International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

ENABLING EFFICIENT SECURE AND PRIVACY-PRESERVING MOBILE CLOUD STORAGE

Arun kumar.V[1], A. Bhargavi[2], M. Yaswitha[3], P. manovya[4]

[1] Assistant Professor, Department of IT,Malla Reddy Engineering College For Women (UGC-Autonomous), Maisammaguda,Dhulapally,Secunderabad,Telangana-500100

[234]UG Scholar, School of CS,Malla Reddy Engineering College for Women, (UGC-Autonomous), Maisammaguda,Dhulapally,Secunderabad,Telangana-500100

ABSTRACT

Mobile cloud storage (MCS) provides a convenient way for clients to store data in the cloud. In this paper, we present an efficient, secure, and privacy-preserving MCS solution that ensures both data confidentiality and privacy, with a particular focus on protecting access patterns. At the heart of our approach is the Oblivious Selection and Update (OSU) protocol. This protocol uses onion additively homomorphic encryption with fixed encryption layers, allowing the client to retrieve an encrypted data item from the cloud and update it with a new value. The process generates a small encrypted vector, which significantly reduces both the client's computational load and communication costs. Compared to existing methods, our scheme offers several benefits, such as fine-grained data structures with smaller item sizes, minimal computational effort on the client side (requiring only a few homomorphic operations), and constant communication overhead, making it well-suited for MCS applications. Furthermore, by utilizing the "verification chunks" technique, our scheme provides verifiability to prevent malicious behavior from the cloud. Our experimental results demonstrate that our solution is more efficient than previous oblivious storage methods in terms of both client and cloud workloads.

Keywords- Mobile Cloud Storage (MCS),Data confidentiality, Oblivious Random Access Machine (ORAM) Cloud Storage Efficiency, Cloud Computing Security

I. INTRODUCTION

Mobile Cloud Storage (MCS) allows users to store and access data from the cloud using mobile devices, making it a highly convenient and widely adopted service. Major companies like Apple iCloud, Dropbox, Microsoft OneDrive, and Google Drive offer MCS services, which are increasingly popular for both personal and business use. However, despite its advantages, many cloud services are not fully trusted, leading clients to encrypt their data before uploading it to the cloud to protect its confidentiality.

In MCS applications, especially those related to location-based services, the data being accessed may leak additional information about the user's behavior or context, such as their location or activity. This access pattern leakage could potentially allow the cloud server to infer sensitive details about the client's actions or the content of their encrypted data. For example, in a searchable encryption system, the cloud could recognize

up to 80% of search queries based on the access patterns alone.

To address this, oblivious technologies like Oblivious Transfer (OT), Oblivious Storage (OS), and Oblivious Random Access Machine (ORAM) have been introduced. These technologies allow clients to access their encrypted data stored on an untrusted cloud without revealing which specific items are being accessed or what operations are being performed. These methods provide high privacy protection and have been applied in various scenarios, such as searchable encryption, encrypted hidden volumes, and multi-party computation.

However, employing existing oblivious schemes in MCS scenarios presents several challenges. Mobile devices typically have limited communication bandwidth, which makes schemes with high communication overhead unsuitable. Furthermore, mobile devices have less computational power compared to personal computers, meaning schemes requiring complex cryptographic operations would reduce device performance and battery life. Additionally, many oblivious schemes have larger minimum effective item sizes, making fine-grained access more difficult and increasing communication or computation overhead.

Some oblivious schemes aim to improve efficiency by considering data locality, such as spatial and temporal locality. Spatial locality refers to the tendency of clients to access nearby data, while temporal locality refers to the repeated access of the same data within a short time. However, most existing oblivious schemes do not incorporate temporal locality, which could significantly enhance efficiency.

In this paper, we propose an efficient, secure, and privacy-preserving mobile cloud storage scheme that addresses these challenges. Our scheme protects both data confidentiality and access pattern privacy, ensures constant communication bandwidth overhead, minimizes client-side computation, and maintains small item sizes. Additionally, our scheme incorporates temporal locality to improve efficiency and uses a verification method to resist malicious behavior from the cloud. The proposed solution outperforms previous works in terms of efficiency and practicality for mobile cloud storage applications.

II. RELATED WORK

Goldreich and Ostrovsky pioneered the concept of Oblivious Random Access Machine (ORAM) as a way to safeguard access pattern privacy in systems like cloud storage. Their initial contribution, the Square Root ORAM, demonstrated that it was impossible to achieve communication without a lower bound blow-up of at least $\Omega(\log N)$. In their design, the memory (or cloud storage in cloud computing scenarios) was treated as a passive storage entity, meaning it merely stored data and did not perform any computations on it. Their work set the foundation for subsequent improvements in both theoretical and practical aspects of ORAM.

One of the key developments in this area was proposed by Shi et al., who structured the storage mechanism as a binary tree of buckets.

This design allowed for an efficient retrieval of data while maintaining a communication cost of $O(\log^3 N)$ in the worst case. Later, Stefanov et al. introduced Path ORAM, which optimized the binary tree framework. Path ORAM retained the communication overhead lower bound of $\Omega(\log N)$ while simplifying the architecture and making it more efficient by avoiding complex cryptographic operations. It was particularly beneficial in terms of achieving lower end-to-end delays for practical parameters.

As cloud computing capabilities evolved and became more powerful, there was a shift toward utilizing the cloud's computational resources to assist with data retrieval and processing, rather than relying solely on the client's device. This move opened the door for a new class of schemes that aimed to bypass the lower communication bound by enabling the cloud to perform computations on behalf of the client. One notable development was introduced by Apon et al., who formalized the concept of **verifiable oblivious storage**. This extended the original ORAM model by allowing the storage medium (in this case, the cloud) to execute computations, which previously was not possible in the passive setting. This new approach provided greater flexibility and efficiency.

Building on these advancements, Devadas et al. introduced **Onion ORAM**, a system that achieved constant communication bandwidth by employing multi-layer additively homomorphic encryption (or sometimes somewhat homomorphic encryption). In this approach, data blocks were encrypted in layers that resembled an "onion," allowing the client to retrieve or evict data blocks through encrypted vectors that were computationally lightweight. The security of this system was bolstered by the use of reverse lexicographical eviction orders, ensuring that the probability of overflow remained negligible for properly chosen security parameters.

Another important approach was proposed by Moataz et al. with **C-ORAM**, which also aimed to achieve constant communication bandwidth. Unlike Onion ORAM, which relied on layered homomorphic encryption, C-ORAM introduced a more efficient oblivious merging technique, which replaced the homomorphic encryption layers. This optimization led to better overall performance, reducing complexity and increasing the system's efficiency.

In summary, while early ORAM schemes were built on the premise of passive cloud storage, modern iterations have embraced the cloud's computational power to improve efficiency and reduce communication overhead. Innovations such as Onion ORAM and C-ORAM have shown how combining advanced cryptographic techniques with cloud computation can address performance bottlenecks and enhance both privacy and efficiency in cloud storage systems. These developments have laid the groundwork for building more sophisticated and secure storage systems, which are essential for the privacy-sensitive applications in today's cloud-based world.

III. IMPLEMENTATION

To implement a system for **secure and privacy-preserving mobile cloud storage**

with the concepts like **Oblivious Random Access Machine (ORAM)**, **Onion ORAM**, and **homomorphic encryption**, we need to focus on several aspects, including encryption, access pattern privacy, and efficient communication. Below is an outline and basic steps for implementing such a system. **.**

1.System Architecture

The system consists of three primary components: the **client**, the **cloud server**, and the **encryption protocol**. The client is responsible for encrypting data before uploading it to the cloud and decrypting it when retrieved. The cloud server stores encrypted data and facilitates the retrieval and update operations without access to the plaintext data. The encryption protocol ensures that the access patterns remain hidden from the cloud, thereby preserving user privacy.

2. Data Encryption

The client utilizes **additively homomorphic encryption (AHE)** to encrypt data before uploading it to the cloud. AHE enables the cloud to perform certain operations (like retrieval and updates) on encrypted data without decrypting it. For efficient encryption, we employ **multi-layer onion encryption**, where each data block is encrypted with several layers of encryption, preventing the cloud from inferring any meaningful information about the data.

3. Oblivious Access Protocol

The core of the implementation is the **Oblivious Selection and Update (OSU) protocol**. OSU allows the client to access or update encrypted data stored on the cloud without revealing which data is being accessed or modified. The protocol ensures that the cloud is oblivious to both the content of the data and the access patterns. It achieves this by generating small encrypted vectors that represent the data, which significantly reduces the computation and communication overhead for the client.

4. Communication Efficiency

Since mobile devices often operate under constrained bandwidth conditions, minimizing the communication overhead is crucial. The OSU protocol is designed to ensure **constant communication overhead** regardless of the size of the data, achieving a constant time complexity of **O(1)** for both data retrieval and updates. This ensures that mobile users can interact with the cloud efficiently, even on low-bandwidth networks.

5. Client-Side Computation

To ensure the system is feasible for mobile devices with limited processing power, the client-side computation is minimized. The client only performs basic homomorphic operations for encryption and decryption, which require minimal computational resources. This is in contrast to more computationally intensive schemes like **Fully Homomorphic Encryption (FHE)**, which would be unsuitable for mobile devices due to their significant computational overhead.

6. Data Integrity and Cloud Verifiability

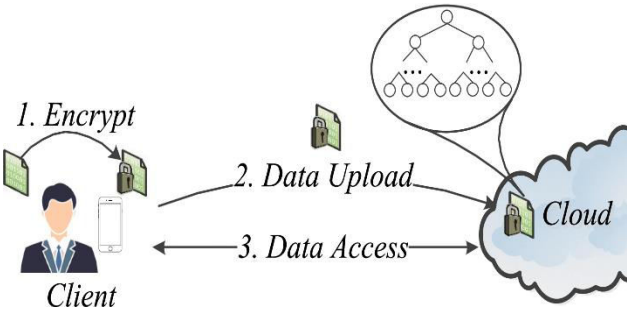
To prevent malicious behavior by the cloud, the system incorporates a **verification mechanism**. The "verification chunks" method is used to periodically check the integrity of the stored data and ensure that the cloud is executing the correct operations without

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

tampering with the data. This provides the client with the assurance that their data remains secure, even in the presence of potentially malicious cloud services.

7. Temporal Locality

In order to further optimize the system, **temporal locality** of data access is considered. The system predicts that data items accessed recently are more likely to be accessed again in the near future. By leveraging this, we reduce the communication and computation overhead for frequently accessed data, ensuring efficient retrieval and update operations.



System Architecture

IV. ALGORITHM USED

1.Oblivious Selection and Update Protocol (OSU)

In the OSU protocol, the client can select and update encrypted data without revealing which data item is being accessed. The core of this is based on Oblivious Transfer (OT) or similar protocols.

The formula for oblivious selection typically involves a randomization technique to ensure the client is oblivious to the choice being made and that no one can learn the outcome without knowledge of the key.

$$\mathrm{OT}_{\text{sender}}(x_0, x_1, r) \rightarrow \mathrm{OT}_{\text{receiver}}(x_r)$$

Where:

- x_0, x_1 are two possible data values.
- r is a random bit chosen by the receiver, who will receive either x_0 or x_1 based on their choice.
- The receiver remains oblivious to the choice made by the sender.

2. Additive Homomorphic Encryption

Additive Homomorphic Encryption allow computations to be performed on encrypted data without the need to decrypt it. A simple formula for additive homomorphic encryption, which supports operations like addition, is:

$$E_k(m_1 + m_2) = E_k(m_1) + E_k(m_2)$$

Where:

- E_k represents the encryption of message m under a key k.
- m_1 and m_2 are the plaintext messages.
- The encrypted sum $E_k(m_1 + m_2)$ can be computed directly without needing to decrypt the individual messages first, which allows secure data operations in the cloud.

3.Verification Chunks Method

The Verification Chunks method relies on hashing each data chunk to create a fingerprint of the data, ensuring integrity.

The formula for a hash function HHH could be:

$$H(D) = h_1 \| h_2 \| \ldots \| h_n$$

Where:

- D represents the data being stored.
- $h_1, h_2, \ldots, h_n$ are hash values for each chunk of data.
- The "||" symbol represents concatenation, creating a composite hash for the entire data set.

4. Temporal Locality Consideration

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

Temporal locality optimizes the data retrieval process by focusing on recently accessed data. This is usually managed by using a **Least Recently Used (LRU)** or **Most Recently Used (MRU)** cache strategy. In an MCS scenario, this could involve maintaining a priority queue where the most recently accessed data items are stored, and the formula for maintaining the cache could be:

$$Q(t) = \text{LRU}(Q(t-1), d_{\text{new}})$$

Where:

- $Q(t)$ represents the queue at time t, which contains the ordered data items based on a time.
- $Q(t-1)$ is the queue at the previous time step.
- d_{new} is the newly accessed data item.

V. RESULTS

Fig 1 : Cloud Service Provider Login

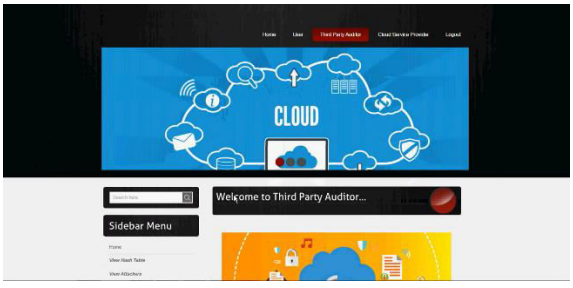


Fig 2 : Third Party Auditor

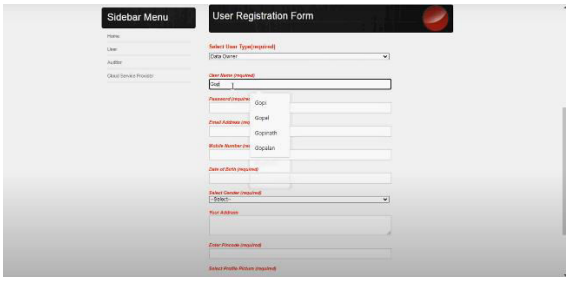


Fig 3 : Registration from

Fig 4 : View Time Delay Results

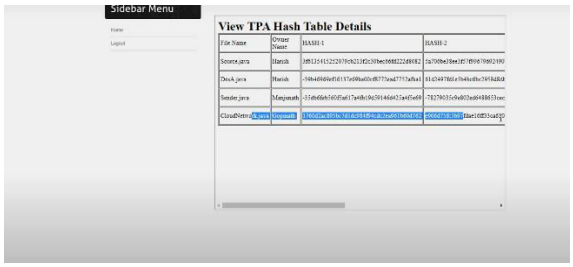


Fig 5 : View TPA Hash Table Details

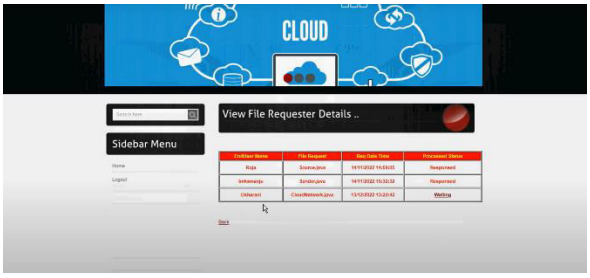


Fig 6 : View File Requester Details

VI. CONCLUSION

In this paper, we introduce an efficient, secure, and privacy-preserving mobile cloud storage (MCS) solution. Our proposed scheme is designed to protect both the data and the

access patterns of users, ensuring that sensitive information remains confidential. Unlike existing solutions, our approach stands out due to its smaller item size, minimal client-side computation, and constant communication overhead, which are key advantages for mobile environments where resources are limited.

One of the unique aspects of our scheme is its consideration of **temporal locality**, which optimizes performance by reducing redundant data retrievals. By utilizing this feature, we can further enhance the overall efficiency of the system. Additionally, we incorporate a verification method that ensures the integrity of the system and helps guard against malicious actions by the cloud provider, ensuring trustworthiness.

At the heart of our MCS scheme is an innovative **Oblivious Selection and Update (OSU) protocol**, which allows the client to securely select and update its encrypted data items stored in the cloud, all while maintaining privacy. This process is facilitated through the use of a small encrypted vector, significantly reducing both client-side computational effort and the required communication with the cloud. We believe that this OSU protocol is a powerful building block, not only for our MCS solution but also for other secure multi-party computation scenarios, where privacy and security are paramount.

Through rigorous security and privacy proofs, we demonstrate that our scheme offers robust data confidentiality and strong privacy guarantees. To validate its effectiveness, we compare our approach against two existing oblivious storage schemes, running extensive simulations. The results show that our scheme outperforms the alternatives, achieving greater efficiency and superior overall performance, making it a promising solution for real-world mobile cloud storage applications.

In conclusion, our work presents a highly efficient and secure mobile cloud storage framework that addresses key privacy concerns and operational challenges. By incorporating innovative protocols and optimizations, we provide a practical and scalable solution for modern mobile cloud environments.

REFERENCES

[1] E. Blass, T. Mayberry, G. Noubir, and K. Onarlioglu, “Toward robust hidden volumes using write-only oblivious RAM,” 2014,

[2] D. S. Roche, A. J. Aviv, S. G. Choi, and T. Mayberry,“Deterministic, stash-free write-only ORAM,” in Proceedings of the 2017,

[3] E. Stefanov and E. Shi, “Oblivistore: High performance oblivious cloud storage,” 2013.

[4] D. Cash, A. Kupc¸ ¨ u, and D. Wichs, “Dynamic proofs of retrievability via oblivious RAM,” in Advances in Cryptology-EUROCRYPT 2013.

[5] E. Stefanov and E. Shi, “Multi-cloud oblivious storage,” in 2013

[6] B. Carbunar and R. Sion, “Write-once read-many oblivious RAM,”2011.

[7] X. S. Wang, Y. Huang, T. H. Chan, A. Shelat, and E. Shi, “SCORAM:
oblivious RAM for secure computation,” 2014

[8] E. Boyle, K. Chung, and R. Pass, “Large-scale secure computation:Multi-party computation for (parallel) RAM programs,” in Advances in Cryptology - CRYPTO 2015

[9] C. Gentry, K. A. Goldman, S. Halevi, C. S. Jutla, M. Raykova,and D. Wichs, "Optimizing ORAM and using it efficiently for secure computation," in Privacy Enhancing Technologies 2013.
[10] S. Lu and R. Ostrovsky, "Distributed oblivious RAM for secure two-party computation," 2013
[11] S. Zahur, X. Wang, M. Raykova, A. Gascon, J. Doerner, D. Evans, and J. Katz, "Revisiting square-root ORAM: efficient random access in multi-party computation," 2016
[12] D. Apon, J. Katz, E. Shi, and A. Thiruvengadam, "Verifiable oblivious storage," in Public-Key Cryptography - 2014
[13] S. Devadas, M. van Dijk, C. W. Fletcher, L. Ren, E. Shi, and D. Wichs, "Onion ORAM: A constant bandwidth blowup
oblivious RAM," in Theory of Cryptography - 2016.
[14] A. Chakraborti, A. J. Aviv, S. G. Choi, T. Mayberry, D. S. Roche, and R. Sion, "roram: Efficient range ORAM with o(log2 N) locality,2019
[15] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification,
attack and mitigation," 2012.
[16] J. Kilian, "Founding cryptography on oblivious transfer," in Proceedings of the 20th Annual ACM Symposium on Theory of
Computing, 1988
[17] D. Boneh, D. Mazieres, and R. A. Popa, "Remote oblivious storage: Making oblivious ram practical,2011.
[18] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious rams," J. ACM, 1996.
[19] J. Camenisch, M. Kohlweiss, A. Rial, and C. Sheedy, "Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data," 2009.
[20] T. Hoang, A. A. Yavuz, F. B. Durak, and J. Guajardo, "Oblivious dynamic searchable encryption via distributed PIR and ORAM," 2017
[21] S. Garg, P. Mohassel, and C. Papamanthou, "TWORAM: efficient oblivious RAM in two rounds with applications to searchable encryption," in Advances in Cryptology -2016