Shamir's Secret Sharing for Authentication without Reconstructing Password

Kishor Datta Gupta
Department of CS
The University of Memphis
Memphis, TN, USA
kgupta1@memphis.edu

Md Lutfar Rahman
Department of CS
The University of Memphis
Memphis, TN, USA
mrahman9@memphis.edu

Dipankar Dasgupta
Department of CS
The University of Memphis
Memphis, TN, USA
ddgupta@memphis.edu

Subash Poudyal
Department of CS
The University of Memphis
Memphis, TN, USA
spoudyal@memphis.edu

***Abstract*—Shamir's Secret sharing is a quantum attack proof algorithm and is used heavily for secret sharing. But it can also be used for authentication protocols as a replacement of hashing. In this paper, we propose an authentication protocol which will use Shamir's secret sharing method to authenticate with server. Hashing may not be able to hide data as effective in post quantum era. So in post quantum era, if any data server get exposed, users credentials can be also compromised as they were hidden by using hashing as an one way encryption. Our protocol will be able to solve this problem in a way that complete data exposure from server will not reveal the actual password provided by the user. So, even if the user uses same password for other online services/systems, these services and systems will not be effected.**

***Index Terms*—Shamir's secret sharing, Security, Threshold cryptography, Authentication, Password**

I. Introduction

Numerous individuals utilize powerless or customary passwords. Conspicuously trifling passwords like '123456' or 'abcdef' are the most widely used passwords. Software engineers are always endeavoring to drive the client to settle on the complex and lengthy pass key, making it harder to recall sundry passwords. So clients are inclined to utilize just complex and lengthy pass key for all the digital services. Tragically, that offers ascent to a significantly increasingly perilous issues like, regardless of whether a framework is consummately secure; the adversary can get access as a substantial client if another framework got traded off [1].

As example, in figure 1 we can see, clients utilize similar passwords so as to get to two totally detached systems A and B. Assume an enemy has assumed break in System A. Presently having control of any framework implies he can peruse any information or any system messages in System A. In post-quantum era we may accept that it won't take such a long time for him to mimic a client and access framework B with a similar secret key as in System A. In this paper, we are proposing a framework so that it will keep the adversary from knowing any helpful data from System A to enter framework B, which has similar pass keys.

In this paper, the next section provides a general conception of the current state of password protocols. The third section provided some background on related topics need to understand this paper. In the following section, we provide our methodology. After that, in the sixth section, we will analyze the efficacy of our proposed protocol with the conclusion section, which will summarize the paper and recommend some future works.

TABLE I
CONTRAST WITH "KERBEROS" SYSTEM

Kerberos protocol	**Our Method**
Single point of failure	Not exist
Vulnerable to quantum attack	Quantum safe
If hacked can give away meaningful credentials	Not possible
requires user accounts, clients services on the server in the same Kerberos domain	No such requirement
Can not support third party authentication	can support

II. Related works

In this section, we will discuss how other methods are trying to resolve the issue we discussed in the previous section.

Security specialists are utilizing various methodologies, for instance, utilizing framework explicit "salt" to make pass keys one of a kind to every framework or client explicit salt qualities to make every secret phrase extraordinary. This methodology depends on strength of hashing strength [2]. These days, biometric verifications are engendering however it has an absence of convenience, and it is unrestrained to execute. As clients are winding up increasingly more protection mindful, mainstream lodging need to avoid biometric apperception. Two factor or multifaceted verifications also endure the issues of using a similar secret phrase and executions for all use cases. Just secret phrase predicated confirmation remains the most effortless and advantageous methodology for both utilizer and convenience supplier. Our method adds enhanced security against Cryptocurrency attacks [3], [4] and advance malware [5], [6] by providing a strong authentication scheme.

Industry standard "Kerberos" is presently most broadly utilized convention for authentication yet at the same time it has dependency on encryption which are not quantum assault evidence. In table 1, we demonstrated contrast of our convention with "kerberos" system.

If we contrast with different strategies (other secret sharing techniques for validation), our principle contrasts are we

978-1-7281-3783-4/20/$31.00 ©2020 IEEE

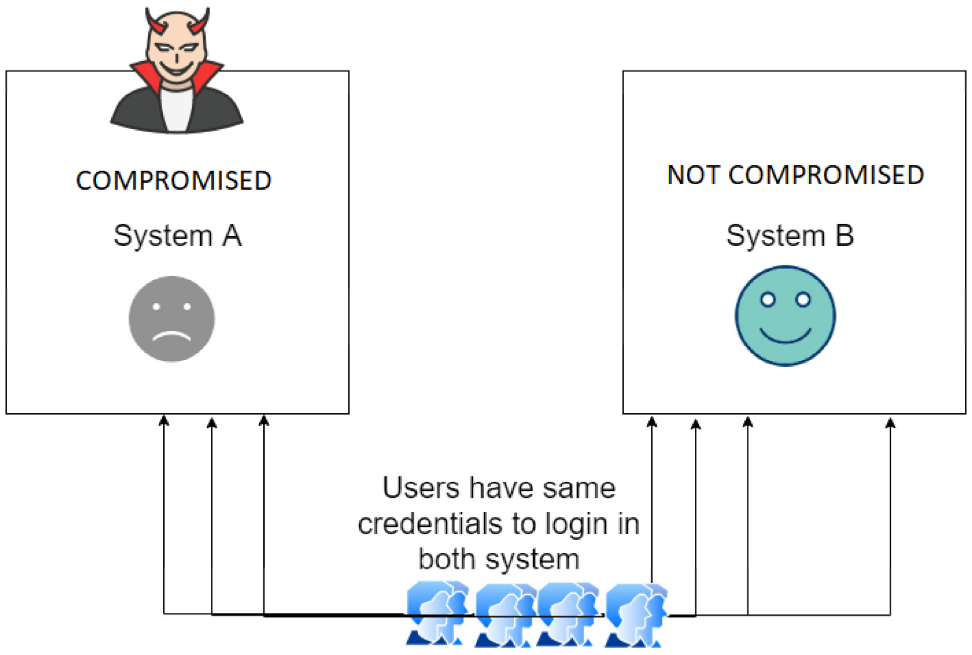

Fig. 1. Fall Back from Users using same passwords for multiple systems

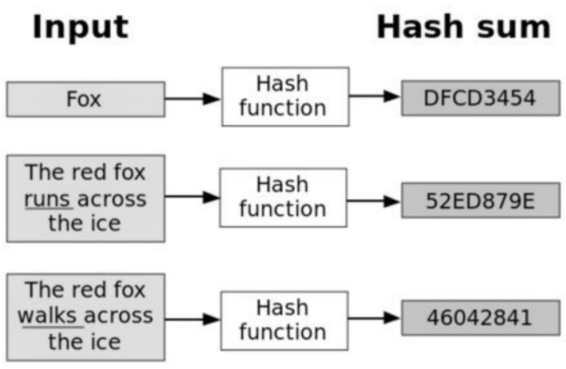

Fig. 2. Simple Hashing

don't have to recreate the secret phrase and we are utilizing Shamir's secret sharing key, which is quantum-safe. We see that most advanced hashing algorithms are helpless against future quantum PC age [7]. Another significant point we have is , we don't have any single point of failure as we don't have any data in server which can be valuable to an aggressor [8].

III. Background

Our proposed methodology is using Shamir's secret sharing for preventing password exposure. In this section, we will discuss the basic concept behind our methodology.

A. *Password Based Authentication*

For user authentication, there are numerous methodologies, yet generally beneath 3 class of strategies are utilized [9] they are : Something the user knows , Something the user has and Something the user is.

Password-based authentication (PAP) falls into the top of the line, and it is the most direct, most open and a financially savvy route for user validation. It is all inclusive, and there is no framework where it isn't appropriate [11]. Password-based authentication is most convenient due to bio metric authentications are not feasible in many cases. And two-factor authentications are time-consuming. The password stored at the server in a hash format, so in case of server exposure, the user password did not get revealed. In a simple scenario, the user saved a key text in the server (server which user required service manages) at registration time. Following that, when the user requires to use the service, he will supply key text. The server will match it with the collected key text if it resembles, then the user is authentic [12].

To provide integrity, password check mundanely depends on cryptographic hashes. Putting away all client passwords as cleartext can establish a security break if the password is undermined. One approach to diminish this jeopardy is to just store the hash digest of every secret word. To verify a utilizer, the secret phrase exhibited by the client is hashed and contrasted and the stored hash. A secret phrase reset technique is required when secret key hashing is performed; unique passwords can't be recalculated from the stored hash value [13]. An intruder may target a client system or database computer. But the server database usually holds the table of validation. It is, hence, invariably more productive to target servers than to attack client workstations. He can obtain only one user's passwords as an attacking user device. If he succeeds in exploiting the server, he can get all of the system's

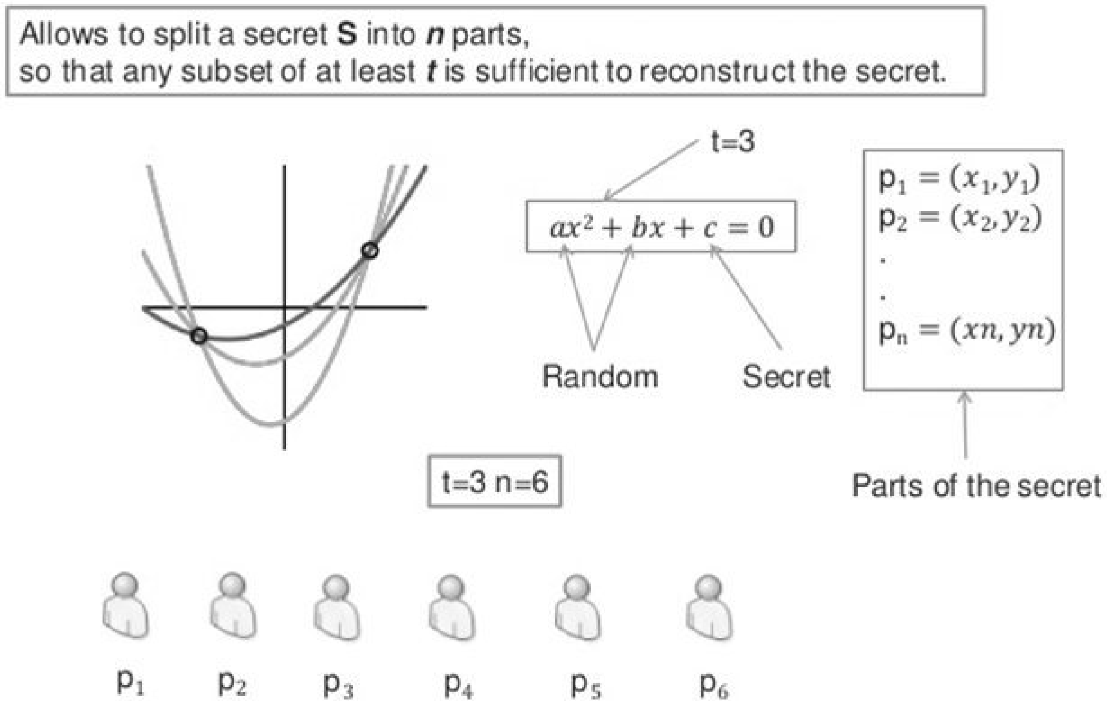

Fig. 3. Shamir's secret sharing [10]

user credentials. Servers are, therefore, more fruitful. [14].

B. Hashing

Hash-functions are functions that can map any list of values to a smaller list of fixed size Keys. A collision occurs when two values allocated to the same keys. Cryptography hash functions are some secure algorithm functions used for security protocols design. SHA-1, MD5 is now broken, with Argon2, PBKDF2, Bcrypt, and scrypt being considered safer for hashing. The hash functions now have different salt to avoid a brute-force attack, and converting a code to hash is easier. They are vulnerable, however, to advanced attacks like Dictionary Attack, Reverse lookup tables, Birthday attack, Hybrid password attack, etc.

When two key-value has the same hash digest after hashing conversion, that known as hash collision, a hash function in the set of inputs has no perception of "other" items. Only some arithmetic AND/OR bit-magic manipulations are executed on the input value passed to it. Hence, 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

$$Prob_(hashcollision) = 1 - e^{\frac{-k(k-1)}{2N}}. \quad (1)$$

if we simplified it ,

$$Prob_(hashcollision)_(simplified) = \frac{k^2}{2N}. \quad (2)$$

C. Shamir's Secret Sharing

It is one of the precise and classical cryptography calculations (1979) by cryptographer Adi Shamir for sending a secret over several parties [10]. It expresses that . 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. That is conventionally called an (n, k) threshold scheme. [10]".It does not divide a secret into parts that are shorter than the first secret. Rather, it breaks the secret into parts that do the equivalent extent as the initial secret. This algorithm is a quantum-proof algorithm, which suggests even in a quantum computer can not retrieve the secret if it has less than the threshold numbers of the key. Figure 3 shows the basic overview of Shamir's algorithm. As it is using random part everytime it will create new keys.

IV. Our methods

In this section, we will formulate the problem, describe the methodology with an example, and provide the protocols for registration and sign-in time.

A. Problem formulation

If a user with user id U_{id} and password P for a access a system . We will have to design a function $F = f(P)$ that convert the P to a P_u and store that in the systems server S. The F will work in a way that P_u can not be converted back to P even if there is a quantum supremacy and will also have below properties

1) Different P will not have the same P_u.

2) For a user registration and sign in time valid P will create same P_u.
3) From the P_u, P reconstruction will not be possible.

B. Our Methodology

Our methodology is very straight forward. We will make shamir's secret sharing algorithm to function as a one way function like hash algorithm. For authentication purpose we need to generate same key for registration time and each of the sign in times. As original shamir's algorithm uses a random part we need to provide this random part, always same for every user. We will generate numerical number deriving from user provided password. As example any ASCII conversion will give us numerical values. That way we will always get same keys. In next part, we will select less than the threshold number of keys than our Shamir's algorithm generated keys. We will store the merged part of these keys in server. In case of server exposure, due to having less than the threshold number of keys the password wont be able to recover as per as Shamir's algorithm mathematical proof is Information-theoretic security which means it is perfect secure unless someone knows the random part and in our customize version the random part is coming from user actual password, so guessing these number will be brute force methods.

In short, We will convert user-provided password (with server salt and user id for uniqueness [15]) to some numerical values. Then we will send the password, and the numerical values to our customize Shamir's secret sharing algorithm to generate N number of keys with the T threshold were (T<N). We will pick the first T-1 number of keys from the total N keys. We will merge them as a single string and consider it is as our new password.

Algorithm 1 Registration Protocol

```
1 $P_w \leftarrow plaintextPassword$
$U_{id} \leftarrow Userid$
$S_s \leftarrow ServerSalt$
2 Create $512$ bit hash $T_H$ from $P_w + U_{id} + S_s$,
3 Split $T_H$ into $T_(H_1)$ and $T_(H_2)$
4 Generate values $x_1, x_2, x_3....x_n$ from $T_(H_1)$ by taking the ANSI values from $T_(H_1)$
5 Let take shamir's secret $S \leftarrow T_(H_1)$
6 Using $x_i$ as x-coordinates to generate Shamir's secret keys $k_1, k_2, k_3....k_n$, if threshold number is $T$, Pick $k_1, k_2, k_3....k_(T-1)$ keys and send them to server.
7 server will store the keys as single string $P_{ss} = k_1 + k_2 + k_3....k_(T-1)$
8 server will store the $U_{id}$ and $P_{ss}$
```

C. Implementation example

In figure 5 ,We got username JOHN, password ABCDEF, and salt S1. We merged it and got a single string "JOHNABCDEFS1", first we generated some numerical values from "JOHNABCDEFS1" by simple converting each of character to their corresponding ASCII values. Then we send the string "JOHNABCDEFS1" and the numerical values to Shamir's secret key generator, which will generate six keys with four thresholds. We will pick the first 4-1=3 keys and merged them and send them to the server. Now in case of server exposure, as these keys only have three parts, it will not be able to reconvert to the password as that requires at least 4part of keys. But it is unique enough as a password.

Algorithm 2 Signin Protocol

```
1 $P_w \leftarrow plaintextPassword$
$U_{id} \leftarrow Userid$
$S_s \leftarrow ServerSalt$
2 Create $512$ bit hash $T_H$ from $P_w + U_{id} + S_s$,
3 Split $T_H$ into $T_(H_1)$ and $T_(H_2)$
4 Generate values $x_1, x_2, x_3....x_n$ from $T_(H_1)$ by taking the ANSI values from $T_(H_1)$
5 Let take shamir's secret $S \leftarrow T_(H_1)$
6 Using $x_i$ as x-coordinates to generate Shamir's secret keys $k_1, k_2, k_3....k_n$, if threshold number is $T$, Pick $k_1, k_2, k_3....k_(T-1)$ keys and send them to server.
7 server will generate the keys as single string $P_{ss} = k_1 + k_2 + k_3....k_(T-1)$
8 server will check $P_{ss}$ and $U_{id}$ with the stored record, if these matched user will grant access or not.
```

D. Key Collision probability

If, Total representation number(eg: all possible symbol in password): C Number of keys: K Length of Keys: M Number of user in the system: N

Probability of one collision:

$$P_c = 1 - \left(\frac{(C-1)}{C}^{M \times k^2} \times \frac{N \times (N-1)}{2}\right) \tag{3}$$

If we increase the number or length of keys probability of collision will get decrease [16]. To achieve SHA-256 level of collision probability we need large set of keys. But with combination of representation number and adding uniqueness to password from server side can reduce the probability of collision

E. Simulation result

We simulated our result with 10000000 user id and password, and we weren't able to generate a single collision. In our simulation, we merged user id, password, and random value from the server (which server stored against the user in Database). After the merge, we generated a 512 hash key using SHA-512. We send these keys to Shamir's secret algorithm, and we made 350keys with 300 thresholds. We merged 290keys as a single password. And 10000000 generated password no collision happened, simulation example will be published in Github.

V. Advantages, Limitation and future work

This year, Microsoft and Google announced that they could develop a quantum computer. So the future of hashing algorithms is at risk. A significant advantage of this methodology is

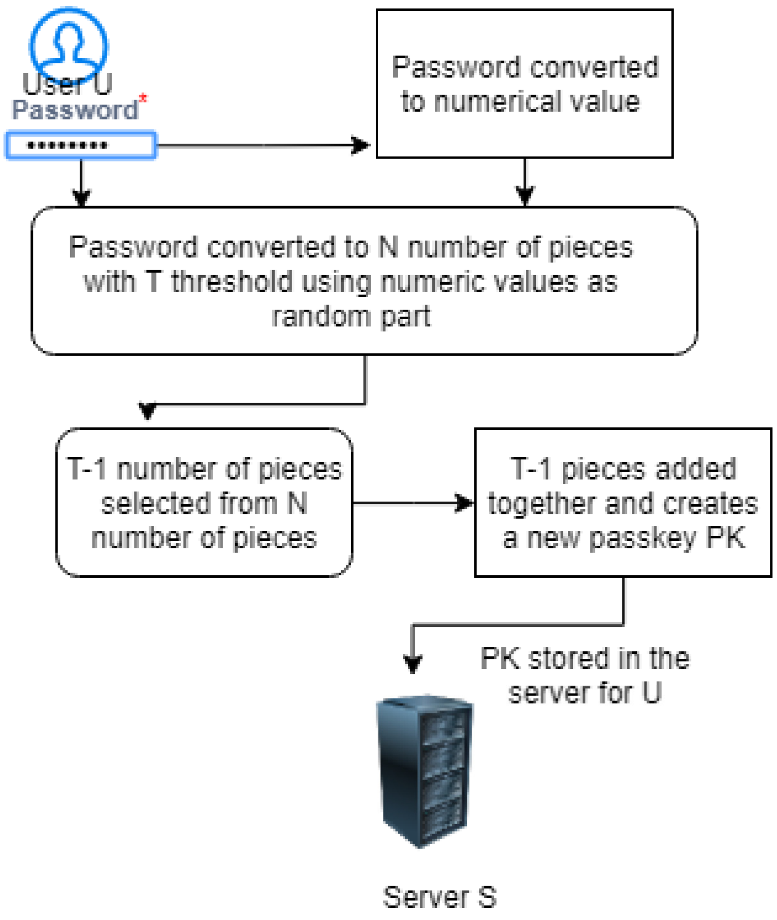
User U
Password*
••••••••
Password converted to numerical value
Password converted to N number of pieces with T threshold using numeric values as random part
T-1 number of pieces selected from N number of pieces
T-1 pieces added together and creates a new passkey PK
PK stored in the server for U
Server S

Fig. 4. Basic Registration flow [10]

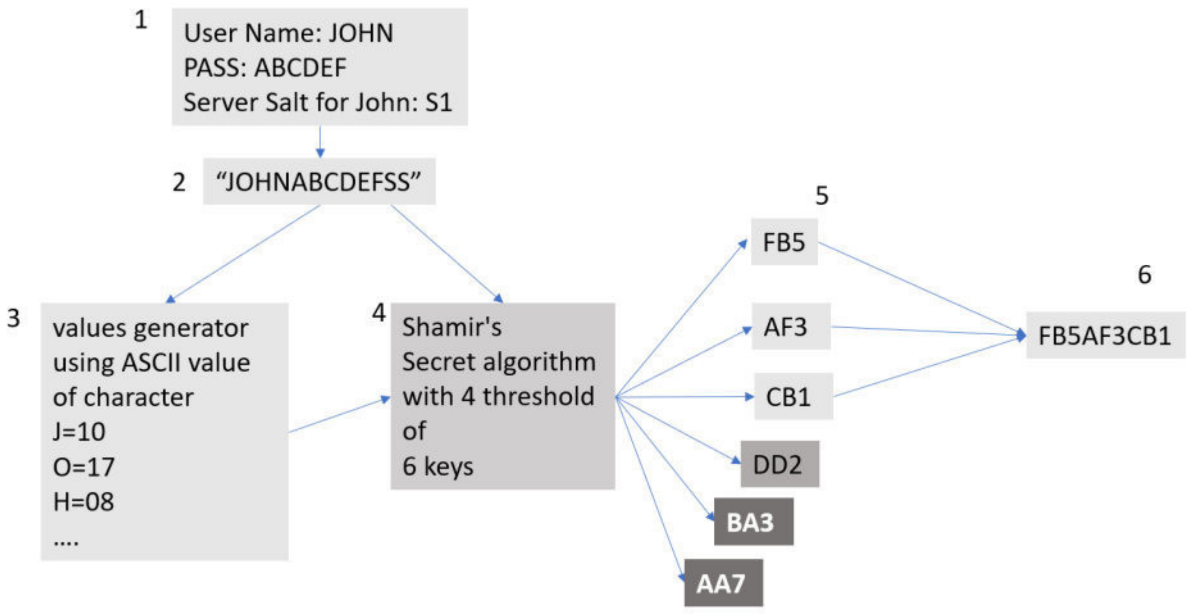

Fig. 5. A simplest example for implementation

it is not vulnerable to quantum attack as it is based on infinite space problems. As the Password is not recoverable, even a system server exposed to all data is another advantage of this method. Hash lookup tables or dictionary attack which made hashing vulnerable will not work here if the numerical value generation method remains closed source or it remains random and we able to use threshold cryptography [17] there. In the future, we plan to publish how to randomize the key protect against lookup and dictionary attacks. The limitation of this method is the Collision rate higher than the hashing methods. But by increasing keys and length we can circumvent this problem, Also adding some other method to create additional uniqueness will be able to solve this problem. In the future, we will work on that.

VI. Conclusion

Using different passwords for each separate online service is a problem from a user perspective. Users typically use the same passwords for multiple services. Any of the services server exposure makes user's other online services vulnerable. Especially as we can see, Hash functions are not effective against quantum computers. Our proposed protocols can able to solve this problem by utilizing the power of Shamir's secret sharing algorithm. In the future, we will use threshold cryptography to improve these methods.

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 Technology*, 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 compromises," *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 verification 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 Computation*, 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.