A KEY MANAGEMENT FOR SECURE CLOUD STORAGE USING A HIERARCHICAL SECRET SHARING SCHEME

Dang Ninh Tran[1,*], Tuyen Nguyen Gia[1], Le Vy Hoang Thi[1], Thu Trang Vu Thi[1]

DOI: http://doi.org/ 10.57001/huih5804.2025.344

ABSTRACT

Although cloud computing has been widely used in recent years, securely sharing data in the cloud remains a major concern. As users face significant security risks in the cloud environment, users have to ensure data confidentiality with cryptographic algorithms while maintaining secure and convenient key distribution to other authorized users. This paper proposes a novel key management method based on a hierarchical secret-sharing scheme for key distribution in a one-to-many encryption scheme. By utilizing Shamir's Secret Sharing scheme, the method efficiently distributes cryptographic keys to the users through interaction, providing a scalable and robust solution for secure key distribution. In this paper, Shamir's Secret Sharing scheme is performed and analyzed. Its processing time is in the acceptable range with small numbers of participants and low threshold values. However, the share generation and key reconstruction with high input values take significantly longer, indicating the need for further improvements to the scheme.

Keywords: *Cloud Encryption, Key Management, Secret Sharing Scheme.*

[1]People's Public Security University of Technology and Logistics, Vietnam
[*]Email: dangninhtran@dhkthc.edu.vn

Received: 30/5/2025
Revised: 29/7/2025
Accepted: 28/9/2025

1. INTRODUCTION

Cloud storage has become one of the most popular services because of its usability, scalability, cost-effectiveness, and convenience. However, it also comes with some security risks such as unauthorized access, cyberattacks, or even insider threats from cloud service providers and third parties [1]. One solution that users can use to keep their data confidential is encrypting their data before uploading it to the cloud. Although there is a challenge in key management in the case of personal use, encryption can largely ensure the confidentiality of data stored in the cloud [2, 3].

In [4], we proposed a user-centric key management for cloud encryption that is significantly effective for individuals. Users only need to remember a short, fixed password, while the derived keys are complex and unique for each working session, enhancing both security and user convenience. However, when data are shared with multiple entities, the one-to-many encryption scheme is more suitable for implementation.

A secret sharing scheme (SSS) is a method that allows data owners to share data with multiple users efficiently while ensuring its confidentiality. Data is divided into a number of shares, and each share is distributed to authorized users. The reconstruction of the original data requires a certain number of shares, which ensures that even if some shares are compromised, the data remains secure unless this threshold is met. In addition to being used for cloud storage, a secret sharing scheme can be applied to cryptographic key distribution [5, 6]. However, equal share distribution is usually implemented so that all shares have to be sent securely, which consumes the time and resources of the entity distributing the shares. Furthermore, there are few works studying the performance of SSS with practical numbers of participants and threshold values.

This paper proposes a key management for secure cloud storage using a hierarchical secret sharing scheme. The hierarchical SSS allows data owners to control the share distribution so that a small number of shares are transmitted from data owner to multiple users efficiently while ensuring confidentiality. The remainder of this paper is structured as follows. Section 2 introduces related knowledge to our proposed method including key derivation function, symmetric-key cryptography, and secret sharing scheme. Section 3 describes our proposed scheme with three algorithms while Section 4 discusses the experimental design and analyzes the corresponding results. Finally, the conclusions are given in Section 5.

2. RELATED WORKS

2.1. Key Derivation Function

A key derivation function (KDF) is used for cryptographic key generation from a given key considered as seed and other information [7]. It is recommended for application with low-entropy and poor random sources which are not suitable to be used directly as cryptographic keys [8]. One of the most popular KDFs is the password-based key derivation function (PBKDF) which generates a large set of high-quality keys from a single user's password and additional data, called $salt$. In our proposed method, we use a PBKDF for producing session keys from a fixed $password$ and the $salt$ which is the hash value h(m_i) of user's data concatenated with a random value IV_i specifying for the current ith working session. One of these algorithms such as Argon2 [9], HKDF [10], PBKDF2 [11], and scrypt [12] can be utilized for this implementation.

2.2. Symmetric-key cryptography

Symmetric key cryptography is a method that encrypts comprehensible data into an unreadable form using secret information, called the cryptographic key, and decrypts it back to the original state with the same secret key or easily derived from it [13]. They are the best solutions for cloud encryption because of fast execution speed, low resource consumption, and high-security level [14, 15]. Our proposed method uses the block cipher AES [16] which has been widely used for data-at-rest protection in Google Cloud Storage [17], Amazon Web Services [18]. In addition, AES can be implemented in Galois/Counter Mode (GCM) and Cipher Block Chaining (CBC) which requires an unpredictable initial vector (IV) for the beginning step of the execution. In Algorithm 1 and Algorithm 3, the encryption and decryption are denoted as $c_i = E_{dKey_i}(m_i, IV_i)$ and $m_i = D_{dKey_i}(c_i, IV_i)$ respectively for the ith working session. The inputs and outputs of these cryptographic processes may have other parameters which depend on the specific operational mode.

2.3. Secret Sharing Scheme

Secret sharing scheme is a method that distributes n pieces, called shares, derived from a secret to a set of n participants in such a way that only specified groups of at least t participants, called a threshold and $t \le n$, can reconstruct the secret by pooling their shares [19]. It is used in many applications such as secure distributed storage, multi-party computation, and fault-tolerant systems [20]. Our proposed scheme uses Shamir's secret sharing [21], one of the most popular SSS. This scheme selects randomly a $(t-1)$ degree polynomial $f(x) = a_0 + a_0 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1}$ in which $a_0 = s$ is the secret represented as an integer. Each s_i is calculated in a prime field Z_p as $s_i = f(i) \bmod p$ where the prime $p > \max(S, n)$, and the total n shares $s_1, s_2, s_3, \ldots, s_n$ will be sent equally to the n participants. Using all the shares from any group of $t'(t' \ge t)$ participants, the sharing secret s is computed with Equation 1.

$$S = \sum_{i=1}^{t'} c_i \times s_i \quad \text{với} \quad c_i = \prod_{\substack{1 \le j \le t' \\ j \ne i}} \frac{s_j}{s_j - s_i} \tag{1}$$

A hierarchical SSS can be derived from the traditional SSS described above by an importance-based distribution. As illustrated in Figure 1, a (t, n) SSS is implemented in which the author A uploads $(t-1)$ shares to the cloud while keeping $(n-t+1)$ shares. The reader B can be given only one share from author A so reader B needs to contact the Cloud Service Provider (CSP) to provide other $(t-1)$ shares required for key reconstruction. While others can get $(t-1)$ shares from CSP, they can not compute the sharing secret if they do not contact author A to get the other share. In this way, author A can either ensure the confidentiality of data stored in the cloud or share it securely with others with permission.

3. PROPOSED METHOD

3.1. Notation

The notation used in the proposed scheme is presented in Table 1.

Table 1. The notation using in the proposed scheme

Notation	Description
c_i	The encrypted data which will be stored in the cloud
$dKey_i$	The derived key at ith working session
$D_{dKey_i}(c_i, IV_i)$	A function decrypting the ciphertext c_i using a symmetric key algorithm with a key $dKey_i$ and initial vector IV_i
$E_{dKey_i}(m_i, IV_i)$	A function encrypting the message m_i using a symmetric key algorithm with a key $dKey_i$ and initial vector IV_i

$f(x, h(m_i) \| \| IV_i)$	A key derivation function uses user's password x and the hash value h(m$_i$) concatenated with random value IV$_i$ as inputs
$h(m_i)$	The hash value of the message m_i
ID_A, ID_B	Identification information of author A, reader B respectively
IV_i	The initial vector using in a symmetric key algorithm
$k_i{}^{csp}$	A set of $(p - 1)$ shares obtained from SSS
k_{ID_B}	A share among $(n - t + 1)$ shares that author A send to reader B
k_{PU_A}, k_{PU_B}	The public key of author A, reader B respectively
m_i	The data which will be encrypted and stored in the cloud
n	The maximum number of shares obtained from $dKey_i$
t	The required number of shares to reconstruct $dKey_i$
$request_1$	The request of reader B send to author A for access permission
$request_2$	The access request of reader B send to CSP for data retrieval
x	The user's password, a fixed string known only by user, which is involved in the key derivation for all working sessions

3.2. Proposed hierarchical secret sharing scheme

Our proposed scheme is an improvement over our previous work [4], which adds a retrieval phase for others who want to access encrypted data. Therefore, it enhances the security and convenience of our proposed user-centric key management and allows cryptographic key sharing with permission. Furthermore, the cryptographic key can be reconstructed without the password, which is useful in case that user forgets this information. All three phases: encryption, retrieval, and decryption phases are illustrated in Figure 1 and described in detail with the pseudo-code of Algorithm 1, Algorithm 2, and Algorithm 3 respectively.

At the encryption phase illustrated in Algorithm 1, a cryptographic key, denoted as $dKey_i$, is generated from the user's password x, the hash value $h(m_i)$ of the message m_i and a random value IV_i specifying the i^{th} session. Then the AES algorithm is performed to encrypt message m_i with the derived key $dKey_i$ and the random value IV_i as an initial vector, depending on its operational mode. The author A uses SSS to produce n shares from $dKey_i$, keep $(n - t + 1)$ shares $(t < n)$, and uploads the rest $(t - 1)$ shares, denoted as $k_i{}^{csp}$, with ciphertext c_i and the hash value $h(m_i)$ to the cloud.

When reader B wants to access the encrypted data, an interaction between reader B, author A, and the CSP must be done, as illustrated in Algorithm 2. A pre-setting in this phase is that author A and reader B know each other's public keys. Reader B first encrypts the data including an identification ID_B and a data access request, denoted $request_1$, and sends it to author A. If author A decrypts the received data, checks the request, and allows reader B to access encrypted data, author A encrypts the data consisting of the identification ID_A and a share k_{ID_B} among $(n - t + 1)$ shares and sends it to reader B. This encrypted communication prevents various cybersecurity attacks, especially interception which explores the share k_{ID_B}. Reader B then sends CSP a request for stored data within related identification ID_A, ID_B, and CSP sends stored data $(c_i \| h(m_i) \| k_i{}^{csp})$ to reader B if all information is valid. Although $(t - 1)$ shares may be disclosed because the communication between read B and CSP is not encrypted, an attacker gains no information of the sharing secret from fewer than t shares. Only reader B gets enough t shares from data that author A and CSP sent; therefore, reader B can construct the secret session key $dKey_i$ using SSS and decrypt c_i to obtain m_i.

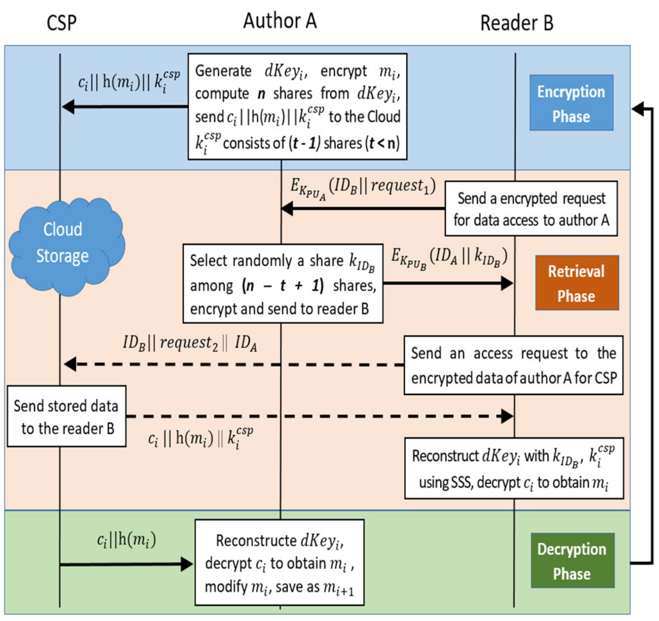

CSP
Author A
Reader B
$c_i \| h(m_i) \| k_i^{csp}$
Generate $dKey_i$, encrypt m_i, compute n shares from $dKey_i$, send $c_i \| h(m_i) \| k_i^{csp}$ to the Cloud
k_i^{csp} consists of $(t - 1)$ shares $(t < n)$
Encryption Phase
$E_{K_{PU_A}}(ID_B \| request_1)$
Send a encrypted request for data access to author A
Cloud Storage
Select randomly a share k_{ID_B} among $(n - t + 1)$ shares, encrypt and send to reader B
$E_{K_{PU_B}}(ID_A \| k_{ID_B})$
Retrieval Phase
$ID_B \| request_2 \| ID_A$
Send an access request to the encrypted data of author A for CSP
Send stored data to the reader B
$c_i \| h(m_i) \| k_i^{csp}$
Reconstruct $dKey_i$ with k_{ID_B}, k_i^{csp} using SSS, decrypt c_i to obtain m_i
$c_i \| h(m_i)$
Reconstructe $dKey_i$, decrypt c_i to obtain m_i, modify m_i, save as m_{i+1}
Decryption Phase

Figure 1. The key management for secure cloud storage using a hierarchical SSS

The decryption phase described in Algorithm 3 is used when author A has to access and/or modify the message m_i. Only the ciphertext c_i and the hash value $h(m_i)$ need to be downloaded from the cloud, and the same key $dKey_i$ is reconstructed by KDF from the user's password, the hash value $h(m_i)$, and the IV_i value stored in the user's device. The $dKey_i$ can also be computed from $(n - t + 1)$ shares in case the password x cannot be provided. However, we prefer using KDF in this phase because of their computational performance while the SSS can act as an effective backup plan, ensuring the safety of the session key.

Algorithm 1: Encryption Phase for Secure Cloud Storage
1: Generate a random value IV_i for the i^{th} working session. **2: Perform** a hash function on message m_i to obtain the hash value $h(m_i)$. **3: Perform** a derivation key function to obtain $dKey_i = f(x, h(m_i) \| IV_i)$. **4: Encrypt** the message m_i using AES algorithm with $dKey_i$ and IV_i to obtain the ciphertext $c_i = E_{dKey_i}(m_i, IV_i)$ in which IV_i is used as an initial vector. **5: Compute** n shares from $dKey_i$ using SSS and keep $(n - t + 1)$ shares ($t < n$). **6: Upload** the encrypted data c_i, the hash value $h(m_i)$ of the message m_i, and the rest $(n - t + 1)$ shares, denoted as ${k_i}^{csp}$, to the cloud.
Algorithm 2: Retrieval Phase of Encrypted Data
1: Reader B securely sends a request for data access to author A. **2: Author A** securely sends a randomly selected share, k_{ID_B}, among $(n - t + 1)$ shares to reader B if author A allows reader B to access data. **3: Reader B** sends an access request, $request_2$, to the encrypted data of author A with the related identification ID_A, ID_B for CSP. **4: CSP** sends stored data $(c_i \| h(m_i) \| {k_i}^{csp})$ to the reader B. **5: Reader B** reconstructs $dKey_i$ from t shares using SSS, decrypts c_i to obtain m_i.
Algorithm 3: Decryption Phase for Secure Cloud Storage
1: Download stored data $(c_i \| h(m_i))$ from the cloud. **2: Perform** the same DKF to reconstruct exactly $dKey_i = f(x, h(m_i) \| IV_i)$. **3: Perform** the Shamir's SSS to compute the secret key $dKey_i$ from $(n - t + 1)$ shares in case of password x cannot be provided. **4: Decrypt** the ciphertext c_i to obtain the plaintext $m_i = D_{dKey_i}(c_i, IV_i)$. **5: Access** the message m_i for reading, modifying, and executing other tasks. **6: Save** a new version of message m_{i+1} after the working session is done. **7: Continue** the encryption phase with the updated message version.

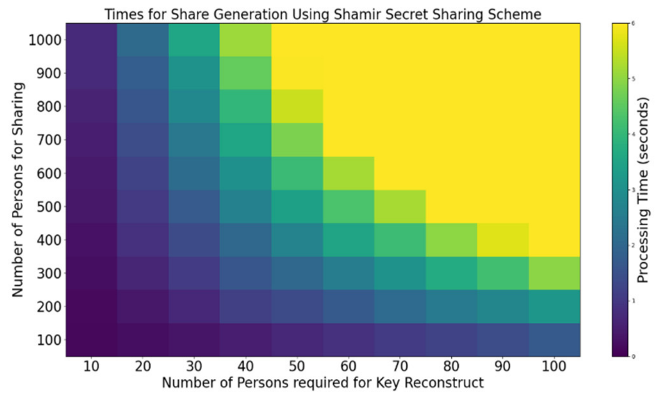
4. EXPERIMENT AND NUMERICAL RESULTS

4.1. Experimental design

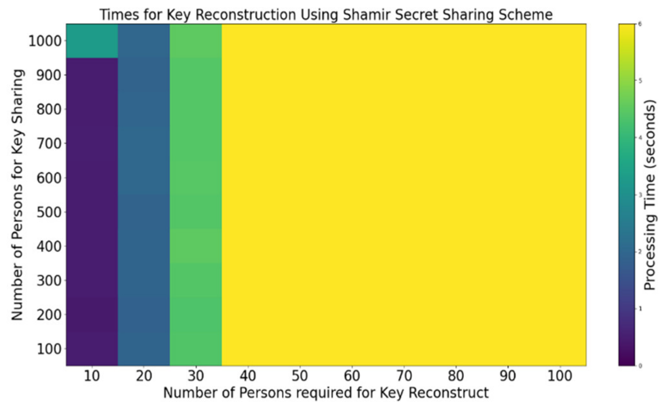
In this paper, we experiment with the share generation and the key construction using Shamir's secret sharing scheme. As the key size of the AES algorithm varies from 128, 192, and 256 bits, the value of secret s can be up to 2^{128}, 2^{192}, and 2^{256} respectively, which is extremely large for processing as a whole. The AES key should be broken into shorter blocks of bits and handled separately to avoid multiprecision arithmetic operations. Therefore, we perform SSS with inputs of 16 bits and the AES key sharing will be done by applying SSS for 8, 12, and 16 blocks of 16 bits. The polynomial representation is used for addition and multiplication in $GF(2^{16})$ because the arithmetic operations with integers still cost significant time and resources even with a short block of 16 bits. We vary the number of participants for secret sharing n from 100 to 1000 while the number of shares required for reconstruction increases from 10 to 100.

4.2. Numerical results

Numerical results are shown by the heat maps in Figure 2. The time required to generate n shares depends on either the number of participants or the threshold t, the number of shares required for secret construction, as shown in Figure 2(a). The time processing rises from nearly zero to 6 seconds in both directions as the colors of the heatmap change from dark purple at the lower-left corner, through green and light green at the center, to bright yellow at the upper-right corner. The colors gradually transition through green and light green as the processing time increases with more participants and higher thresholds. In particular, the increasing threshold value impacts significantly the performance. With an increased of just 10 shares required for key reconstruction, the clear differences in time processing can be seen even with the lowest of participants ($n = 100$). This result shows that the complexity of polynomial $f(x)$, expressed in terms of the degree $(t - 1)$, causes higher time consumption than the number values of $f(x)$ needs to be calculated.



(a) Share Generation



(b) Secret Reconstruction

Figure 2. Processing Time of Shamir Secret Sharing Scheme with 16-bit blocks

While the number of participants affects a little to the share generation, the key reconstruction depends mainly on the threshold value t. As illustrated in Figure 2(b), the color keeps at the same range for threshold t at 10, 20, 30, and from 40 to 100, except for one case of threshold 1000 participants and threshold $t = 10$ at the upper-left corner. It is true because a fixed number of t shares is enough for key reconstruction and the number of participants is not involved in this process, as shown in Equation 1.

Both share generation and key reconstruction have acceptable performance with a small number of participants, below 200, and a low value of threshold, not higher than 30. With a higher demand for sharing, Shamir's SSS needs to be improved for fast speed and low computational complexity.

5. CONCLUSION

This paper introduces a hierarchical secret-sharing scheme for secure key management in cloud storage environments, addressing critical challenges such as efficient data sharing and confidentiality. We have modified the share distribution of traditional Shamir's secret sharing so that our proposed method enhances security for sharing cryptographic keys while reducing the processing load for data owners. Experimental results demonstrate the scheme's feasibility, showing acceptable performance in terms of share generation and key reconstruction under typical use cases. Future research can explore further optimizations to reduce computational complexity, particularly for scenarios with large-scale participants or high threshold values.

REFERENCES

[1]. Alouffi B., Hasnain M., Alharbi A., Alosaimi W., Alyami H., Ayaz M., "A Systematic Literature Review on Cloud Computing Security: Threats and Mitigation Strategies," *IEEE Access*, 57792-57807, 2021.

[2]. Kaanich N., Laurent M., "Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms," *Computer Communications*, 111, 120-141, 2017.

[3]. Yang P., Xiong N., Ren J., "Data Security and Privacy Protection for Cloud Storage: A Survey," *IEEE Access*, 8, 131723-131740, 2020.

[4]. Dang Ninh T., Xuan N. T., Thu N. X., Phuong L. V., "A User-Centric Key Management for Cloud Encryption Using Key Derivation Function," in *The Int. Conf. on Intelligent Systems and Networks*, Ha Noi, Vietnam, 2024.

[5]. Subrahmanyam, R., Rekha, N. R. and Rao, Y. V. S, "Multi-Group Key Agreement Protocol Using Secret Sharing Scheme," *International Journal of Security and Networks*, 18, 3, 143-152, 2023.

[6]. Chhabra S., Singh A. K., "Security Enhancement in Cloud Environment using Secure Secret Key Sharing," *Journal of Communications Software and Systems*, 16, 4, 296-307, 2020.

[7]. L. Chen, *Recommendation for Key Derivation Using Pseudorandom Functions SP 800-108r1*. National Institute of Standards and Technology, Gaithersburg, MD, USA, 2022.

[8]. Turan M. S., Barker E., Burr W., Chen L., *Recommendation for Password-Based Key Derivation SP 800 - 132*. National Institute of Standards and Technology, Gaithersburg, MD, USA, 2010.

[9]. Biryukov A., Dinu D., Khovratovich D., Josefsson S., *Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications RFC 9106*. Internet Engineering Task Force, 2021.

[10]. Krawczyk H., Eronen P., *HMAC-based Extract-and-Expand Key Derivation Function (HKDF).* Internet Engineering Task Force, 2010.

[11]. Kaliski B., *PKCS#5: Password-Based Cryptography Specification Version 2.0. Internet* Engineering Task Force, 2000.

[12]. Percival C., Josefsson S., *The scrypt Password-Based Key Derivation Function*. Internet Engineering Task Force, 2016.

[13]. Stinson D. R., *Cryptography: Theory and Practice.* CRC Press, Taylor & Francis Group: Boca Raton, FL, USA, 2019.

[14]. Kong J. H., Ang L. M., Seng K. P., "A Comprehensive Survey of Modern Symmetric Cryptographic Solutions For Resource Constrained Environments," *Journal of Network and Computer Applications,* 49, 15-50, 2015.

[15]. Lohachab A., Lohachab A., Jangra A., "A Comprehensive Survey of Prominent Cryptographic Aspects For Securing Communication In Post-quantum IoT Networks," *Internet of Things,* 9, 100174, 2020.

[16]. *Advanced Encryption Standard (AES) FIPS 197*. National Institute of Standards and Technology, Gaithersburg, MD, USA, 2001.

[17]. *Google Cloud Security Whitepapers*. Google, 2018.

[18]. *AWS Key Management Service: AWS KMS Cryptographic Details*. Amazon, 2024.

[19]. Calkavur S., Bonnecaze A., Cruz R. D., Sole P., *Code Based Secret Sharing Schemes: Applied Combinatorial Coding Theory.* World Scientific Pub. Co. Inc., Toh Tuck Link, Singapore, 2022.

[20]. Krenn S., Lorünser T., *An Introduction to Secret Sharing: A Systematic Overview and Guide for Protocol Selection.* Cham, Switzerland: Springer Nature, 2023.

[21]. Shamir A., "How to share a secret," *Communication of the ACM,* 22, 11, 612-613, 1979.