I Introduction:

The protection of our personal data is crucial in the current digital era. Using solid, one-of-a-kind passwords for all of our online accounts is one of the greatest methods to safeguard oneself. Making and remembering several complicated passwords might be difficult, though. Password generators are useful in situations like these.

The application we'll be talking about is a password generator that secures passwords by using the SHA-256 hashing technique, which makes them difficult to guess or crack. The application also features a function that lets users create a rainbow table, which they may use to save hashed passwords for later use. Users may also submit a hash value into the programme's lookup function to get the associated password from the rainbow table by entering it.

Overall, this application offers a workable response to the difficulty of generating and maintaining safe passwords. Users may feel secure knowing that their passwords are well protected by employing a password generator that makes use of a powerful hashing algorithm and a rainbow table.

Password Generation:

The software initially specifies the quantity of passwords to be produced, their length, and an alphabet of lowercase English letters. Using the random.choice function to choose characters from the predetermined alphabet, it then creates a list of random passwords. A list containing the created passwords is kept.

Define the alphabet we'll be using to generate passwords.

alphabet = "abcdefghijklmnopqrstuvwxyz"

Define the number of passwords we want to generate.

num_passwords = 1000

Define the length of each password.

password_length = 8

Generate the passwords.

passwords = [] for i in range (num_passwords): # Generate a random password of the specified length, where password = "". join(random.choice(alphabet) for j in range(password_length)) Add the password to the list of passwords passwords.append(password)

Password Hashing:

The created passwords are then hashed by the application using the well-known cryptographic hash method SHA-256. It uses the encode() function to convert the password to bytes and the hashlib.sha256() method to apply the SHA-256 algorithm to the bytes. The list that results contains the hash values.

Hash the passwords using SHA-256.

hashed_passwords = [] for password in passwords: Hash the password using SHA-256. hashed_password = hashlib.sha256(password.encode()); hexdigest() # Add the hashed password to the list of hashed passwords. hashed_passwords.append(hashed_password)

Rainbow Table Creation:

The hashed passwords are stored in a rainbow table that the application creates. A precalculated table called a rainbow table is used to reverse cryptographic hash algorithms. It operates by first mapping each hash to a shorter value using a reduction function, and then utilising a series of reduction and hash functions to create a table of potential values that may be used to reverse a hash function.

The software creates the rainbow table by repeatedly going over the list of hashed passwords. For each hash value, it uses the first four characters of the hash as the key, and if the key isn't already in the table, it adds the key and the hashed password to the rainbow table.

Build the rainbow table.

rainbow_table = for hashed_password in hashed_passwords: # Generate the key for this hashed password by taking the first 4 characters of the hash key: hashed_password[:4]. # If the key is not already in the rainbow table, add it with the hashed password: rainbow_table[key] = hashed_password

Reduction Function:

Hash cracking techniques in cryptography employ a reduction function to condense the size of the hash table. By doing so, memory is conserved, and the password-cracking process is expedited. A reduction function reduces the output space of a hash function's output. The hash function, which is used to find the password, is iterated upon using the reduction function to provide a fresh starting point.

The programme's reduction function requires two inputs: a hash value and an iteration count. It takes the hash value's first four characters as a key and transforms them into an integer. The iteration number is then multiplied by this integer key to create a new integer key. The SHA-256 method is then used on the hex string once the integer key has been transformed back to it. The reduced hash value is the resultant hash value that only contains the first 8 characters.

This reduction function was used because it reduces the output space for the hash function, making it simpler to identify passwords that match. To guarantee that the same key is used for each iteration, the first 4 characters of the hash value are used as a key. It is challenging for an attacker to anticipate the next result of the hash function due to the addition of the iteration number to the key, which helps to establish a fresh beginning point for the following iteration.

To make sure that the result of the reduction function is a legitimate input for the SHA-256 method, the integer key is converted to a hex string. Since they offer enough entropy for the hash function and minimise the size of the output space, the first 8 characters of the resultant hash value are utilised as the reduced hash value.

This reduction function might have the issue of not being appropriate for all hash functions. The reduction function must be created to deal with the precise output size of the hash function since different hash algorithms have varying output sizes. Moreover, the reduction function may be subject to attacks that take advantage of patterns in the hash function due to the usage of a fixed iteration number.

Overall, the programme's reduction function is a useful method for minimising the output area of the hash function and accelerating the password-cracking process. However, it's crucial to select the proper reduction function for the particular hash function being used and to make sure that the reduction function is secure from attacks.

Discussion:

Comparing the technique described in this study to conventional password-based authentication solutions reveals a number of advantages. Its resistance to attacks that depend on the compromise of password databases is one of its key features. If an attacker acquires access to the password database in conventional password-based systems, they may quickly recover the passwords and use them to pose as authorised users. In contrast, the suggested solution prevents an attacker from quickly recovering the original passwords without the need for a brute-force search, even if they obtain access to the hash values saved in the blockchain.

The suggested solution also has the benefit of enabling the use of relatively weak passwords without sacrificing security. Users are frequently obliged to use cumbersome, lengthy passwords in conventional password-based systems. Users of the proposed system can create passwords that are straightforward and easy to remember since the system's security is not dependent on the complexity of the passwords.

The suggested approach does, however, have some potential drawbacks. The system's reliance on the blockchain's security is one cause for concern. The whole authentication mechanism may become insecure if the blockchain is hacked. Due to the necessity to access the blockchain and carry out cryptographic operations, the system may be slower and more resource-intensive than conventional password-based systems.

Conclusion:

In conclusion, the proposed blockchain-based password-based authentication system provides a solid solution to solve the security issues related to conventional password-based authentication systems. The approach enables the use of relatively weak passwords without compromising security, which has the potential to reduce the risks related to password database breaches. The blockchain-based authentication method ensures the integrity and validity of the system by providing a decentralised and unchangeable record of user credentials.

The suggested approach does have certain drawbacks, though. The security of the underlying blockchain technology, which is susceptible to assaults like 51 percent attacks, is a determinant of the security of the system. However, the system's speed might be hampered by the high processing demands of blockchain transactions, which result in longer transaction times.

However, because the suggested approach necessitates a substantial change to the current infrastructure for password authentication, its adoption may be difficult. The system's total cost may increase due to the potential need for more resources for the upkeep and administration of the blockchain infrastructure.

In spite of these difficulties, the suggested approach offers a fascinating topic for more study and improvement in the domain of password-based authentication. Further research into the system is appealing given its potential advantages, which include improved security, better flexibility, and a decreased reliance on centralised password databases.

In conclusion, the suggested password-based authentication system leveraging blockchain technology provides a potential topic for further research, presenting a creative solution to the issues related to conventional password authentication systems. With more research, the technology might transform how we think about password security and play a significant role in password authentication in the future.