Revisiting Secure Multi-Server Oblivious RAMs

Chaewon Kwak, Kangmo Ahn
Department of Computer Science and Engineering
Korea University
Seoul, South Korea
chaewonkwak0905@gmail.com, kmahn@isslab.korea.ac.kr

Changhee Hahn
Department of Electrical and Information Engineering
Seoul National University of Science and Technology
Seoul, South Korea
chahn@seoultech.ac.kr

Dongyoung Koo
Department of Convergence Security
Hansung University
Seoul, South Korea
dykoo@hansung.ac.kr

Junbeom Hur
Department of Computer Science and Engineering
Korea University
Seoul, South Korea
jbhur@korea.ac.kr

***Abstract*—Oblivious RAM (ORAM) facilitates search and update on outsourced encrypted databases without leaking access patterns. Since ORAM typically requires large client storage and high computational overhead, many studies proposed more efficient ORAM schemes. For example, Thang et al. proposed a multi-server ORAM, S^{3}ORAM, by utilizing Shamir's secret sharing and secure multi-party computation, instead of fully homomorphic encryption to enhance the efficiency. However, recent studies show that Shamir's secret sharing is no longer secure in Galois Fields. To solve this security problem while achieving the efficiency, we propose two ORAM schemes: (1) BSS-ORAM by applying Blakley's secret sharing to S^{3}ORAM, aiming to improve security with high efficiency, and (2) VH-BSS-ORAM, which further prevents size pattern leakage by storage padding. According to our analysis, BSS-ORAM is secure in Galois Fields with the same computational overhead of S^{3}ORAM; and VH-BSS-ORAM further hides size pattern with higher overhead compared to S^{3}ORAM.**

***Index Terms*—Oblivious RAM, Blakley's secret sharing, searchable encryption, information leakage**

I. Introduction

Oblivious RAM (ORAM) was first proposed to enhance the security of searchable encryption schemes [1] by concealing the access pattern to the remote storage. The operations within ORAM are classified into two catergories: *oblivious read* which involves reading data beyond the requested one, and *oblivious write* which incorporates shuffling the read data to random locations. Dynamic Symmetric Searchable Encryption (DSSE) schemes [2] enable clients to search on encrypted databases without the need for a separate decryption procedure. DSSE aims to prevent untrusted server from obtaining any information related to the data. However, the use of keywords in client searches and the corresponding responses from the server can still provide some linkabilities between different queries [3], which allows an honest-but-curious server to find search patterns by discovering correlations for the same keyword. By observing how often the client accesses the same data during the search and update process, the server can ascertain correlations pertaining to the access patterns, which is then used to recover the hidden keywords with auxiliary information [4]. To mitigate the search pattern and access pattern leakage, ORAM is applied to DSSE as it thwarts the server from observing the client's behavior by performing obfuscating operations.

Layer-based ORAM and tree-based ORAM are two traditional types of ORAM. Tree-based ORAM shows higher efficiency than layer-based one due to its lower communicational and computational costs during the oblivious write phase [5], [6]. In tree-based ORAM, specifically, metadata (e.g., encrypted indices) is stored in a tree structure and uploaded to the server. When a client searches for a keyword, the server returns a path together with the corresponding data, and the client can figure out the correct one from the received bulk of data. ORAM provides strong security by obfuscation per every data access, but existing schemes have generally sought to achieve an $O(\log N)$ client-server bandwidth where N is the number of keyword-document pairs, or $O(1)$ communication overhead by leveraging the expensive homomorphic encryption. To achieve $O(1)$ bandwidth without costly fully or partially homomorphic encryptions, Thang et al. [7] proposed S^{3}ORAM, a multi-server ORAM that leverages Shamir's secret sharing scheme [8]. S^{3}ORAM harnesses multi-party computation in a distributed setting to reduce the computational complexity of ORAM. Moreover, it reinforces its security by securely dividing and storing data across multiple servers.

Motivations. Although S^{3}ORAM has made significant progress in enhancing the efficiency and security of existing ORAMs by taking advantage of a distributed setting, it still faces a critical security challenge caused by the use of Shamir's secret sharing under Galois Fields [9]. Galois Fields, a mathematical structure consisting of a finite number of binary bits that support XOR and AND operations, are widely used in cryptographic protocols and algorithms. In Shamir's (n, t)-threshold secret sharing scheme, a secret is divided into

This work was supported as part of Military Crypto Research Center(UD170109ED) funded by Defense Acquisition Program Administration(DAPA) and Agency for Defense Development(ADD).

n shares, and more than t shares are required to reconstruct the secret. However, Shamir's scheme selects n hyperplanes to generate n shares in polynomial way, making it vulnerable to brute-force attacks under Galois Fields. For instance, Shamir's secret sharing offers only 1,024 choices to select n shares in GF(2^{10}), which is susceptible to brute-force attacks. As shown in Table I, the state-of-the-art client-efficient multi-server ORAMs (i.e., [5], [7]) are insecure under Galois Fields, while single-server ORAMs (i.e., [10], [11]) are secure. We note that there is no client-efficient multi-server ORAM which is secure under Galois Fields.

Proposed Approach. In order to address the problem, we propose a novel ORAM method by utilizing Blakley's secret sharing scheme [12]. Similar to Shamir's scheme, Blakley's scheme selects n coefficient vectors to generate n shares of a secret, and more than t shares should be gathered for the reconstruction. In contrast to Shamir's scheme, which polynomially chooses n hyperplanes, Blakley's secret sharing utilizes a random selection approach. For example in GF(2^{10}), Blakley's secret sharing offers $t \times 2^{10}$ options for selecting coefficient vectors, surpassing those of Shamir's scheme. Thus, Blakley's scheme makes ORAM more resilient to brute-force attack by the malicious server, when t is large enough. Therefore, we construct our method by appling Blakley's secret sharing to S^{3}ORAM, instead of Shamir's secret sharing.

Our Contribution. After S^{3}ORAM was proposed, lots of works have proposed ORAM-based DSSE schemes in a distributed setting to take its advantages of computational and communicational efficiency. Accordingly, revisiting the security of S^{3}ORAM, which serves as a foundational scheme for many other multi-sever ORAM and identifying its vulnerability under Galois Field have significant implications. In this regards, we propose BSS-ORAM by harnessing Blakley's secret sharing on S^{3}ORAM, which has the same level of time complexity as Shamir's secret sharing, but overcomes its security concern. To the best of our knowledge, BSS-ORAM is the first secure multi-server ORAM under Galois Fields. Furthermore, we propose VH-BSS-ORAM by applying storage padding to further hide the size patterns with higher overhead.

II. Blakley's Secret Sharing

Secret sharing schemes divide a secret among multiple parties, ensuring that only predefined subsets of the parties can collectively reconstruct the secret [13]. This concept is applicable in various cryptographic scenarios, including multi-party computation and threshold cryptography [14], [15]. Shamir's secret sharing scheme [8] divides a secret into n shares, with more than t shares required for successful secret reconstruction. Blakley's scheme is also a kind of the classical (n, t)-threshold secret sharing scheme [12].

A. BSS-Protocol

Blakley's secret sharing (BSS) uses a graph-based approach employing hyperplanes. In this scheme, a secret is represented as a specific point on hyperplanes. The intersection of more than t hyperplanes helps reconstructing the secret point, without requiring knowledge of each individual hyperplane. We note that each of the coefficients of n hyperplanes represents each of the n shares. Specifically, a secret matrix S is divided into n shares in the form of a linear system $Sx \mod p = y$, where p is a prime number [16]. According to geometrical principles, the intersection of over t non-parallel hyperplanes results in a certain point. Encoding S as any single coordinate of this intersection point allows the reconstruction of the secret via these hyperplanes' intersection. When we generate n hyperplanes with n random coefficient vectors in the form of $Sx \mod p = y$, the secret matrix S is ready to be securely shared. Algorithm 1 shows how Blakley's secret sharing works.

Algorithm 1 Blakley's Secret Sharing Protocol

($[\![\alpha]\!]_1$, ... , $[\![\alpha]\!]_n$) $\leftarrow$ **BSS.Create**(α, t): Create t-private shares of α
1: **for** i = 1, ... , n **do**
2: **for** j = 1, ... , t **do**
3: $a_j \xleftarrow{\$} \mathbf{F}_p$
4: $C \leftarrow RandomPermutation(\mathrm{a}_1, ...a_t)$
5: $[\![\alpha]\!]_i \leftarrow Concatenate(C)$
6: **return** ($[\![\alpha]\!]_1$, ... , $[\![\alpha]\!]_n$)

$\alpha \leftarrow$ **BSS.Recover**(A,t): Recover the value α from $k \geq t+1$ shares
1: **if** A.length $\geq t+1$ **then**
2: $\alpha \leftarrow$ [0]* t
3: **for** i = 1, ... , k **do**
4: $\alpha[i] \leftarrow \alpha[i] \oplus [\![\alpha]\!]_i$
5: **return** α

As Blakley's scheme relies on a geometric approach with high resilience to brute-force attacks, it is appropriate for managing complex access structures. As a result, we utilize Blakley's secret sharing scheme to construct a secure ORAM method for DSSE in cloud computing that securely distributes a secret among multiple severs, ensuring that the secret can only be reconstructed by the client, unless all servers collude to attack the client.

B. Comparison with Shamir's Secret Sharing

We compare Blakley's secret sharing with Shamir's secret sharing from the aspects of scalability, security, and computational complexity.

Scalability. The primary difference between Blakley's secret sharing and Shamir's secret sharing lies in their approach to selecting n coefficient vectors for n shares. Shamir's method involves polynomials, while Blakley's method takes a geometric approach utilizing hyperplanes. In Shamir's secret sharing, the shares of a secret are selected from a finite field $\mathbf{F}_p$ in which p is a prime number greater than n. Once n distinct coefficient vectors have been successfully chosen, the client selects t random elements from $\mathbf{F}_p$ to define the secret as a polynomial $P(x) = k + \Sigma_{i=1}^{t} a_i x^i$. To recover the secret, Shamir's secret sharing uses Lagrange interpolation. In contrast, Blakley's scheme defines a secret with hyperplanes in a t-dimensional space as $a_1x_1 + a_2x_2 + ... + a_tx_t = b$. By intersecting any t of these hyperplanes, the secret can be reconstructed. Due to the utilization of hyperplane geometry,

Blakley's scheme is more scalable than Shamir's secret sharing when extending the number of servers in a multi-server setting.

Security. From a security aspect, Shamir's secret sharing offers significantly fewer choices for coefficient vectors compared to Blakley's secret sharing. Shamir's polynomial algorithm generates a matrix composed of coefficient vectors $(1,\ x,\ x^2,\ x^3,\ \ldots\ ,\ x^{n-1})$, resulting in a finite number of distinct coefficient vectors under Galois Fields, whereas it is infinite under rational number fields [9]. On the other hand, Blakley's secret sharing provides a greater number of choices by randomly selecting n coefficient vectors to generate n hyperplanes, which makes it resilient to the brute-force attack when t is large. Thus, Blakley's scheme is secure under Galois Fields in contrast of Shamir's scheme.

Computational Cost. Generally, the most significant and fundamental operation in secret sharing schemes is matrix multiplication. Blakley's and Shamir's schemes generate matrices during their share generation and division phases, yet other operations are almost the same. Since the complexity of protocol depends on the most demanding computation, the matrix multiplications, both schemes have equivalent level of computational costs.

III. Proposed Schemes

In this section, we propose two ORAM schemes, Blakley's secret sharing-based ORAM (BSS-ORAM) and volume-hiding BSS-ORAM (VH-BSS-ORAM), by applying it to the search and access algorithms in S^{3}ORAM [7].

BSS-ORAM. Figure 1 shows how BSS-ORAM operates in client-server environments. Since the internal structures of two secret sharing schemes differ, the method used to compute the share on the client side is modified as follows. First, the client selects coefficient vectors in a random and geometric manner from a finite field F_p. Then, the client performs an exclusive-OR with coefficient vectors and original query, resulting in temporary vectors **T**. Next, the client conducts a matrix multiplication with random matrix **R**, composed of elements from the coefficient vectors and temporary vectors, resulting in a matrix **RS**. When the client concatenates **R** and **RS** to construct the final matrix, each row of the final matrix is divided into n shares. For each data access, the client performs all these operations using the BSS.Create protocol in Algorithm 1, and sends the generated shares to n servers. On receipt of a query, n servers perform computations on the stored data and the query shares. The servers have the same data structure, but the data is split into n shares and stored across n servers. Once each server responds to the client with its respective share, the client use the BSS.Recover protocol in Algorithm 1 to find the response of its original query.

VH-BSS-ORAM. Although BSS-ORAM provides higher security in Glois Field without performance degradation of the baseline scheme, S^{3}ORAM, it still leaks size pattern information which is regarded as important privacy violation [5], [17]. To address the issue, we additionally propose VH-BSS-ORAM that eliminates size patterns by storage padding. Size patterns are revealed when the server figures out the linkability between search queries with the same response length by observing the queries. In VH-BSS-ORAM, the response is padded such that its length is fixed to the maximum number of matching documents. Considering that ORAM returns all data along the accessed path, we design VH-BSS-ORAM to always return the maximum number of paths corresponding to the keyword. Thus, the size pattern leakage is prevented due to the indistinguishability among each response.

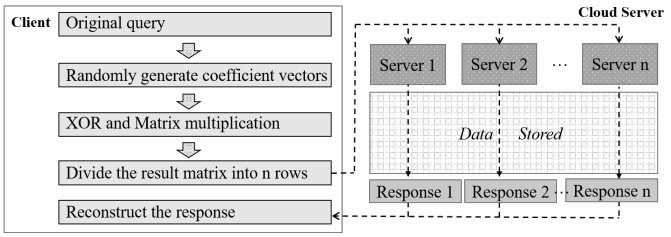

Fig. 1. BSS-ORAM Overview

IV. Analysis

Table I provides the comparative analysis result among different ORAM schemes, where APH, SPH, and ZPH represent whether the scheme hides access patterns, search patterns, and size patterns, respectively. The GF indicates whether the scheme is secure under Galois Fields, with 'o' denoting secure and 'x' denoting non-secure schemes. BSS-ORAM replaces Shamir's secret sharing with Blakley's secret sharing without altering the other foundational designs, thereby preserving $O(1)$ bandwidth and $\tilde{O}(n_w)$ computational cost. While the polynomial approach of Shamir's secret sharing is vulnerable under Galois Fields, it offers the advantage of simplicity and low space complexity. By polynomially selecting n hyperplanes, the client can significantly reduce space when generating shares. Table I displays the associated costs with recursive S^{3}ORAM and recursive BSS-ORAM. The typical approach to reduce the client storage overhead is recursive ORAM, which stores mapping information recursively in smaller ORAM trees. In the case of recursive S^{3}ORAM and recursive BSS-ORAM, the difference in required storage between both secret sharing schemes is negligible. Although Blakley's secret sharing requires the client to store $t\times$ more hyperplane candidates, this has minimal impact on the client storage. Consequently, the proposed BSS-ORAM satisfies the security of Galois Fields through the use of Blakley's secret sharing, maintaining the efficiency of S^{3}ORAM.

VH-BSS-ORAM pads the response length to the maximum to eliminate size patterns. Although the bandwidth, communication, and computation costs of VH-BSS-ORAM inevitably increase due to the storage padding as a trade-off, it provides the strongest security against size pattern leakage-abuse attacks. Therefore, in the cloud environment where the size pattern leakage is a major threat, VH-BSS-ORAM can be a proper solution if the computational resource is not significantly limited. Consequently, the proposed BSS-ORAM and VH-BSS-ORAM are client-efficient multi-server ORAM

TABLE I
COMPARISON OF ORAM SCHEMES

Scheme	Bandwidth	Communication Cost	Computation Cost	Client Storage	APH	SPH	ZPH	GF
Path ORAM [10]	$O(\log N)$	$O(\log^2 N)$	$O(\log^2 N)$	$O(N)$	o	o	x	o
Path ORAM (recursive) [10]	$O(\log^2 N)$	$O(\log^3 N)$	$O(\log^3 N)$	$O(\log N) * \omega(1)$	o	o	x	o
Orion [11]	$O(\log^2 N)$	$O(n_w \log^2 N)$	$O(n_w \log^2 N)$	$O(1)$	o	o	x	o
S^{3}ORAM [7]	$O(1)$	$\tilde{O}(\log^2 N)$	$\tilde{O}(n_w)$	$O(1)$	o	o	x	x
Eurus [5]	$O(\log^2 K)$	$O(F^2 \log^2 N)$	$O(max\{n_w, F\} + K)$	$O(K)$	o	o	o	x
Proposed								
BSS-ORAM	$O(1)$	$\tilde{O}(\log^2 N)$	$\tilde{O}(n_w)$	$O(1)$	o	o	x	o
VH-BSS-ORAM	$O(\log N)$	$O(M \log^2 N)$	$O(M \log N)$	$O(K)$	o	o	o	o

(N: number of (keyword, document) pairs, K: number of distinct keywords, F: number of files, M: maximum number of corresponding file identifiers to the index, n_w: number of files containing keyword w, APH: Access Pattern Hiding, SP: Search Pattern Hiding, ZP: siZe Pattern Hiding, GF: Galois Fields security.)

schemes secure under Galois Fields, which protect well-known information leakages, such as the search pattern and the size pattern.

V. CONCLUSION AND FUTURE WORKS

We propose two multi-server ORAM schemes, BSS-ORAM and VH-BSS-ORAM, both secure under Galois Fields and utilizing Blakley's secret sharing. To the best of our knowledge, BSS-ORAM is the first ORAM scheme achieving $\tilde{O}(n_w)$ computational cost, ensuring resistance against brute-force attacks under Galois Fields. Additionally, VH-BSS-ORAM goes a step further by preventing size pattern leakage.

Oblivious RAMs play a key role in preventing information leakage in DSSE among the various trade-offs between security and performance. While our methods enhance security through the implementation of Blakley's secret sharing, such primitives can be replaced or modified depending on the system environment, taking into account security threats and computational limitations. Adapting more advanced secret sharing algorithms to multi-server ORAM, and understanding its implications will provide valuable insights and contribute to performance and security improvements, which will be important future works.

REFERENCES

[1] O. Goldreich, "Towards a theory of software protection and simulation by oblivious rams," in *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pp. 182–194, 1987.

[2] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 965–976, 2012.

[3] C. Liu, L. Zhu, M. Wang, and Y.-a. Tan, "Search pattern leakage in searchable encryption: Attacks and new construction," *Information Sciences*, vol. 265, pp. 176–188, 2014.

[4] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: ramification, attack and mitigation.," in *Ndss*, vol. 20, p. 12, Citeseer, 2012.

[5] Z. Liu, Y. Huang, X. Song, B. Li, J. Li, Y. Yuan, and C. Dong, "Eurus: towards an efficient searchable symmetric encryption with size pattern protection," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 2023–2037, 2020.

[6] Z. Wu and R. Li, "Obi: a multi-path oblivious ram for forward-and-backward-secure searchable encryption.," in *NDSS*, 2023.

[7] T. Hoang, C. D. Ozkaptan, A. A. Yavuz, J. Guajardo, and T. Nguyen, "S3oram: A computation-efficient and constant client bandwidth blowup oram with shamir secret sharing," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 491–505, 2017.

[8] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[9] S. Chen, Y. Chen, H. Jiang, L. T. Yang, and K.-C. Li, "A secure distributed file system based on revised blakley's secret sharing scheme," in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 310–317, IEEE, 2012.

[10] E. Stefanov, M. v. Dijk, E. Shi, T.-H. H. Chan, C. Fletcher, L. Ren, X. Yu, and S. Devadas, "Path oram: an extremely simple oblivious ram protocol," *Journal of the ACM (JACM)*, vol. 65, no. 4, pp. 1–26, 2018.

[11] J. Ghareh Chamani, D. Papadopoulos, C. Papamanthou, and R. Jalili, "New constructions for forward and backward private symmetric searchable encryption," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1038–1055, 2018.

[12] G. R. Blakley, "Safeguarding cryptographic keys," in *Managing Requirements Knowledge, International Workshop on*, pp. 313–313, IEEE Computer Society, 1979.

[13] A. Beimel, "Secret-sharing schemes: A survey," in *International conference on coding and cryptology*, pp. 11–46, Springer, 2011.

[14] C.-C. Yang, T.-Y. Chang, and M.-S. Hwang, "A (t, n) multi-secret sharing scheme," *Applied Mathematics and Computation*, vol. 151, no. 2, pp. 483–490, 2004.

[15] I. N. Bozkurt, K. Kaya, A. A. Selcuk, and A. M. Güloglu, "Threshold cryptography based on blakley secret sharing," *Information Sciences*, pp. 1–4, 2008.

[16] A. Shamsoshoara, "Overview of blakley's secret sharing scheme," *arXiv preprint arXiv:1901.02802*, 2019.

[17] Y. Zhao, H. Wang, and K.-Y. Lam, "Volume-hiding dynamic searchable symmetric encryption with forward and backward privacy," *Cryptology ePrint Archive*, 2021.