd8313ed055” so this hash is stored against its plaintext 1234. Depending upon the length of the chain, hash value is reduced and rehashed and chain is formed.

If the chain length is 2 then hash value reduced, reHash(81dc) = “ee399c9308de5cc0c5a5d87a84c064ef” this process is continued till we get the required chain length

UserInterface:



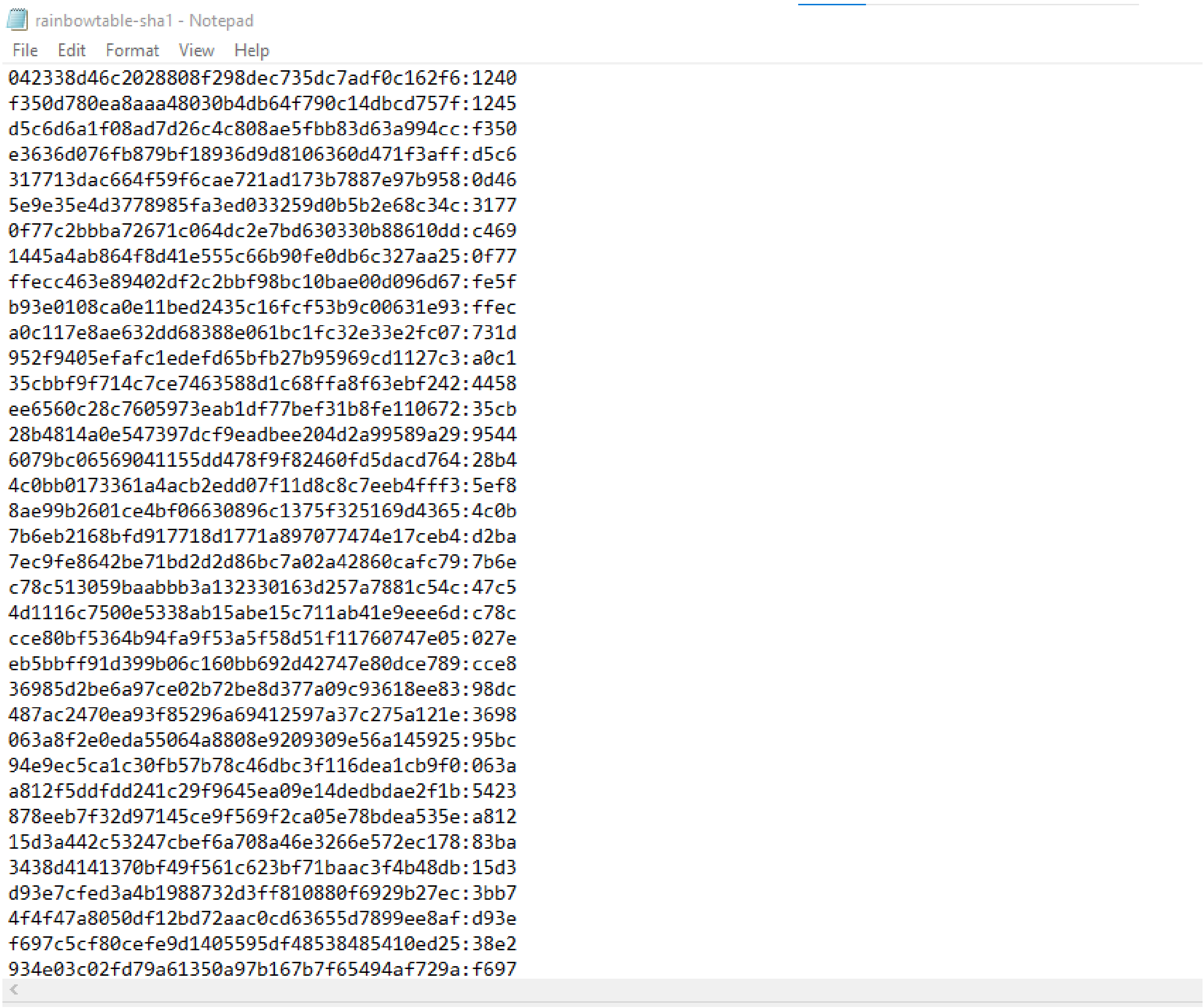
RainBowTableGenerator

Rainbow Table Generator

Hash: md5

Charset: numeric

Table Format:



```
042338d46c2028808f298dec735dc7adf0c162f6:1240
f350d780ea8aaa48030b4db64f790c14dbcd757f:1245
d5c6d6a1f08ad7d26c4c808ae5fbb83d63a994cc:f350
e3636d076fb879bf18936d9d8106360d471f3aff:d5c6
317713dac664f59f6cae721ad173b7887e97b958:0d46
5e9e35e4d3778985fa3ed033259d0b5b2e68c34c:3177
0f77c2bbba72671c064dc2e7bd630330b88610dd:c469
1445a4ab864f8d41e555c66b90fe0db6c327aa25:0f77
ffecc463e89402df2c2bbf98bc10bae00d096d67:fe5f
b93e0108ca0e11bed2435c16fcf53b9c00631e93:ffec
a0c117e8ae632dd68388e061bc1fc32e33e2fc07:731d
952f9405efafc1edefd65bfb27b95969cd1127c3:a0c1
35cbbf9f714c7ce7463588d1c68ffa8f63ebf242:4458
ee6560c28c7605973eab1df77bef31b8fe110672:35cb
28b4814a0e547397dcf9eadbee204d2a99589a29:9544
6079bc06569041155dd478f9f82460fd5dacd764:28b4
4c0bb0173361a4acb2edd07f11d8c8c7eeb4fff3:5ef8
8ae99b2601ce4bf06630896c1375f325169d4365:4c0b
7b6eb2168bfd917718d1771a897077474e17ceb4:d2ba
7ec9fe8642be71bd2d2d86bc7a02a42860cafc79:7b6e
c78c513059baabbb3a132330163d257a7881c54c:47c5
4d1116c7500e5338ab15abe15c711ab41e9eee6d:c78c
cce80bf5364b94fa9f53a5f58d51f11760747e05:027e
eb5bbff91d399b06c160bb692d42747e80dce789:cce8
36985d2be6a97ce02b72be8d377a09c93618ee83:98dc
487ac2470ea93f85296a69412597a37c275a121e:3698
063a8f2e0eda55064a8808e9209309e56a145925:95bc
94e9ec5ca1c30fb57b78c46dbc3f116dea1cb9f0:063a
a812f5ddfdd241c29f9645ea09e14dedbdae2f1b:5423
878eeb7f32d97145ce9f569f2ca05e78bdea535e:a812
15d3a442c53247cbef6a708a46e3266e572ec178:83ba
3438d4141370bf49f561c623bf71baac3f4b48db:15d3
d93e7cfed3a4b1988732d3ff810880f6929b27ec:3bb7
4f4f47a8050df12bd72aac0cd63655d7899ee8af:d93e
f697c5cf80cefe9d1405595df48538485410ed25:38e2
934e03c02fd79a61350a97b167b7f65494af729a:f697
```