**Shamir’s Secret Sharing for Authentication without Reconstructing Password**

***Sai Trinadh kumar Pulivarthi – Vamsi Krishna Muppala – Vandana Priya Muppala – Lavanya Thota – Mahitha Pammi – Kavya Gaddipati – Dheeraj Chowdary Yalamanchili – Soumya Shetty***

For the term project we have reviewed the paper ‘Shamir’s Secret Sharing for Authentication without Reconstructing Password**’** and partly implemented it using python.

**Abstract:** Shamir's Secret Sharing is a way to secure a secret in a distributed way and is quantum attack resistant. Shamir’s Secret Sharing is widely used to secure encryption keys and for secret sharing. However, in authentication protocols hashing is used and Shamir’s Secret Sharing can be used in the place of hashing to improve the already existing authentication protocols. From the paper we reviewed we present an authentication protocol which will use Shamir’s secret sharing method to authenticate a secret and it can be used to store in the server. In the post quantum era hashing is not less effective at hiding the data. So, in post quantum era, if any data from the server gets exposed, user's data can also be compromised as they were hidden by using hashing as a one-way encryption. The protocol (i.e., Shamir’s Secret Sharing) will be able to solve this problem in a way that complete data exposure from server will also not reveal the actual data of the user. So, even if the user uses same password for other online services/systems, these services and systems will not be affected.

**Introduction**

Many people use simple and common passwords like ‘123456’ or ‘abcdef’. These are easy to guess and hack ,so many software engineers are encouraging to dive the client to create passwords which are as complex and lengthy as possible .This leads to another significant increase in perilous issue like consummately security[1] .We have considered an example ,fig.1[7], client uses similar passwords for two totally disconnected system A and system B. If the attacker breakers in system A and has control over the framework of system A,which implies he peruse any information or messages. It wont take long time for the attacker to act/minic like the client and get access over system B as well using similar secret key of system A.In this paper we are proposing a framework that prevents the adversary from knowing any helpful data from System A to enter framework B, even it has similar pass keys.

Diagram

Description automatically generated[18]

Next part of the paper consists of the currently used password protocols .After that it consists of the background topics to understand this paper.Next we provide our methology and finally the implementation results and future work and conclusion.

**Related Works**

This provides a general conception of current state of password protocols. Different approaches are being used by security experts, such as using framework explicit "salt" to make pass keys unique to each framework or client explicit salt attributes to make each hidden sentence exceptional. A salt is a random data that uses an additional input to a one - way function, that hashes data, a password or passphrase. This methodology depends on strength of hashing strength[2]. The major advantage of using salts is they are used to safeguard passwords during storage.

Biometric verification are becoming more popular these days, although they are inconvenient to use and unrestricted to carry out. Because customers are becoming more security conscious, mainstream lodgings must avoid biometric detection. Even the usage of a similar secret word and executions for all use scenarios also plagues two-factor or multidimensional verification's. The industry standard "Kerberos" is currently the most widely used authentication convention, although it is also reliant on encryption, which is not quantum attack proof.

Only secret phrase-based confirmation remains the most simple and advantageous option for both the user and the convenience provider. Our method adds enhanced security against Cryptocurrency attacks[3],[4] and advance malware[5],[6] by providing a strong authentication scheme.

**Background**

1. **Password Based Authentication:** The secret sharing methodology mainly uses the concept of Password-based Authentication.Password-Based Authentication[9] is an authentication technique where a user has permission to the desired resource with the help of user credentials that are stored on the system. The passwords created by the users are securely stored in the form of a hash digest value on the system and the access is granted only when the authenticity is checked with the help of a hash digest value. When a user provides the password while using the system, the password is hashed and is compared with the already stored hash digest value. To perform hashing on the given passwords a password reset method is used, which uses a random, non-secret salt value with the hash digest value. Recalculating the authentic password from the already stored hash digest value is not possible.
2. **Hashing:** Hashing converts plain text into cipher text. cryptographic hash function is an algorithm that maps data of random size to a fixed size. Hashing is a one way function and it can’t be reversed hence if the attacker access the database though he can see the hashes he can’t reverse the hash value back to the actual password. Hash functions are used for security protocols design. SHA-1. MD5 is now broken. Argon2, PBKDF2, Bcrypt, and scrypt being considered safer for hashing. Different salt is used in hash function to avoid brute force attack. salt is made up of random bits added to each password before hashing.

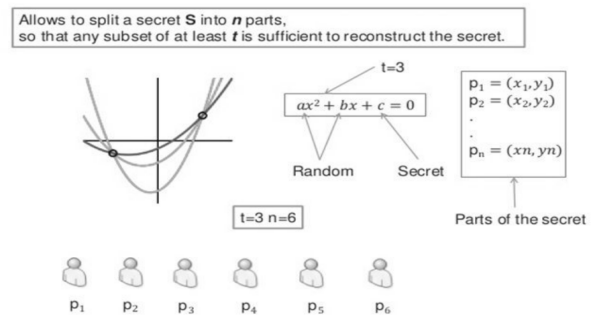
Hash collision occurs when hash algorithm produces same hash value for two different inputs. They are vulnerable to advanced attacks like dictionary attack, hybrid password attack etc. there is always a probability that two distinct inputs generate the identical hash value.

If N is all possible hash value and K is all generated value from inputs, than probability of two input , having same hash value is

Prob (hash collision) = 1 –

Prob (hash collision)(simplified) =

**Shamir’s Secret Sharing :**It is one of the precise and classical cryptography calculations for sending a secret over several parts. It states that "split a secret S into n parts such that with any k-out-of-n pieces you can reconstruct the original secret S, but with any k-1 pieces, no information is exposed about S. This algorithm is a quantum-proof algorithm, which suggests even in a quantum computer cannot retrieve the secret if it has less than the threshold numbers of the key. The only way to retrieve the secret is to combine the shares distributed. The control of the secret is distributed, and this algorithm is an example of Threshold Crypto systems in which several parties must cooperate to reconstruct the secret. The algorithm is based on the concept of polynomial interpolation which states that a polynomial of degree t-1 can be reconstructed from t or more points which are lying on the curve.

[18]

**Our Methodology**

Shamir’s algorithm functions as same as all hash algorithms as one-way function. In which password will be converted into its numerical values with a threshold T using random numbers and the converted values will be selected in such a way that the selected values will be less than the converted values (i.e., T-1 values). Same key will be generated for both registration and sign in time These values are combined to form a new key PK which will be stored in the server. Even though the server exposes the key PK cannot form the original text.

**Registration Algorithm:**

**Step1:** Pw ← plaintext Password, Uid ← User-id, Ss ← Server Salt

**Step2:** Create 512-bit hash TH from Pw + Uid + Ss,

**Step3:** Split TH into T(H1) and T(H2)

**Step4:** Generate values x1, x2, x3....xn from T(H1) by taking the ASCII values from T(H1)

**Step5:** We need to pass the generated values into Shamir’s secret key algorithm S ← T(H1)

**Step6:** Using Xi as X-coordinates to generate Shamir’s secret keys k1, k2, k3....kn, if threshold number is T, pick k1, k2, k3....k(T − 1) keys and send them to server.

**Step7:** Server will store the keys as single string Pss = k1 + k2 + k3....k(T − 1)

**Step8:** server will store the Uid and Pss

Key Collision probability:

prob.png

Pc: Probability of one collision

C : Number of keys

M : Number of user in the system

K : Length of Keys

N : Total representation number(eg: all possible symbol in password)

**Implementation**

We have used, username JOHN, password ABCDEF, and salt S1. We merged it and got a single string "JOHNABCDEFS1“.

1. First, we generated some numerical values from "JOHNABCDEFS1" by simple converting each of character to their corresponding ASCII values.

2. Then we send the ASCII numerical values to Shamir’s secret function, generate\_shares, which will generate the shares for the data.

3. Now when we login with the same userID and password, the shares are compared and matched, we get, login successful.

**ADVANTAGES, LIMITATION AND FUTURE WORK**

**Advantages:**

1. Sharmir’s secret algorithm is adaptable and extensible, which means the secret owner can add, edit, or remove shares at any moment without affecting the original secret.

2. It is unaffected by quantum attacks.

3. Despite the fact that the system server is vulnerable, the password is unrecoverable.

**Disadvantages**:

1. The collision rate is larger than that of hashing.

2. One of the issues with Shamir's secret algorithm is ensuring that stockholders are not submitting false shares.

**Future Work:**

We will use threshold cryptography to improve these methods in the future. This method's collision rate is higher than that of hashing algorithms, which is its limitation. However, we can get around this problem by extending the key length. We can also fix the problem by introducing a new approach for enhancing more uniqueness. In the future, we'll work on it.

**Conclusion**

It is inconvenient for consumers to have separate passwords for each online service. Passwords are commonly reused across many services by users. Any intrusion of a service server puts the user's other online services at risk. As we can see, hash functions are useless against quantum computers. Our proposed methods can tackle this problem by using the strength of Shamir's secret sharing approach.

**Question asked while Presentation:**

Where is Shamir’s secret sharing algorithm used?

1. It is used currently in banking
2. services for sharing the secret information.

What are the limitations in the Shamir’s secret sharing algorithm?

1. Collision rate is higher than the hashing methods.

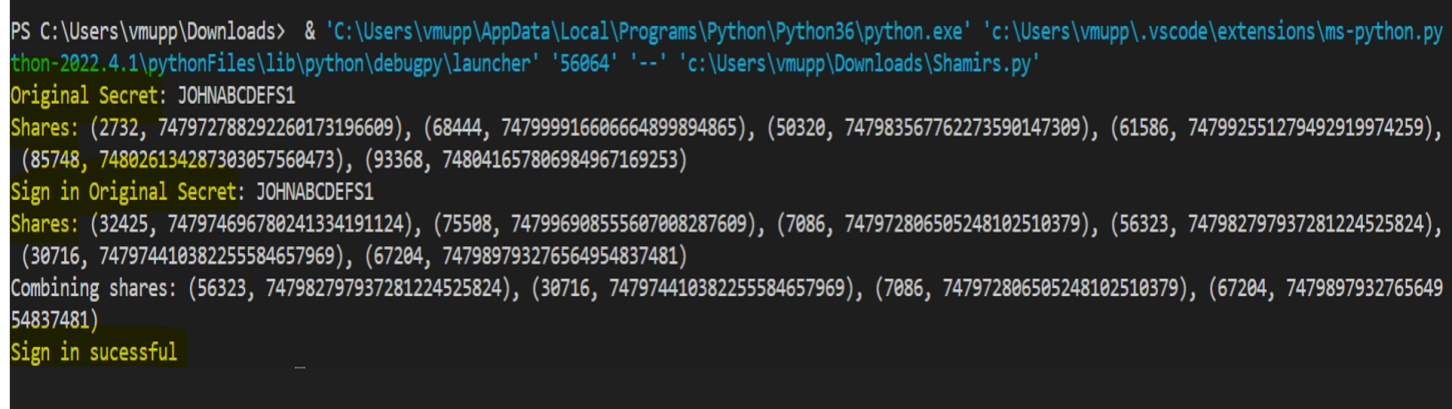
2. Each share of the secret must be at least as large as the secret itself.

3. To ensure that shareholders are not submitting fake shares is one of the problems in Shamir secret algorithm

**Code:**



**Results:**



From the above figure ,we could see the login is successful, when the same userID and password is used.

**References:**

[1] H. Almarabeh and A. Sulieman, “The impact of cyber threats on

social networking sites.” International Journal of Advanced Research

in Computer Science, vol. 10, no. 2, 2019.

[2] K. Aditya, S. Grzonkowski, and N.-A. Le-Khac, “Riskwriter: Predicting

cyber risk of an enterprise,” in International Conference on Information

Systems Security. Springer, 2018, pp. 88–106.

[3] K. D. Gupta, A. Rahman, M. N. Huda, M. A. Parvez, and S. Poudyal,

“A hybrid pow-pos implementation against 51% attack in cryptocurrency

system,” in 11th IEEE International Conference on Cloud Computing

Technology and Science” CloudCom, 2019.

[4] D. Dasgupta, J. M. Shrein, and K. D. Gupta, “A survey of blockchain

from security perspective,” Journal of Banking and Financial Technol

ogy, vol. 3, no. 1, pp. 1–17, 2019.

[5] S. Poudyal, D. Dasgupta, Z. Akhtar, and K. Gupta, “A multi-level

ransomware detection framework using natural language processing and

machine learning,” in 14th International Conference on Malicious and

Unwanted Software” MALCON, 2019.

[6] S. Poudyal, K. P. Subedi, and D. Dasgupta, “A framework for analyzing

ransomware using machine learning,” in 2018 IEEE Symposium Series

on Computational Intelligence (SSCI). IEEE, 2018, pp. 1692–1699.

[7] B. E. Kane, “A silicon-based nuclear spin quantum computer,” nature,

vol. 393, no. 6681, p. 133, 1998.

[8] P. Rai and P. Singh, “An overview of different database security

approaches for distributed environment,” IJISET-International Journal

of Innovative Science, Engineering & Technology, vol. 2, no. 6.

[9] L. O’Gorman, “Comparing passwords, tokens, and biometrics for user

authentication,” Proceedings of the IEEE, vol. 91, no. 12, pp. 2021–

2040, 2003.

[10] A. Shamir, “How to share a secret,” Communications of the ACM,

vol. 22, no. 11, pp. 612–613, 1979.

[11] M. L. Das, A. Saxena, and V. P. Gulati, “A dynamic id-based remote user

authentication scheme,” IEEE Transactions on Consumer Electronics,

vol. 50, no. 2, pp. 629–631, 2004.

[12] D. Mirante and J. Cappos, “Understanding password database compro

mises,” Dept. of Computer Science and Engineering Polytechnic Inst. of

NYU, Tech. Rep. TR-CSE-2013-02, 2013.

[13] L. Lamport, “Password authentication with insecure communication,”

Communications of the ACM, vol. 24, no. 11, pp. 770–772, 1981.

[14] M. Dietzfelbinger, M. Mitzenmacher, R. Pagh, D. P. Woodruff, and

M. Aumüller, “Theory and applications of hashing,” 2018.

[15] D. R. Morris, P. D. Kotharl, and R. A. S. Florin, “Method and system

for facilitating printed page authentication, unique code generation and

content integrity verifification of documents,” Apr. 17 2008, uS Patent

App. 11/582,101.

[16] L.-J. Pang and Y.-M. Wang, “A new (t, n) multi-secret sharing scheme

based on shamir’s secret sharing,” Applied Mathematics and Computa

tion, vol. 167, no. 2, pp. 840–848, 2005.

[17] Y. G. Desmedt, “Threshold cryptography,” European Transactions on

Telecommunications, vol. 5, no. 4, pp. 449–458, 1994.

[18] K. D. Gupta, M. L. Rahman, D. Dasgupta and S. Poudyal, "Shamir's Secret Sharing for Authentication without Reconstructing Password," 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), 2020, pp. 0958-0963, doi: 10.1109/CCWC47524.2020.9031270.

**Team Members:**

Sai Trinadh Kumar Pulivarthi (R11800471)Vamsi Krishna Muppala (R11770853)Vandana Priya Muppala (R11751885)Mahitha Pammi (R11800440)Lavanya Thota (R11801643)Kavya Gaddipati (R11800877)Dheeraj Chowdary Yalamanchili (R11799764)Sowmya Shetty (R11770733)