IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 22, Issue 4, Ser. I (Jul. – Aug. 2020), PP 42-51
www.iosrjournals.org

Hybrid Password Authentication System

Harish Manikandan Balaji[1], Abisek K.S[2], Dharmesh Gopinath[3]
[1,2,3]*(Computer Science and Engineering, Anna University, India)*

Abstract:
The storage and handling of passwords is a vital part of security systems. We introduce a novel approach for password authentication to store and handle passwords securely, which could be seamlessly, merged with existing security systems. This approach protects the genuine plaintext password from being discovered by a malicious actor even if the database storing the passwords suffers a data breach. The data breach could be a result of an unpatched or zero-day vulnerability. A plaintext password is hashed using a hash function. The hash is then split into two parts. One part is used to create a series of patterns (or) Anti-passwords involving the use of an Anti-password algorithm, which are then stored securely in a database. The other part is encrypted using AES algorithm. The encryption and the patterns together create two layers of security making it harder to crack through and access the passwords. The analysis and complexity of the system shows that the encrypted passwords can resist lookup attacks and provide even stronger protection against dictionary attacks. It should also be mentioned that the system combines the cryptographic hash function, patterns, and encryption, without requiring anything more than just the password.

***Key Word**: Advanced Encryption Standard (AES); Anti-password; Negative Database (NDB); Hybrid password authentication system (HPAS).*

Date of Submission: 05-07-2020 Date of Acceptance: 21-07-2020

I. Introduction

With the evolution of the internet and its far-reaching capabilities a variety of online services have emerged. In such services passwords are the most used method of authentication. Since passwords are a vital part of accessing various services the security of those passwords is of great importance to the industries and academies. Despite all the effort put into the security of passwords they are cracked due to a variety of reasons. Most passwords are compromised due to user's negligence and others due to the system itself being compromised [13]. In addition, the problems in the systems may cause passwords to be compromised. It is very challenging to get passwords from high security systems. On the one hand, stealing authentication data tables [3] containing usernames and passwords in high security systems is difficult and carrying out guessing attacks involves limited login attempts [15]. Weak points in the system are always being discovered and most of the systems cannot be updated to resist attacks this enables attackers to exploit and to gain access.

After obtaining authentication data tables from vulnerable systems threat actors can execute offline attacks. In the data tables the passwords are usually in the form of hashes. As computing power and storage resources are getting more sophisticated hashed passwords cannot withstand precomputation attacks such as rainbow table attack and lookup table attacks [2] [12]. Large scale password guessing attacks can be used to crack a lot of passwords [11]. Some powerful attack tools [6] like hashcat and John the Ripper(JtR) provide a spread of functions like multiple hash algorithms, attack models, operating systems, and platform. It raises a better demand for secure password Storage.

One of the major reasons for the success of lookup table attack is, the corresponding hashed password is decided for a given plain password. Therefore, the lookup table might be quickly constructed, and the size of the lookup table is sufficiently large. This enables a high success rate of cracking hashed passwords [5]. Normal password protection schemes include password hashing, salted password and key stretching. Among these schemes, hashed password is usually eliminated for its susceptibility towards precomputation attacks [8]. Salted passwords can resist precomputation attacks but it would introduce an extra element and it cannot resist dictionary attacks. The key stretching schemes provide stronger password protection than salted password but configuring parameters requires lot of effort by the programmers.

A password protection system called Hybrid Password authentication system (HPAS) is proposed, it uses a combination of hashing and encryption algorithms along with the proposed anti-password algorithm. The anti-password algorithm converts a password to a hash using hash functions; it is converted to a series of patterns. The strength of the algorithm depends on the confidential key which is the hash of the password and is nearly unique for every user and need not be specially generated. Subsequently the HPAS enables symmetric encryption to be used for password protection. The patterns are saved in a database called negative database (NDB) [1] or Anti-password database.

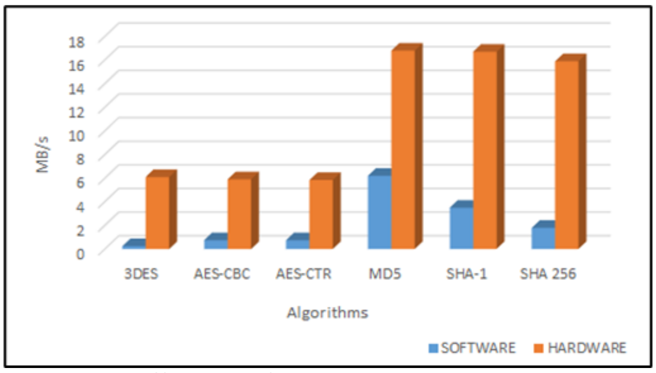

Fig.1 Efficiency of Various Hashing Algorithms

II. Related Works

Fernando Esponda proposed an algorithm for creating and maintaining a negative database [1]. Negative databases should be considered as logical containers of strings. The DB is considered to remain defined over the {0, 1} alphabet. The strings stored in Online Negative Databases [16] implement some partial matching rules, such as, removing or inserting. A single string changes the definition of Database according to the specifics of that match rule. The proposed algorithm protects passwords in an authentication data table [18] [19] [20]. The system designer must first select a cryptographic hash function and a symmetric key algorithm.

Typical Password Protection Plans:

Hashed Password

To securely store passwords a standard method is to hash passwords employing a cryptographic hashing algorithm. It is mathematically impossible to directly recover plain passwords from hashed passwords. The cryptographic hash function quickly converts data of variable size to a sequence of fixed-size bits. Various hash functions available are shown in Table 1 and the output size for each hash function is also mentioned [17]. The hashed passwords are unable to prevent lookup table attack. The rainbow table attack is very effective to crack hashed passwords [11]. The effectiveness of various hashing algorithms is shown in fig. 1.

Password Salting

To resist precomputation attacks the foremost common scheme is salted password. During this scheme, the concatenation of a clear password and a random data called salt is hashed using cryptographic hashing algorithm. The salt is generated randomly. It ensures the hash values of equivalent plain passwords are nearly always different. The greater the dimension of the salt the higher the password security is. Passwords that are salted are still weak against dictionary attacks [4] [5] [6].

Table no 1: Cryptographic Hash Functions

Hash function	Number of bits
MD5	128 bits
BLAKE-256	256 bits
RIPEMD-320	320 bits
SHA-512	512 bits

Anti-password Database

In the database the complement of a positive database is compressed and stored. It is denoted as DB. U = {0, 1} and there are many n-bit values where ‘n’ denotes the universal set of n-bit sequences and x ∈ U indicates an n-bit sequence. DB = {x1, x2,…, xm} indicates a positive database that contains m entries. The Anti-Password DB [9] stores the compression implemented using the wildcard ‘*’ of U − DB [14]. Every entry in the database contains three symbols: ‘0’, ‘1’, and ‘*’. The symbol ‘0’ only match the bit 0, and therefore the symbol ‘1’ only match the bit 1. The symbol ‘*’ can match either the bit 0 or 1 [10]. Every entry in the database consists of two sorts of positions, specified positions, and unspecified positions. The positions consisting of the symbols ‘0’ or ‘1’ are called specified positions. The positions having the symbol ‘*’ are called unspecified positions.

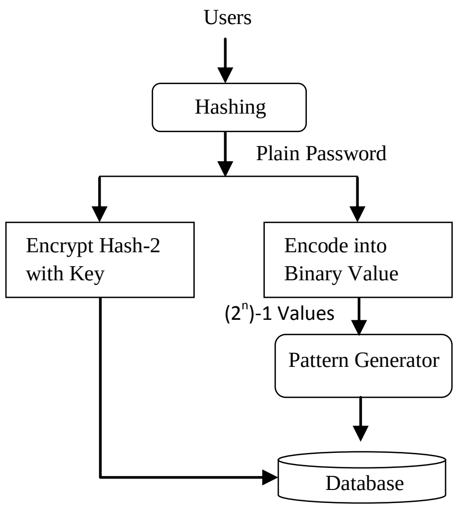

Fig.2 Account Registration Phase

III. Proposed Framework

The proposed framework includes two stages, the registration stage and authentication stage. When adopting our framework to protect passwords in an anti-password database the system designer must first select a cryptographic hash function and a symmetric-key algorithm. The condition to be satisfied is that the key size for the symmetric-key algorithm should be minimum 128 bits. For convenience some matches of cryptographic hash functions are given in Table 1. In addition, hash functions that are not present in Table 1 can be used in the anti-password, this adequately indicates the flexibility of our framework. The proposed HPAS is based on the anti-password algorithm hence, for better understanding; the data flow diagram of the anti-password is shown in fig. 2.

User Registration Phase

The steps involved in the user registration phase are:
Step-1: On the client side, a user enters their username and password. The password is converted into a hash with the selected hashing algorithm and is transmitted to the server through a secure tunnel along with the username.
Step-2: If the received username exists in the database, “The username already exists!” Message is returned, this implies that the server has rejected the registration request, and the registration phase is terminated; otherwise, go to Step-3.
Step-3: The received hashed password is split into 2 equal parts: hash-1 and hash-2. Hash-1 is encrypted using AES algorithm with the key value as hash-2 and is stored in the Anti-password database.
Step-4: The hash-2 part is converted into a binary value ‘b’. It is then converted to anti-passwords using the Anti-password algorithm. This algorithm takes in all combinations of binary values of the length same as ‘b’ as inputs. The values ‘b’ and binary equivalent of 1 and 0 are excluded as inputs.
Step-5: The anti-passwords are a set of patterns. Each pattern has the characters ‘*’, ‘1’, ‘0’ as its values. If required, the admin of the server can use multi-iteration
encryption to enhance the protection of the passwords stored in the server.
Step-6: The username and the resulting anti-password are stored in the Negative database and “Registration success” is returned, implying that the server has accepted the registration request.

Password Hashing

Hash functions are any mathematical function used for converting data of variable size to a fixed size value. A hash functions output is called a hash. Hans Peter Luhn was the first person to have used the concept of hash. The result of the hash function is mathematically irreversible, and the plain text cannot be retrieved from the hash.

Binary Value Generation

The hashed password is split into two parts and the second part is encoded into a binary value. The encoded value's length is 64 characters. In the proposed work, we split hash-2 value consisting of 64 characters into 16 equal parts each of length four. We randomly pick a value from each of the 16 parts and it is dynamically generated for each user. A sample conversion is shown in Table 2.

Table no 2: Binary Encoding Algorithm-Sample

Position	132432
Binary	100110001010010110000101
Split	1001|1000|1010|0101|1000|0101
Final	100001

The total number of binary values to be given as input to the Anti-password algorithm can be established, using the eqn. (1)

$$\lim_{0\to 6}\left(2^{2^{n}}\right) \qquad (1)$$

2^n in eqn. (1) gives the length of the binary value. The value of 'n' depends on the complexity of the algorithm required by the developer; the complexity will increase with the increase in the size of 'n'. The binary generation for 2 variables is shown in Table 3.

Table no 3: Binary Generation for Two Variables

A	B	Fn1	Fn2	Fn3	Fn4	Fn5	Fn6	Fn7	Fn8
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

A	B	Fn9	Fn10	Fn11	Fn12	Fn13	Fn14	Fn15	Fn16
0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

AES Encryption

It is a symmetric cipher used to guard confidential data and used in many technologies globally to encrypt private data. The algorithm specifies variety of transfigurations to be done on the data, saved in an array. The primary step of the cipher is to place the information into an array. After which, the cipher transfigurations are repeated over variety of encryption rounds. First round of algorithm is represented in fig. 3.

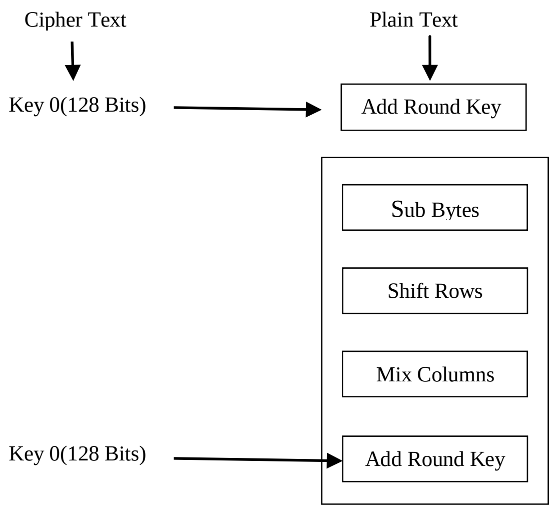

Fig.3AES Encryption [Round 1]

The number of iterations is decided by the key length, with ten iterations for 128-bit keys, twelve for 192-bit keys and fourteen for 256-bit keys. We use hash to encrypt the plaintext passwords. The initial hashed password is split into two equal parts and by using the first part as key, the second part is encrypted. The AES algorithm is a strong and effective algorithm available.

Pattern Generator

The Pattern Generator takes (2n)-1 binary values as its input and produces a set of values or patterns, with each pattern comprising of the characters '*', '1', '0'. In Table 4 a sample pattern generation is shown. The user's hash-2 is binary encoded to 001. All the values in the second column of Table 4 are values excluding "001", every value in it except "011" will match with at least one of the patterns in column 3. The '*' can either be replaced with '0' or '1'.
For instance, 000 will match with *0* and **0, but the value 011 does not match with any of the patterns. The binary value not having any match with the patterns stored for a user is the part of the correct password.

Table no 4: Pattern Generation

Binary Value B	Pattern Generator Input Values	Output Values (Or) Patterns
011	000 001 010 100 101 110 111	*0* 1** **0

Password Authentication

Based on client and server interaction the password authentication phase is classified as five steps.
Step-1: On the client side, a user enters their username and password. Then, the username and hashed password are transferred through a protected channel to the server.
Step-2: If the received username does not exist in the database, "Incorrect username or password" message is returned, which means that the server has rejected the authentication request, and the authentication phase is terminated, otherwise go to Step-3.
Step-3: The hash value received is split into two parts: hash-1 and hash-2. Retrieve the stored anti-passwords in the database shown in Table 4 for the corresponding user. Convert hash-2 value into a binary equivalent and

compare bit values to the anti-passwords. If no pattern is matched with the binary value, one part of the password is authenticated.
Step-4: Retrieve the encrypted portion of the password from the database and decrypt using hash-2 value as the key. If the decrypted value matches with the hash-1 value, the second portion of the password is authenticated.
Step-5: If the hash value of the received password is not the solution of the anti-password, “Incorrect username or password” message is returned. It can be inferred that; the server has denied the authentication request. As a result, the authentication is terminated. Otherwise, “Authentication success” message is returned, it can be understood that the server has approved the request.

Anti-passwords

Anti-passwords are obtained through the steps in fig. 2. Firstly, the received plain text password from the client is converted to hash using a hash function. Next, the hash is transformed into an anti-password using the Anti-password algorithm. Thus, the anti-passwords are obtained. In the above processing, each component, such as cryptographic hash function, symmetric-key algorithm, and the database generation algorithm is indispensable.

The value space of anti-passwords for a hashed password is sufficient to resist precomputation attacks. The database generation is straightforward and efficient. Algorithm 1 elaborates how patterns are compared with the binary value. Based on random and inverse permutations, randomness is introduced to implement reversible one-to-many mappings.

Moreover, multi-iteration encryption could be introduced to enhance anti-passwords strength, which is based on the implementation of key stretching technique. The greater the number of encryptions is the more secure the anti-passwords. However, the authentication speed decreases. The system designer must select a proper encryptions algorithm to balance the speed of authentication.

IV. Analysis of Attack Models

There are several types of attack models commonly used by attackers to compromise password systems [7] and databases. Using these sophisticated attacks, combined with the available technology, attackers can compromise passwords of various services both online and offline effectively.

Attack Models

Upon acquiring an authentication data table, attackers may use brute force attacks, lookup table attacks, dictionary attacks to crack the passwords stored within the table. Depending upon the precomputation technique the attacks can be split up into two different categories. The first category includes rainbow table attack, lookup table attack and the second category comprise of dictionary attack, advanced dictionary attack, reverse lookup table attack and brute force attack.

Algorithm 1: To Check 2 Terms Differ by One Bit
Input:*2 binary values b1, b2 (E.g. 1010, 1000)*
Output:The value 10*0 is returned*
Pattern_match(a, b)

```
1.  Maximum← LENGTH (bin)
2.  for s←zero to Maximum incrementally do
3.      bin ← "0" + bin
4.  end for
5.  return bin
6.  flag ← 0
7.  long ← LENGTH(a)
8.  for s ←zero tolong incrementally do
9.      if a[s] ≠ b[s]
10.             flag ← flag + 1
11.     end if
12. end for
13. return flag ← 1
```

In the first category, due to the precomputation technique, both lookup table attack and rainbow table attack are effective methods to quickly crack many passwords. Therefore, the latter could attempt more possible passwords for its space-time trade-off. In the second category, all four attacks simply attempt every possible password within the password list. Moreover, the difference between those four attacks is the construction of the password list. The password list in the brute force attack is the list of all combinations of characters in a given character set. Without loss of generality, we select lookup table attack from the primary category and dictionary attack from the second category to analyze the capability of our scheme. The details of the two attacks are described in the following section.

Lookup Table Attack

When completing a lookup table attack, an attacker first prepares a password list that mostly constitutes of frequently-used passwords, concatenations of words in a vocabulary list and so on. Then, a lookup table is made, where the keys are encrypted passwords, converted from elements within the password list by an encryption algorithm. Once the lookup table is prepared, the attacker performs the attack. Finally, for each encrypted password within the authentication data table, the adversary searches for the initial plain password by matching the encrypted password and therefore the keys within the lookup table. The adversary could shorten the search time by adopting binary search algorithm or the information structure of hash table.

Dictionary Attack

When executing a dictionary attack, the adversary first steals the authentication data table. Next, they prepare a password list for the attack. Finally, for each encrypted password in the authentication data table, the attacker converts each element within the password list to cipher-text. It is understood that the attacker knows the encryption algorithm utilized in the authentication data table. Then, it determines whether the cipher-text matches with the encrypted password. If any match is established, the adversary immediately obtains the initial plaintext password.

Attack Complexity Analysis of Attack Models

The attack complexity of the encrypted anti passwords under lookup table and dictionary attack is analyzed.

Lookup Table Attack Analysis

There are a lot of corresponding anti-passwords for a given plaintext password. If the attacker intends to crack the proposed algorithm, they must compute all the possible anti-passwords for every element in the password list. Thus, the number of possible anti-passwords is computed. The permutation causes the diversity among the anti-passwords. Hence, the number of anti-passwords that are converted from a plaintext password are identical to the number of permutations. The magnitude of the lookup table increases rapidly with the magnitude of the password hash. Usually magnitude of the hash is usually 128, 256, or 512 bits. The anti-password algorithm can resist lookup table attacks.

Dictionary Attack Analysis

To crack anti-passwords using dictionary attack, a password list is created based upon the values obtained from the authentication table. First a set of passwords in a list are converted to password hashes. The anti-passwords are decrypted where the key is the hashed password, to determine whether the hashed password is the solution of the negative password using certain algorithms and the success indicates that the attackers cracked this anti-password.

Attack Complexity Analysis of Algorithm

To crack anti-passwords using dictionary attack, a password list is created based upon the values obtained from the authentication table. First a set of passwords in a list are converted to password hashes. The anti-passwords are decrypted where the key is the hashed password, to determine whether the hashed password is the solution of the negative password using certain algorithms and the success indicates that the attackers cracked this anti-password.

Attack Complexity Analysis of Algorithm

In order to further highlight the benefits of our scheme, we analyze and compare the attack complexity with the existing algorithms under lookup table attack and dictionary attack.

Hashed Passwords

Hashed password is a widely used scheme to guard passwords in an authentication data table. Hashed passwords are often calculated using eqn. (2).

$$\text{Hash}(P) = \text{HASH}\ (\text{Pplain}) \qquad (2)$$

Where HASH is a cryptographic hash function like SHA-256. Pplain is a plain password, and Hash(P) is the hash value of Pplain. Hashed passwords might be easily cracked by precomputation attacks. For a given plain password the corresponding hashed password is decided. Consequently, attackers could precompute the hash values of all elements within the password list; the hashed passwords constitute the keys of the lookup table and the records are the corresponding elements within the password list. After constructing the lookup table, a linear search algorithm is adopted to lookup hashed passwords within the table, the time complexity of cracking passwords is O (Nd∗ Np ∗Tm_hash). Therefore, hashed passwords are vulnerable under lookup table attack.

Salted Passwords

In order to compensate for the weaknesses of hashed passwords and resist lookup table attacks, salted passwords are used to improve the security. Salt is basically extra elements added to a password to make it more secure and difficult to crack. There is no need to have any specific order and elements can be random. The parameter of the cryptographic hash function is the concatenation of a plaintext password and salt. The size of the salt is usually large and is generated at random. Furthermore, salted passwords vary based on the salt value used.

$$P \rightarrow \text{Plaintext Password}$$
$$S \rightarrow \text{Salt}$$
$$\text{HPS}= \text{HASH}\ (P||S) \qquad (3)$$

In eqn. (3), "||" is a concatenation operator and 'HPs' is the salted password. The salt can be on the left side of the plaintext password. When the dimensions of the salt are sufficiently large the dimensions of the lookup table become too big to precompute and needs plenty of storage resources. The salted passwords can resist lookup table attack but cannot resist dictionary attacks.

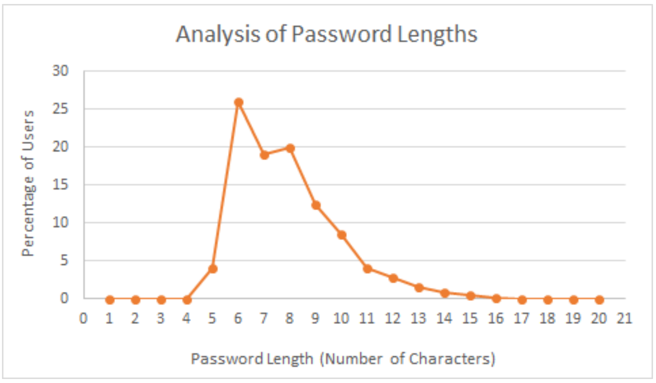

Fig.4 Analysis of Typical Password Length

Anti-Passwords

The proposed Anti-password algorithm can resist a lot of precomputed attacks. When a brute force attack was performed on the AES-128 and the anti-password algorithms, it was found that the attack on the proposed algorithm takes 4 times the time to break the password than on AES-128 algorithm. The comparative analysis is shown in Table 5. We are considering two instances to break the password:

1) Permutations of only 26 characters are allowed to create passwords.

2) Permutation of only 65 characters are allowed. The attack complexities of the two algorithms are represented in fig. 5 and fig. 6. The typical password length is analyzed in fig. 4.

Table no 5: Attack Complexity Analysis of AES-128 and Proposed Algorithm

Length (no. of characters)	I Permitted Characters (26)	II Permitted Characters (65)	AES-128 Algorithm		Proposed Algorithm	
			Time to Break I (Years)	Time to Break II (Years)	Time to Break I (Years)	Time to Break II (Years)
5	11881376	1160290625	3.6125162 X 10-5	0.0027379070	0.00014450065	0.010951628
6	308915776	75418890625	3.6125162 X 10-	0.17796396	0.0039357413	0.71185582

			5			
7	8031810176	4902227890625	0.024641163	11	0.10130256	44
8	208827064576	318644812890625	0.65983559	692	2.6393424	2768
9	5429503678976	20711912837890624	17	42000	68	168,000

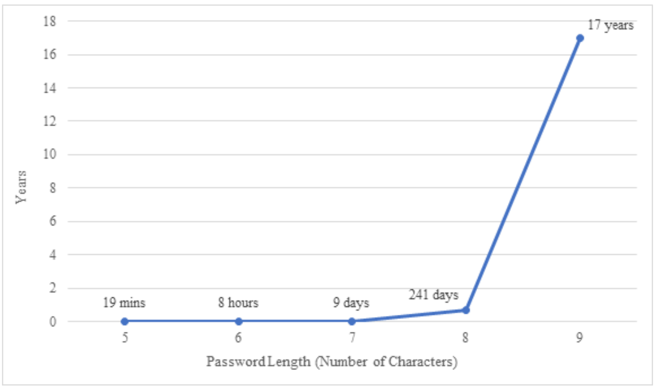

Fig.5 Attack Complexity Analysis of AES 128-bit Algorithm

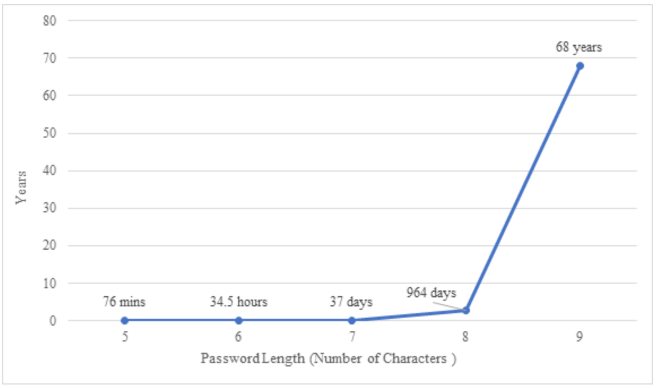

Fig.6 Attack Complexity Analysis of Anti-password Algorithm

V. Advantages

The Encrypted anti-passwords only need to determine a pair of cryptographic hash function and symmetric-key algorithm without the need for extra elements. It is a programmer-friendly algorithm. Even if the database is attacked by attackers the passwords themselves are safe from perusal due to anti-passwords. The complexity of cracking the encrypted anti-password system gradually increases with the increase in size of the binary value. The size of the binary value can be decided by the developer.

VI. Conclusion

Password systems are an essential security feature required in modern technology to protect the user's privacy. Also, it prevents the ever-rising data and related services from being compromised by malicious hackers and system vulnerabilities. The compromise of passwords can lead to the loss of various resources and vital information and in severe cases can impact financial and asset stability.The proposed Hybrid password authentication system (HPAS) is aimed at better securing passwords. After analysis, we have found that the Anti-passwords can protect against lookup table attacks, dictionary attacks and other precomputed attacks. Even under the compromise of the database, the passwords themselves are protected and not compromised. In time, the Anti-passwords will be studied and further improved. Also, the time complexity can be improved by increasing difficulty of cracking systems. Machine Learning techniques can be considered to strengthen the current methodology. It shall prevent malicious actors from compromising a user's password through precomputed attacks and blocks the attacker.

References

[1]. Fernando Esponda, “Everything that is not important: Negative databases”, IEEE Computational Intelligence Magazine, vol. 3, no. 2, pp. 60-63, 2008.

[2]. Emin İslam Tatl, “Cracking More Password Hashes with Patterns”, IEEE Transactions on Information Forensics and Security, vol.10, no. 8, pp. 1656-1665, 2015.

[3]. Hung-Min Sun, Yao-Hsin Chen and Yue-Hsun Lin, “oPass: A User Authentication Protocol Resistant to Password Stealing and Password Reuse Attacks”, IEEE Transactions on Information Forensics and Security, vol. 7, no. 2, pp. 651-663, 2012.

[4]. Liang and J, Lai, X, “Improved Collision Attack on Hash Function MD5”, Journal of Computer Science & Technology vol. 22, pp. 79–87, 2007.

[5]. Chao Gong and Brandon Behar, “Understanding Password Security through Password Cracking”, Journal of Computing Sciences in Colleges, Evansville, United States, vol. 33, no. 5 pp.81-87, 2018.

[6]. Shiva Houshmand, Sudhir Aggarwal and Randy Flood, “Next Gen PCFG Password Cracking”, IEEE Transactions on Information Forensics and Security, vol.10, no. 8, pp. 1776-1791, 2015.

[7]. Ignacio Sanchez, IwenCoisel and Javier Galbally, “A New Multimodal Approach for Password Strength Estimation—Part II: Experimental Evaluation”, IEEE Transactions on Information Forensics and Security, vol.12, no. 12, pp. 2845-2860, 2017

[8]. Hee Jung Lee, Taekyoung Kwon and Young-Ho Park, “Security Analysis and Improvement of the Efficient Password-Based Authentication Protocol”, IEEE…… Communications Letters, vol. 9, no. 1, pp. 93-95, 2005.

[9]. Danezis G et al., “Efficient Negative Databases from Cryptographic Hash Functions”, Information Security Conference, Valparaiso, Chile, vol. 4779, pp. 423–436, 2007.

[10]. Freedman, M.J, Nissim, K and Pinkas, “Efficient Private Matching and Set Intersection”, Advances in Cryptology, Eurocrypt, Springer, Berlin., vol. 3027, pp. 1–19, 2004.

[11]. Mansour Alsaleh, Mohammad Mannan and Paul C. van Oorschot, “Revisiting Defences against Large-Scale Online Password Guessing Attacks”, IEEE Transactions on Dependable and Secure Computing, vol. 9, no. 1, pp. 128-141, 2012.

[12]. Ari Juels and Thomas Ristenpart, “Honey Encryption: Encryption beyond the Brute-Force Barrier”, IEEE Security & Privacy, vol. 12, no. 4, pp. 59-62, 2014.

[13]. K. M. Renuka, SaruKumari, Dongning Zhao and Li, “Design of a Secure Password-Based Authentication Scheme for M2M Networks in IoT Enabled Cyber-Physical Systems”, IEEE Access, vol. 7, pp. 51014 – 51027, 2019.

[14]. KanishkaBajpayee, Surya Kant, Bhaskar Pant, Ankur Chaudhary and Sharma S.K, “Mining Frequent Itemset Using Quine–McCluskey Algorithm”, Advances in Intelligent Systems and Computing, Springer, vol. 437, pp. 763-769, 2016.

[15]. J. Nam et al., “An Off-Line Dictionary Attack on a Simple Three-Party Key Exchange Protocol”, IEEE Communications Letters, vol.13, no. 3, pp. 205-207, 2009.

[16]. Esponda, F., Trias, E., Ackley, E and Forrest, S, “A Relational Algebra for Negative Databases”, International Journal of Information Security, University of New Mexico, vol.8, pp. 331-345, 2007.

[17]. D. R. Ignatius Moses Setiadi et al., “A Comparative Study MD5 and SHA1 Algorithms to Encrypt REST API Authentication on Mobile-based Application”, International Conference on Information and Communications Technology (ICOIACT), Yogyakarta, Indonesia, pp. 206-211, 2019.

[18]. Esponda F, Forrest S and Helman P, “Protecting Data Privacy Through Hard-To-Reverse Negative Databases”, International Journal of Information Security, vol.6, no.6, pp. 403-415, 2007.

[19]. Abdulmotaleb El Saddik, Fawaz A. and Alsulaiman, “Three-Dimensional Password for More Secure Authentication”, IEEE Transactions on Instrumentation and Measurement, vol. 57, no. 9, pp. 1929-1938, 2008.

[20]. Hugo Krawczyk, MalihehShirvanian, NiteshSaxena, and Stanislaw Jarecki, “Building and Studying a Password Store that Perfectly Hides Passwords from Itself”, IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 5, pp. 770-782, 2019.

Harish Manikandan Balaji, et. al. “Hybrid Password Authentication System.” *IOSR Journal of Computer Engineering (IOSR-JCE)*, 22(4), 2020, pp. 42-51.