PEO-Store: Practical and Economical Oblivious Store with Peer-to-Peer Delegation

Wenlong Tian[1], Jian Guo[2], Zhiyong Xu[3], Ruixuan Li[4] and Weijun Xiao[5]

[1]School of Computer Science and Technology, University of South China, China,
[2]School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
[3]Math and Computer Science Department, Suffolk University, USA
[4]School of Computer Science and Technology, Huazhong University of Science and Technology, China
[5]Electrical and Computer Engineering, Virginia Commonwealth University, USA

Abstract. The growing popularity of cloud storage has brought attention to critical need for preventing information leakage from cloud access patterns. To this end, recent efforts have extended Oblivious RAM (ORAM) to the cloud environment in the form of Oblivious Store. However, its impracticality due to the use of probability encryption with fake accesses to obfuscate the access pattern, as well as the security requirements of conventional obliviousness designs, which hinder cloud interests in improving storage utilization by removing redundant data among cross-users, limit its effectiveness. Thus, we propose a practical Oblivious Store, PEO-Store, which integrates the obliviousness property into the cloud while removing redundancy without compromising security. Unlike conventional schemes, PEO-Store randomly selects a delegate for each client to communicate with the cloud, breaking the mapping link between a valid access pattern sequence and a specific client. Each client encrypts their data and shares it with selected delegates, who act as intermediaries with the cloud provider. This design leverages non-interactive zero-knowledge-based redundancy detection, discrete logarithm problem-based key sharing, and secure time-based delivery proof to protect access pattern privacy and accurately identify and remove redundancy in the cloud. The theoretical proof demonstrates that the probability of identifying the valid access pattern with a specific user is negligible in our design. Experimental results show that PEO-Store outperforms state-of-the-art methods, achieving an average throughput of up to 3 times faster and saving 74% of storage space.

Keywords: Oblivious Store · Delegation · Zero-Knowledge Proof · Secure Deduplication

Introduction

As cloud storage continues to prosper, privacy protection becomes an increasingly pressing concern for users and organizations who seek to outsource sensitive data for the sake of flexibility and reliability. However, malicious attackers can exploit remote access patterns to steal sensitive information, as demonstrated in [IKK12]. For instance, the frequency of a patient's medication database searches may inadvertently reveal the patient's disease. Existing solutions to address this security issue aim to obfuscate the valid user's access pattern by incorporating fake access sequences, such as Oblivious RAM (ORAM) and Private Information Retrieval (PIR). In theory, these approaches render the processed access pattern computationally indistinguishable from a random sequence of bit strings, thereby preventing malicious attackers from stealing privacy based on access patterns.

However, existing solutions such as Private Information Retrieval (PIR) fail to provide protection for writing requests, allowing users to retrieve data from a server without revealing which item was retrieved [CHK22, GSW21]. Oblivious RAM (ORAM), initially proposed by Golden Rich [Gol87], offers protection against privacy leakage from both read and write operations. However, approaches such as Path ORAM [SvDS+13] incur heavy computational overheads and additional unrelated block transmissions. Although researchers have attempted to simplify the algorithm [SvDS+13], compress real and fake data to decrease network bandwidth [CCR19], or introduce Intel SGX [SGF18, RRM20] to improve ORAM performance, its practical usage remains challenging, especially at the storage level where it is referred to as Oblivious Store.

There is a rich line to construct oblivious stores in research communities [BNP+15, CS19, CBC+18, RFK+15, TLXX22, WST12, SS13], which aim to protect data privacy against persistent adversaries. Some of these approaches focus on reducing latency [SvDS+13, MPC+18a, RFK+15], while others aim to scale up to handle large data volumes through parallelism or amortization [LPM+13, AKL+22, DFD+21]. To achieve a balance between security and high performance, some researchers have combined Private Information Retrieval (PIR) to protect read access patterns and write-only Oblivious RAM (ORAM) to safeguard write access patterns [JR15, SS13]. These techniques offer powerful oblivious data access guarantees with minimal bandwidth overhead [GKL+20a]. Additionally, introducing a proxy server has been proposed as a means of enhancing the robustness and performance of the oblivious store [JMTS16, GKL+20b, VBKA22]. In summary, these techniques provide a robust and secure means of accessing data without sacrificing performance.

Despite the benefits offered by existing oblivious stores, their adoption in cloud scenarios is still hindered by the associated costs. One reason for this is the unacceptable performance caused by fake accessing. Furthermore, special security requirements necessary for deploying these stores in the cloud are often overlooked, as they require a significant transformation cost. Another issue is that these oblivious designs seriously compromise cloud deduplication function [PKK+22, YXT+22, MJWT13], which is essential for removing duplicate data among cross-users. The benefits of deduplication, such as increasing cloud storage utility and saving network bandwidth, have already been widely accepted and deployed in many cloud storage products [PP14]. Although some secure deduplication schemes [YLL22, TMPD19] have been proposed to remove redundancies without leaking data content to other malicious attackers and cloud storage providers, they still disclose user privacy from access patterns, even when the data is encrypted. Naively applying existing ORAM-related work to cloud storage poses additional challenges, as storage providers are unable to detect duplicate data. This is because the ORAM-like protection results in duplicate data being stored in different locations with varying ciphertext, making it only distinguishable by the client.

The inherent contradiction between maintaining privacy by resisting data access pattern leakage and efficiently removing redundant data from cloud storage presents a significant challenge. This dilemma is exacerbated when we mix fake requests with probabilistic encryption, leading to confusion as traditional ORAM-like protection and rendering it impossible to recognize valid duplicate data across multiple users. In contrast, it is essential for the cloud to accurately identify redundant data to improve storage utilization. Consequently, we must re-consider how we can practically incorporate the oblivious property into existing cloud storage architecture without compromising security. The resulting design should not impede the benefits of the cloud storage provider by removing redundancy with a negligible transformation cost. Notably, we are the first to consider a secure approach to simultaneously address privacy leakage from access patterns and securely remove redundant data among cross-users in cloud storage.

In summary, a sufficient condition to obtain user privacy from access patterns, based on our observation, is to figure out the mapping link between a valid access pattern and the

right person or organization. Tradition ORAM-like protection is to mapping a invalid access pattern to the right person or organization by make the access pattern computationally indistinguishable with a random sequence of bit strings. Obviously, breaking these mapping can also protect the privacy from access pattern leakage. Motivated by this, we propose a practical and economically oblivious store, PEO-Store. It can easily break these links and remove the redundancy in an oblivious way. Specifically, all clients are grouped as a p2p network. Each node has two roles: client and delegate. Each client selects several delegates and assigns each delegate's uploading or downloading tasks. The cloud only interacts with the delegates. From the cloud aspect, the data ownership belongs to the delegate who uploaded it. Each data are encrypted by the client before sending it to the delegate and shared among each actual data owner, which is common in cloud sharing scenario. By leveraging the zero-knowledge and discrete logarithm problem techniques, our method can practically and economically relieve the contradiction between obliviousness and accurately identify and remove the redundancy in the cloud with a trivial modification of existing clouds. Our contributions are summarized as follows:

- Firstly, the use of cloud storage has raised concerns about protecting the privacy of users' data while ensuring efficient storage utilization. We observed conventional approaches to achieving obliviousness have been successful in preventing privacy breaches through access pattern analysis but have hindered the removal of redundant data across multiple users. This contradiction highlights the need to reconsider how obliviousness can be achieved without compromising the secure removal of duplicate data. To address this issue, we propose three key challenges that must be tackled to achieve an effective balance between protecting privacy and optimizing storage utilization in the cloud scenario.

- Secondly, we introduce a novel practical and cost-effective oblivious store, referred to as PEO-Store, which addresses the inherent conflict between maintaining privacy and eliminating redundant data in cloud storage systems. Unlike conventional approaches to achieving obliviousness, our method leverages a delegation mechanism to block the mapping link between the access sequence and specific users. Specifically, the cloud interacts only with authorized delegates and not with clients directly. Each data are encrypted by the client before sending it to the delegate. To maintain data ownership and security, the data ownership belongs to the delegate who uploaded it. Moreover, we propose a secure randomized delegate selection scheme, a non-interactive zero-knowledge based redundancy detection scheme, a discrete logarithm problem-based key sharing scheme, and a secure time-based delivery proof mechanism to ensure privacy preservation and data redundancy elimination in PEO-Store. By practically resolving the contradiction between resisting privacy leakage from access patterns and removing duplicate data in the cloud scenario, our method contributes to the improvement of cloud storage performance and security.

- Thirdly, we formalize the security definition of PEO-Store and provide a proof of its robustness against unauthorized access to users' data. Specifically, we show that the probability of an attacker identifying the valid access pattern associated with a particular user is negligible. Additionally, we demonstrate that our construction can securely and accurately eliminate data redundancy without compromising data contents or encryption keys. To evaluate the effectiveness of PEO-Store, we conduct extensive simulations on real-world workloads. The results demonstrate that PEO-Store outperforms existing state-of-the-art solutions for oblivious data storage, achieving nearly 3× throughput on average and saving up to 74% of storage space. Our findings indicate that PEO-Store is a practical and efficient solution for cloud storage that successfully balances the protection of privacy and the optimization of storage utilization.

The rest of the paper is organized as follows. The background of our work is summarized in Section 1. In Section 2, we re-visit the protection in access patterns and outline the challenges to defend the privacy leakage in access patterns while securely removing the redundancy with a trivial modification in cloud storage services. Then, we propose a practical and economically oblivious store in Section 3. Finally, the theoretical proof and extended experiments are presented in Section 4 and Section 5, respectively.

1 Background

We provide a brief overview of the foundational concepts of Oblivious RAM, Oblivious Store, and secure deduplication as our preliminary discussion. Then, the limitations of naively combining the obliviousness property with secure data redundancy removal in cloud storage are elaborated.

1.1 Oblivious RAM & Oblivious Store

To prevent sensitive information leakage from access patterns, Oblivious RAM (ORAM) transforms each access into an oblivious access that is computationally indistinguishable from a random request sequence. Path ORAM, one of the most widely used ORAM implementations, employs a complete binary tree structure to store data blocks. Each node in the tree is a bucket that can accommodate Z blocks, where at most Z blocks are actual data blocks, while the others are dummy blocks. Each block is encrypted using probabilistic encryption. The client maintains a mapping table that associates each data block with a leaf ID. During an oblivious access, all blocks from the target leaf ID up to the tree root are fetched into a client-side stash, a small memory buffer. Only the actual data blocks are decrypted and held in the stash. The client then randomly assigns a new leaf ID for the target block, refreshes the mapping table, and re-encrypts the block. Several blocks in the stash are randomly selected and written back to their original path after re-encryption.

When considering the integration of Oblivious RAM (ORAM) into cloud storage, which is referred to as an oblivious store, there exist three main architectural approaches: hardware enclave-based ORAM, ORAM with a Trust Proxy, and load balancers with subORAM. Hardware enclaves, such as Intel SGX, have been utilized to deploy and maintain the public state of an oblivious process on a public cloud, while also supporting multiple clients without a central point of attack [SGF18, ZDB+17a, EZ17]. However, the problem of concealing access patterns for oblivious storage remains unsolved as enclave side channels can be exploited by attackers to extract enclave secrets through data-dependent memory accesses [BMD+17, VBWK+17]. Alternatively, the Trust Proxy-based ORAM architecture allows for batching and parallel calculation to improve throughput. However, it is not suitable for deployment in an untrusted cloud environment, and it is susceptible to a central point of attack in the system. The third approach involves building an oblivious load balancer to guarantee batching of requests without any information leakage, along with designing a high-throughput subORAM [DFD+21] for processing the batches in a single linear scan. This architecture provides an effective solution for the problem of concealing access patterns for oblivious storage while maintaining a high throughput.

1.2 Secure Deduplication

Secure Deduplication is a method of securely removing duplicate data in a cloud environment that has gained significant attention from both cryptography and system communities [SKH17]. In this approach, each uploading operation splits the data into small blocks using a chunking algorithm and performs encryption on the client-side. To maintain the

identical content after encryption, each symmetric encryption key is derived from the content of each chunk, referred to as message-locked encryption. Only unique blocks are stored in the cloud storage, as duplicate blocks are encrypted by the same key. Numerous research studies have been conducted to enhance the performance of secure deduplication [BKR13b, LAP15, BKR13a, TLXX18]. These studies have explored various methods such as importing a trusted third party, simplifying key management, improving the secure deduplication ratio, and utilizing the Intel SGX hard enclave. However, the issue of resisting privacy leakage from access patterns in secure deduplication remains largely unexplored. Malicious users can collude with cloud storage and obtain users' valid access patterns, thereby jeopardizing their privacy. Therefore, it is imperative to investigate and address this challenge to enhance the security and privacy of secure deduplication.

1.3 Limitations of Strawman Approaches

we further describe the subtle limitations of naively removing redundancy using existing oblivious store architectures. As previously discussed, the primary purpose of the oblivious store is to obfuscate users' valid access sequences by adding fake blocks, probability encryption, and shuffling the data. This approach prevents malicious attackers from identifying whether a block has been previously requested or identifying duplicate data. However, there are two naive ways to detect duplicate data in oblivious storage. The first method involves leveraging trusted execution environments (TEEs) such as Intel SGX, while the second method is based on the proxy-based oblivious store. Both methods suffer from significant limitations.

Several privacy-preserving ORAM frameworks, such as Ohrimenko et al.'s [OSF+16], Mishra et al.'s [MPC+18b], and Zheng et al.'s [ZDB+17b], have been proposed that use trusted hardware enclaves like Intel SGX to provide secure computation. These frameworks employ compaction and shuffling as core primitives, but are still vulnerable to side-channel attacks on TEEs, as demonstrated by works like Xu et al.'s [XCP15], Liu et al.'s [LYG+15], and Van Bulck et al.'s [VBMW+18]. In particular, these attacks allow adversaries to distinguish real blocks from dummy ones in the ORAM array and infer the final permutation results of a shuffle in the ORAM workflow. To address these security issues, Sasy et al. [SJG22] propose a fully oblivious algorithm for compaction and shuffling in TEEs that is independent of secret keys. However, this approach incurs significant performance overhead, with a complexity of $\mathcal{O}(n \log^2 n)$ in the shuffling process and worsened performance at the data storage level.

When considering secure deduplication in an oblivious store with TEEs, three major obstacles arise. First, the use of encryption in a fully oblivious setting makes the control flow oblivious, meaning that the server cannot recognize duplicate data that is dependent on secret data. Second, secure deduplication requires detection of redundant blocks with different sizes, while the ORAM design requires each block to have the same size in a bucket, leading to poor storage utilization. Finally, hardware enclave designs cannot scale securely across machines, and frequent access to redundancy in this kind of oblivious store will dramatically increase shuffling and concurrency among users.

The other approach to implementing an oblivious store is to utilize a trusted proxy server, as proposed by Bindschaedler et al. [BNP+15], Sahin et al. [SZA+16], and Vuppalapati et al. [VBKA22]. In this approach, a single or distributed proxy server acts as an intermediary between clients and the cloud, communicating with the cloud on behalf of clients in an oblivious way while clients request data through a secure channel such as TLS. The trusted proxy maintains the mapping of users' blocks and logical addresses in the oblivious store and can easily remove redundancy based on plaintext blocks before the oblivious process. This approach significantly increases concurrency and throughput for multi-user data management scenarios. However, the use of additional trusted independent servers is a strong assumption that is difficult to meet in commercial contexts, which can

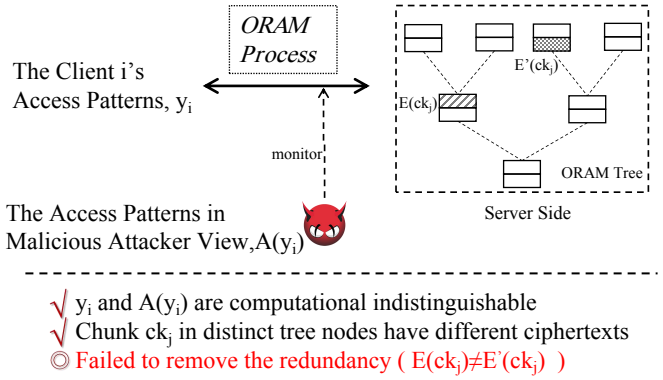

Figure 1: Oblivious Store with Path ORAM failed to remove the redundancy

become a scalability bottleneck.

Furthermore, arranging for trusted third parties remains an unsolved problem [ZLK11]. As long as there are motives of greed, politics, and revenge, those who perform (or supervise) work done by such an entity will provide potential loopholes through which necessary trust may leak. In many environments, the strength of trust is as weak as its weakest link. When the infrastructure of a trusted CA is breached, the whole chain of trust is broken. The 2011 incident at CA DigiNotar [Wik22b] exemplifies the weaknesses of the system and its effects. As Bruce Schneier has pointed out, after the 2013 mass surveillance disclosures, no third party should be trusted [Sch13]. Thus, the use of a trusted third party in an oblivious store system is not without risk, and alternative approaches must be explored to ensure the security and privacy of user data.

2 Challenges & Threat Model

Drawing from the previous analysis, existing oblivious stores have been designed to protect against access pattern leakage by adding fake requests and obscuring the exposed access patterns in a manner that is computationally indistinguishable from randomized request sequences. However, as a semi-honest entity, the cloud is unable to differentiate between the current request block and the previous one due to the definition of the existing obliviousness property. This not only undermines the performance and storage efficiency of the cloud, but also compromises its interests. While hardware enclave or trusted proxy architectures provide some relief to the problem, the goal of obliviousness comes at the cost of removing redundancy and being impractical for usage. Therefore, we propose a re-visit privacy protection in access patterns and identify three challenges in defending against privacy leakage in access patterns while still securely removing redundancy with a trivial modification in cloud storage services.

Challenge #1: Compared with traditional oblivious stores, is there any other more efficient, practical way to resist privacy leakage from access patterns in the cloud scenario?

Our observations have led us to identify two necessary preconditions for achieving privacy from access patterns in cloud environments. The first precondition is to determine the specific user who initiated the access sequence, while the second is to ascertain the exact access sequence for that user. Conventional methods for achieving privacy from access patterns involve confusing the access patterns to prevent malicious attackers from deducing the precise access pattern sequences of the user. This is accomplished by generating randomized strings that are computationally indistinguishable from the user's request sequences. To achieve this, fake requests must be periodically embedded into the valid

request sequences, and the data must be re-encrypted to confuse attackers. However, these conventional methods come at the cost of degraded performance and diminished quality of cloud storage services.

We propose a novel approach that focuses on breaking the mapping from valid access sequences with specific users to resist privacy leakage from access patterns. This approach differs from previous work, which solely relied on confusing the access patterns. By assuming that the first precondition is violated, we prevent a malicious attacker from achieving privacy from access patterns due to the lack of valid access pattern sequences of that user. This approach provides an effective means of safeguarding sensitive data in cloud-based systems, without compromising on the performance or quality of cloud storage services. Further research is necessary to evaluate the efficacy of this approach and to identify other potential solutions for mitigating privacy risks associated with access pattern disclosure.

Challenge #2: How to securely remove the redundancy while keeping the oblivious property?

The removal of redundancy from data in cloud storage while maintaining the obliviousness property presents a significant challenge. Securely removing redundancy requires detecting duplicate data without disclosing its content, while obliviousness requires that no one, except the client, can differentiate between the same data accessed at different times. While combining the trusted proxy architecture can alleviate this paradox, it is not advisable to rely on third-party trust, as Bruce Schneier has pointed out. Additionally, it is challenging to justify the business case for an independent party to run an additional server solely for improving privacy in cloud storage services.

To address this challenge, we propose a novel approach in this paper that groups all clients into a peer-to-peer network. Each node in the network has a delegation role, and clients can randomly delegate their access tasks to other nodes. This approach enables us to leverage zero-knowledge proof and secure multi-party computation techniques, which are detailed in Section 3, to overcome the challenges associated with redundancy removal and obliviousness preservation. By adopting this approach, we can improve the privacy and security of cloud storage services without relying on third-party trust or compromising on performance. Further research is necessary to evaluate the effectiveness of this approach and to identify other potential solutions for addressing privacy risks associated with redundancy and obliviousness in cloud-based systems.

Challenge #3: How to be compatible with the existing cloud storage architectures when alleviating the paradox of securely removing redundancy and keeping obliviousness property.

Cloud businesses are focused on maximizing profits by providing high-quality products with optimal performance. They typically prefer simple and scalable methods with the same level of security complexity. However, existing designs in the oblivious paradigm have not adequately considered compatibility factors and are difficult to integrate into existing cloud storage applications. Furthermore, these designs require cloud storage providers to modify their current data management practices with probabilistic encryption to achieve obliviousness, further complicating the integration process.

To address this issue, our design is more compatible to the existing cloud storage services with a trivial modifications by alleviating the paradox of securely removing redundancy while preserving the obliviousness property, which is a critical challenge in designing an oblivious store for cloud scenarios. By addressing this challenge, we can simplify the integration process for cloud storage providers and enable them to adopt oblivious store designs without significant modifications to their existing data management practices. Our proposed approach can also enhance the overall performance and scalability of cloud storage systems, thereby providing a win-win solution for cloud businesses and their customers. Further research is necessary to evaluate the efficacy of this approach and

to identify other potential solutions for improving compatibility in the design of oblivious stores for cloud-based systems.

Threat Model Although traditional oblivious stores provide data confidentiality and conceal the data access pattern, they significantly affect the profits of cloud storage providers by removing duplicate data and are impractical due to performance degradation. As our previous analysis indicates, the precondition for privacy leakage from access patterns is mapping the request sequences to a specific client. Without mapping, malicious attackers can not determine personal privacy. For instance, the frequency of a specific medicine at a certain time in the hospital database does not reveal the disease until mapped to a specific individual. Therefore, our goal is to practically break this mapping and securely remove duplicate data with minimal modification to existing cloud storage services, while also preserving performance and profitability.

In our threat model, we assume that the cloud server is honest-but-curious, and all clients are grouped as a peer-to-peer network. We also assume that the number of malicious nodes does not exceed half of the total network nodes, which is reasonable, similar to the Bitcoin system. Malicious attackers and cloud servers may collude to compromise the privacy of targeted individuals. In this model, each node has two roles: delegate and client. Delegates directly upload or download data to and from the cloud server, while clients communicate with other delegates for data requests. Our proposed approach provides an efficient and practical solution for addressing the privacy challenges in cloud storage while ensuring compatibility with existing cloud storage services.

For example, the client selects a number of delegates and randomly assigns each delegate with either a data uploading or downloading task. The cloud server only interacts with these delegates. From the cloud perspective, data ownership belongs to the delegate who uploaded it. To maintain confidentiality, each piece of data is encrypted by the user before it is sent to the delegate and shared among the actual data owners, which is a common practice in cloud-sharing scenarios. Only the node with the decryption key can access the data. Our proposed construction does not consider data transaction fees, as this can be easily implemented through online anonymous electrical platforms for each successful transaction, such as Zcash [BFV19].

3 PEO-Store Design

In this paper, we provide an overview of PEO-Store. Next, we introduce two schemes designed to securely prevent repeated uploading of duplicate data in an oblivious manner: a secure randomized delegate selection scheme and a non-interactive zero-knowledge based redundancy detection scheme. Finally, we elaborate on a key-sharing approach based on the discrete logarithm problem and a secure time-based delivery proof.

3.1 Design Overview

PEO-Store employs a common architecture to effectively prevent privacy leaks and securely eliminate redundancy with minimal modification to existing cloud storage services, as illustrated in Figure 2. Each peer node serves in two capacities, as a client and delegate, and all peer nodes form a peer-to-peer (p2p) network. In the delegate role, the peer node communicates directly with the public cloud service. The identifier for each peer node is a consistent hash of its IP address and public key. Additionally, the cloud follows the traditional cloud architecture with minor adjustments. All communication in PEO-Store is transmitted via an SSL-encrypted link, ensuring that no one can monitor the entire p2p network's communication. The delegate role prevents malicious attackers from determining the mapping from access patterns to the valid client, without the need to add any fake

requests. We then outline the PEO-Store workflow, which is based on two major operations: write and read.

PEO-Store's writing process involves several steps. Firstly, the client splits the data into several chunks, with each chunk denoted as ck_j. Before writing ck_j, the client randomly selects k delegates based on the secure randomized delegate selection scheme, which is explained in subsection 3.2. Then, ck_j is accompanied by two metadata, $metad_j$ and $metas_j$. $metad_j$ is used for non-interactive zero-knowledge based redundancy detection by the delegates, while $metas_j$ is used for key sharing in the case of a duplicate chunk. The details of these processes can be found in subsections 3.3 and 3.4. The detecting result is sent back to the client, and if the results from k delegates are the same, the ciphertext of ck_j and $metad_j$ are temporarily stored in the p2p network and the cloud, respectively. The delegate and the cloud then calculate delivery proofs and send them back to the client through the delegates. After validating the proofs, $metad_j$, $metas_j$, and the ciphertext of ck_j are permanently stored.

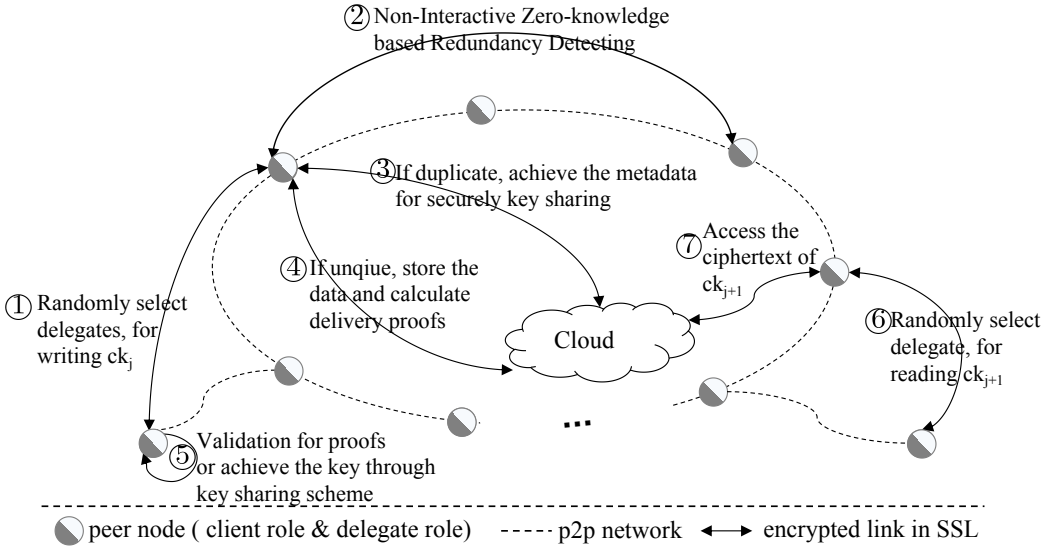

Figure 2: The Overview of PEO-Store

If ck_j is a duplicate, the delegate fetches the $metas_j$ of ck_j from the cloud and sends it to the client, who can then obtain the encryption key from $metas_j$ using the discrete logarithm problem-based key-sharing scheme. Only the client who knows the content of ck_j can get the encryption key. If ck_j is unique, the client assigns a specific time to the delegate, and each delegate and the cloud must forge the delivery proofs before the predefined time, as explained in subsection 3.5. In the case of a malicious node pretending to return invalid detecting information, the client initiates an additional delegate selection processing round and adds the malicious node to its blacklist. The client can also pretend to be a delegate in each round. When reading a chunk ck_{j+1}, the client randomly selects one delegate to fetch ck_{j+1}. If the delegate returns an invalid ciphertext of ck_{j+1}, the client randomly asks other delegates to fetch it again and puts the previous delegate on its blacklist.

For the reading process, the client randomly selects k delegates based on the secure randomized delegate selection scheme and the delegate fetches the ciphertext of ck_{j+1} and sends it back to the client, who can then decrypt it using the corresponding encryption key. It is important to note that PEO-Store uses the chord [SMK+01] as its peer-to-peer distributed hash table due to its robust scalability and rapid lookup function. For example,

to execute lookup queries for m entries in distributed hash table result in $\mathcal{O}(m \log N)$ in N-node p2p network. Each node can join and leave the p2p network without disruption resulting in $\mathcal{O}(\log^2 N)$ messages to re-establish the distributed hash table. Additionally, discrete logarithm problem is used to construct the metadata stored in the p2p network and the cloud, making it highly secure.

3.2 Secure Randomized Delegate Selection Scheme

The selection of delegates is a critical responsibility that involves randomly choosing delegate nodes for each access, including the possibility of the client being selected as a delegate node. This process is essential to ensure a robust and secure system that can effectively detect duplicates and protect against potential security threats. Based on our thread model, each node communicates with the cloud in a delegate role to break the mapping link between a valid access pattern and the client. This communication enhances the security of the system by obscuring the access pattern of the client. Before uploading data, the client splits it into smaller chunks, reducing the impact of potential attacks that target specific chunks. Each chunk is randomly assigned to k delegates for the duplicate-detecting process, contributing to the overall reliability of the system. By selecting delegates randomly, the risk of bias is minimized, maintaining fairness in the delegation process. The detailed pseudocode of the secure randomized delegate selection scheme is shown as Algorithm 1.

To efficiently select delegate nodes, PEO-Store leverages the chord protocol [SMK+01], one of the most widely used distributed hash table (DHT) protocols. Each peer node in the system is assigned a consistent hash value [Wik22a], which is an m-bit identifier derived from the node's IP address and public key. This consistent hashing approach ensures that the nodes are uniformly distributed throughout the network, allowing nodes to join or leave the system without disrupting its operation. In PEO-Store, metadata required for duplicate detection is stored using key-value pairs in a DHT. The keys are assigned to nodes in the network, and the Chord protocol's lookup function, denoted as *lookups* for simplicity, is used to discover the location of the metadata.

Algorithm 1 Secure Randomized Delegate Selection Scheme

Require: n: chunk number to be operated; k: delegate number; *blacklist*: a list of delegates that are regarded as untrustworthy
Ensure: IP addresses queue, L

```
for i=0 to n do
    count ← 0
    while true do
        generate a random number→ rand_num
        hash← consistent_hash(rand_num)
        delegate's IP ← lookups(hash)
        if rand_num%2==0 & delegate's IP ∉ blacklist then
            L.push( delegate's IP )
            count++
            if count==k then
                break
            end if
        end if
    end while
end for
return L
```

For each delegate selection request, the client generates a random number, $rand_num$,

and calculates its consistent hash value. The client then uses the *lookups* function to identify the closest delegate IP address in the P2P network. If $rand_num$ is even and the delegated IP is not included in the *blacklist*, the client adds the delegate's IP to the IP address list, L. For each chunk, PEO-Store selects k delegates for redundancy detection by repeating this process. The selection scheme aims to confuse the chunk access sequence by using different delegates for redundancy detection, making it difficult for malicious delegates to identify other selected delegate identifiers. Ultimately, the chunks are processed in the order specified by the IP address list, L. The use of random selection, combined with the consistent hash values assigned to each node, allows the system to operate in a secure and reliable manner, even in the face of malicious attacks.

3.3 Non-Interactive Zero-Knowledge based Redundancy Detecting Scheme

Motivated by the challenges mentioned in our thread model, we propose a novel scheme for redundancy detection in PEO-Store based on non-interactive zero-knowledge (NIZK) proofs. Specifically, when a client initiates a request to check for duplicate chunks, randomly selected peer nodes act as verifiers, while the provers are the nodes that store relevant metadata in the distributed hash table. By using this scheme, PEO-Store can establish the knowledge of whether a given chunk, ck_j, is a duplicate or not without revealing any information beyond the validity of the zero-knowledge proof itself. This approach not only enhances privacy and security in the P2P network but also enables efficient and scalable redundancy detection.

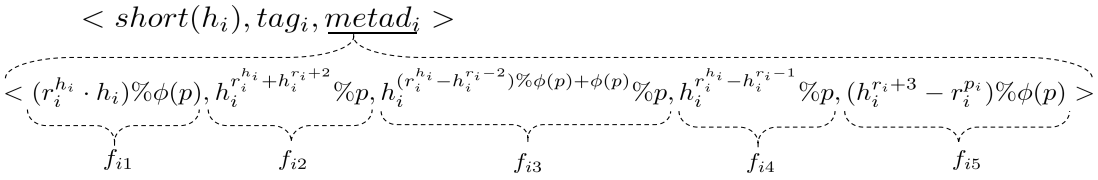

Figure 3: The Metadata Structure for Non-Interactive Zero-Knowledge based Redundancy Detecting Scheme

Formerly speaking, let $\mathbb{F}_p$ be a finite field where p is a large prime number. For each chunk ck_i, the client generates a random number r_i and constructs a three-tuple $< short(h_i), tag_i, metad_i >$ as shown in Figure 3. Here, h_i denotes the SHA-1 hash value of ck_i's content, and ϕ represents Euler's totient function [IRR90]. $short(h_i)$ is the short hash of h_i, containing only the first half of the value. The $metad_i$ is constructed using discrete logarithm problem and consists of five parts. Finally, the tag_i is the hash value of $< h_i, metad_i >$. The non-interactive zero-knowledge based redundancy detecting scheme (zkRD) includes the following steps:

- pp $\leftarrow$ zkRD.$\mathcal{G}(1^\lambda)$: given the security parameter, generate the public parameter pp, including the prime number p.
- $< short(h_i),\ tag_i,\ metad_i >\leftarrow$zkRD.$Commit(ck_i, r_i, p)$: Client generates random number r_i, computes $< short(h_i)\ ,\ tag_i\ ,\ metad_i >$ for the chunk ck_i, and sends the $< short(h_i),\ tag_i,\ metad_i >$ to randomly selected delegate nodes.
- $\mathbf{S}\leftarrow$zkRD.$\mathcal{P}(short(h_i),\ tag_i,\ metad_i, r_i,\ p)$: Delegate node looks up the related sets $\mathbf{S}$ in p2p network where $\mathbf{S} = \{< short(h_j), tag_j, metad_j > \mid short(h_j)$ equals to $short(h_i)\}$
- $\{0,1\} \leftarrow$ zkRD.$\mathcal{V}(short(h_i),\ tag_i,\ metad_i,\ \mathbf{S})$and verifies the proof $< \pi_1, \pi_2 >$ based on Formula 1. If the verification result is not equal to 1 or the $\mathbf{S}$ is empty, it

proves that the ck_i is a unique chunk, and vice versa. For each unique chunk, the $< short(h_i), tag_i, metad_i >$ will be temporary stored in the distributed hash table. It will permanently stored in the p2p network after the validation of secure time-based delegate delivery proof. For duplicate chunk, the delegate fetches the metadata $metas_j$ for securely key sharing from the cloud and send it to the clients.

We say that zkRD is a non-interactive zero-knowledge based redundancy detecting if the following holds:

- **Completeness** For any h_i, pp $\leftarrow$ zkRD.$\mathcal{G}(1^\lambda)$, $<short(h_i)$, tag_i, $metad_i>\leftarrow$ zkRD.$Commit(h_i, r_i, pp)$, $\mathbf{S}\leftarrow$zkRD.$\mathcal{P}(short(h_i)$, tag_i, $metad_i$,r_i, $pp)$, and where exists a $< short(h_j)$, tag_j, $metad_j >$ in $\mathbf{S}$ that h_i equals to h_j,it holds that

$$Pr[zkRD.\mathcal{V}(short(h_i), tag_i, metad_i, \mathbf{S}) =< 1, 1 >] = 1$$

- **Soundness** For any PPT adversary $\mathfrak{A}$, pp $\leftarrow$ zkRD.$\mathcal{G}(1^\lambda)$, $<short(h_i^*)$, tag_i^*, $metad_i^*>$ $\leftarrow$ zkRD.$Commit(h_i^*, r_i^*, pp)$ but the h_i^* is not equals to h_i, the following probability is negligible in λ:

$$Pr[zkRD.\mathcal{V}(short(h_i^*), tag_i^*, metad_i^*, \mathbf{S}) = 1] \leq negl(\lambda)$$

- **Zero-Knowledge**: pp $\leftarrow$ zkRD.$\mathcal{G}(1^\lambda)$, for any expected polynomial-time algorithm $\mathfrak{B}$, which can produce, upon input of the assertions to be proven - but without interacting with the real prover: let and $com_{zk} = zkRD.Commit(h_i, r_i, pp)$ and $com'_{zk} = Sim(x, r_i, pp)$

$$|Pr[\mathfrak{B}(com_{zk}, pp) = 1] - Pr[\mathfrak{B}(com'_{zk}, pp) = 1]| \leq negl(\lambda)$$

$$\begin{aligned}
\pi_1 = (((((& f_{j2}^{f_{i1}+\phi(p)} \% p \times f_{i3}^{-f_{j1}+\phi(p)} \% p) \% p \\
& \times f_{j4}^{f_{i5}+\phi(p)} \% p) \% p \times f_{j3}^{f_{i1}+\phi(p)} \% p) \% p \\
& \times f_{i2}^{-f_{j1}+\phi(p)} \% p) \% p \times f_{i4}^{-f_{j5}+\phi(p)} \% p) \% p \\
\pi_2 = (((((& f_{i2}^{f_{j1}+\phi(p)} \% p \times f_{j3}^{-f_{i1}+\phi(p)} \% p) \% p \\
& \times f_{i4}^{f_{j5}+\phi(p)} \% p) \% p \times f_{i3}^{f_{j1}+\phi(p)} \% p) \% p \\
& \times f_{j2}^{-f_{i1}+\phi(p)} \% p) \% p \times f_{j4}^{-f_{i5}+\phi(p)} \% p) \% p
\end{aligned} \tag{1}$$

Proof Sketch The completeness of our scheme can be easily proved according the Fermat's theorem [IRR90] based on Formula 1. The proof is accepted when both π_1 and π_2 are 1, and vice versa. The soundness holds because of the collision resistance of the hash function. To prove the hiding property, we can construct a simulator $Sim(x, r_i, pp)$, where x represents a simulation trapdoor for all non-uniform polynomial-time adversaries. It is obviously indistinguishable from the real commit algorithm because the adversaries do not know the uniformly random number r_i. We omit the formal proofs here.

3.4 Discrete Logarithm Problem based Key-Sharing

When a chunk is detected as a duplicate, there is no need to repeatedly store the duplicate chunk after the non-interactive zero-knowledge based redundancy detecting scheme. Instead, data owners should securely share the encryption key for duplicate chunks offline. Thus, we further propose a discrete logarithm problem based key sharing. Specifically, when the chunk to be uploaded is duplicated with the ck_i in redundancy detection, the delegate will

request the metadata $metas_i$ according to the tag_i from the cloud and send it back to the client. Only the client, knowing the plaintext of ck_i, can get the correct random number information from $metas_i$. Finally, the encryption key shared among data owners is the hash value of h_i and random number.

$$< tag_i, \underbrace{metas_i}_{< (r_j \cdot h_i^{h_i r_i})\%p,\ h_i^{r_i(\phi(p)-1)}\%p >} >$$

Figure 4: The Metadata Structure for Discrete Logarithm Problem based Key-Sharing

Figure 4 illustrates the metadata $metas$ of the duplicate chunk ck_i stored in the cloud. In this scenario, the client Alice wants to upload a chunk ck_m that is a duplicate of a chunk ck_i previously uploaded to the cloud by Bob. Bob generated a random number r_j and used it to compute the encryption key for ck_i as $hash(h_i, r_j\%p)$, where h_i is the hash value of ck_i. After the redundancy detection, Alice knows that ck_m is a duplicate and can calculate the encryption key of ck_i using the metadata $metas_i$ and tag tag_i of ck_i based on Formula 2. This key is securely shared among data owners offline and can be used for encrypting and decrypting the duplicate chunks. Any malicious attacker or delegates can hardly figure out the random number or other chunk content information from the $metas_i$ and tag_i of ck_i due to the discrete logarithm problem when p is a selected prime number.

$$r_j\%p = (((r_j \cdot h_i^{h_i r_i})\%p) \cdot (h_i^{r_i(\phi(p)-1)}\%p)^{h_m\%\phi(p)+\phi(p)})\%p \tag{2}$$

3.5 Secure Time-based Delivery Proof

This subsection proposes a robust method for verifying the secure time-based delivery of unique chunks. The idea of the proposed method is to verify that a delegate and a cloud have delivered a chunk within a predetermined time period. The delivery proof becomes invalid after the specified period. Specifically, the client selects one of the k designated delegates at random to deliver the unique data chunk to the cloud after redundancy detection. The cloud must store the chunk during the client's designated period. Both the delegate and the cloud must create and provide time-based delivery proofs, which are then sent back to the client via the delegate. Once the client successfully validates the proofs, the delegate notifies the system to store the data permanently. If the verification is unsuccessful, the client initiates a new round of the chunk-writing session.

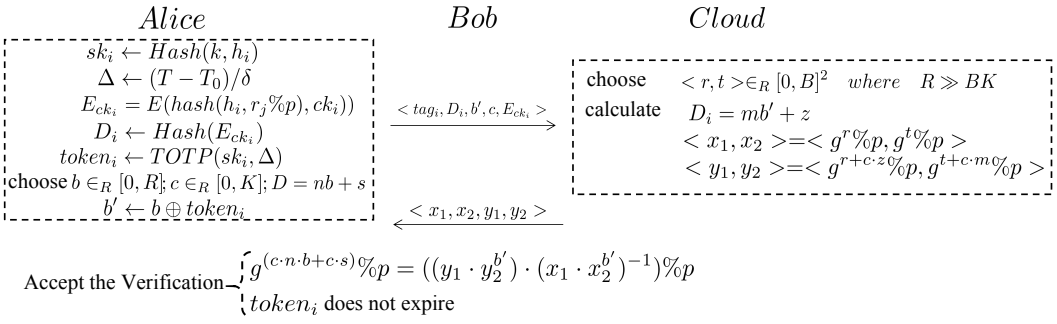

Figure 5: The Protocol for Secure Time-based Delivery Proof

To illustrate the secure time-based delivery proof, Figure 5 provides an example in which Alice, a client, uploads the unique chunk ck_i and Bob is the designated delegate to

write the chunk. To support the time-based proof for the unique chunk ck_i writing process, Alice sends a tuple $< tag_i, D_i, b', c, E_{ck_i} >$ to Bob. Here, E_{ck_i} represents the ciphertext of ck_i, and D_i is the hash value of the ciphertext. b' represents the Xor value of a random number b and the time-based token $token_i$, which is generated by the time-based one-time password (TOTP) function [ES18]. The TOTP function uses a temporary key sk_i and a period setting Δ, where Δ represents a time unit. Additionally, c is another random number generated by Alice. The cloud then receives the tuple $< tag_i, D_i, b', c, E_{ck_i} >$ from Bob and constructs a proof $< x1, x2, y1, y2 >$, as shown in Figure 5. Alice verifies the proof through Bob. Once the verification is successful, the writing process is complete.

4 Security Analysis

In this section, we formalize the security definition of PEO-Store and prove that the probability of figuring out the valid access pattern with a specific user is negligible. To simplify the proven process, We redefine the security definition of oblivious stores based on the previous three challenges. Intuitively, the security definition requires that the malicious attacker learns nothing about a specific user's access patterns and supports securely removing the redundancy among users. In other words, no information should be leaked about: 1) which data is being accessed by the user; 2) when it was last accessed by the user; 3) whether the user accessed the same data; 4) what is the valid access pattern for a user.

Let $\vec{y_i} := ((op_M, a_M, data_M)_i, \cdots, (op_1, a_1, data_1)_i)$ denotes a data request sequence of length M by user i, where each op_j denotes a $read(a_j)$ or a $write(a_j, data)$ operation. Specifically, a_j is the identifier of the request block, and $data_j$ denotes the bock content. In this notation, the tuples in the data request sequence chronologically correspond to the data operations. Let $A(\vec{y})$ denote the sequence of access to the remote storage given the sequence of data requests $\vec{y_i}$ of user i. In other words, the malicious nodes, colluding with the cloud, can observe the access patterns from delegates, like $A(\vec{y_i})$. **Our construction is secure if** (1) the probability that $A(\vec{y})$ is equals to the user x's valid request sequence $\vec{y_x}$ for any malicious is $Prob(A(\vec{y}) = \vec{y_i}|i = x)$, which is negligible, (2) the secure deduplication process leaks no information about the file content and keys, and (3) construction is correct in that it returns on input $\vec{y}$ data that is consistent with $\vec{y}$ with probability $\geq 1 - negl(||\vec{y}||)$.

Then, we assume the peer node number in PEO-Store is W. Due to the delegate design, each client randomly divides their data request sequence into several splices. The malicious attacker can only observe part of them as a delegate role or master the whole data requests initiated by the delegates. Thus, the probability of figuring out the correct client to initiate the data request is $\frac{1}{W}$. Obviously, we can get the $Prob(A(\vec{y}) = \vec{y_i}|i = x)$ by using the Bayes rule as Formula 3. Based on the definition of a negligible function [Wik22c], we can prove that $Prob(A(\vec{y}) = \vec{y_i}|i = x)$ is negligible.

$$Prob(A(\vec{y}) = \vec{y_i}|i = x) = \prod_{i=1}^{M} Prob(target(a_i)) = (W)^{-M} \tag{3}$$

Proof Outline: To simplify the proof, we consider the function $\mu(M) = (W)^{-M}$. Then, we can choose $M_0 = c^W \in N$. For any $M \geq M_0$, we have $(W)^{-M} = (W^{\log_W M})^{-\frac{M}{\log_W M}} = W^{-\frac{M}{\log_W M}}$. Since $M \geq M_0$, we know $\frac{M}{\log_W M} \geq \frac{M_0}{\log_W M_0} \geq \frac{M_0}{\sqrt{M_0}} = \sqrt{M_0} = c$. Thus, $\mu(M) = (W)^{-M} = W^{-\frac{M}{\log_W M}} \leq M^{-c}$. So, we have prove that for any $c \in N$, there exists $M_0 = c^W \in N$ such that for any $M \geq M_0$, $Prob(A(\vec{y}) = \vec{y_i}|i = x) \leq M^{-c}$, which conforms to the negligible function definition.

Therefore, the delegate setting in PEO-Store can easily block the mapping link from valid access patterns to a specific user compared with conventional ORAM. Moreover,

it can also support removing the redundancy while keeping the obliviousness property. The cryptography in PEO-Store is based on discrete logarithm problems, such as the non-interactive zero-knowledge based redundancy detecting scheme, key-sharing, and time-based delivery proof. Except for the client, no one can achieve private information before cracking the discrete logarithm problem when the prime number is extra large. In addition, each chunk's tag or encryption key is related to the random property in SHA-128, and the malicious can hardly crack it in probabilistic polynomial time through the existing cryptoanalysis.

As for consistency, we utilize the distributed hash table (DHT) to maintain the metadata among peer nodes, such as Chord [SMK+01]. Thus, the consistency in PEO-Store depends on the latency in the DHT protocol. When the peer nodes' latency is unacceptable, the peer node may encounter lookup failures and state inconsistency. In other words, An adversary may be able to make some set of nodes fail but have no control over the choice of the whole peer nodes. For example, the adversary may be able to affect only the nodes in a particular geographical region. Fortunately, there are several research works to achieve the low latency lookup in DHT protocol [SMK+01, JMTS16]. When the nodes join and leave, the DHT protocol can automatically keep consistency and replicate the metadata. We expect further improvement to explore the minimized latency of PEO-Store in the future.

5 Evaluation

Experimental Setup In this section, we implement the PEO-Store in JAVA, using Netty as the RPC library and Chord protocol as the distributed hash table. Each peer node and the cloud are equipped with an Intel(R) Core(TM) i7-4790 @3.60GHz 8-core CPU, 16GB RAM, a 500GB 5400 rpm hard disk, and connected to a 1Gbps network. We simulate a network size with 2^9 peer nodes. The cloud storage server is deployed to an Amazon EC2 instance. We compare the PEO-Store with two baselines. The first baseline is encrypted-only cloud storage with encryption by the Pseudo-Random function. That is, it encrypts the data without the deduplication and client requests but does not guarantee oblivious property. The second is the latest oblivious data access work, the Shortstack [VBKA22]. It serves as a reference for performance comparison.

DataSet and Workloads There are two kinds of datasets. The first one is the home dataset contains snapshots of students' home directories from a shared network file system (Fslhome). It was collected in the File system and Storage Lab (FSL) at Stony Brook University by the File Systems and Storage Lab, and Dell-EMC [TMB+12], which includes real users' daily home directories. The snapshots were collected between the end of the year 2011 and the beginning of the year 2014. The other is the standard YCSB benchmark [CST+10] to generate our dataset and workloads. We use workloads A(50% reads, 50% writes) and C (100% reads) for experiments.

5.1 Throughput Analysis

We now study the PEO-Store's throughput by varying workloads with different redundancy percentages. The workloads of Fslhome and TCSB-A exhibit 80% and 75% write redundancy percentages, respectively. Figure 6 shows the throughput results for each baseline. The throughput of the Shortstack baseline is 40 KOps in Fslhome, and it also achieves a similar throughput in YCSB-A and YCSB-C. Furthermore, the PEO-Store achieves almost 3× throughput compared with the Shortstack in Fslhome and YCSB-A while the encryption-only baseline is 2× ~3× faster than PEO-Store.

The benefits of PEO-Store stems from avoiding fake requests and avoiding duplicate data request compared with existing oblivious design. Specifically, once the chunk is detected as a redundancy, there is no need to transfer the ciphertext to the cloud. Instead,

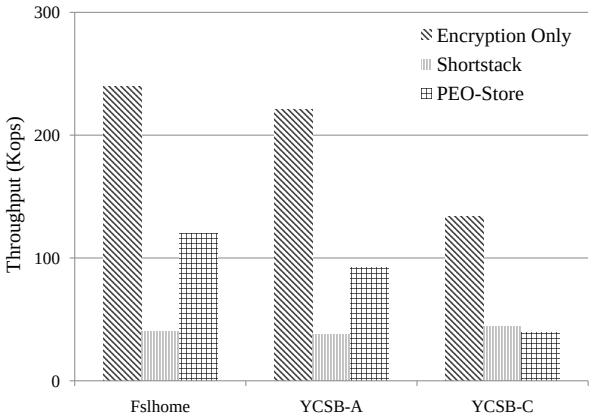

Figure 6: Throughput under Various Workloads

the delegate transfer the metadata of key sharing to the client. Moreover, PEO-Store does not require fake requests to distinguish the access patterns. The delegation way can effectively the mapping from valid access patterns to specific users. But, all of the YCSB-C workloads are read requests. The throughput of PEO-Store is also similar to the Shortstack without the benefits from deduplication.

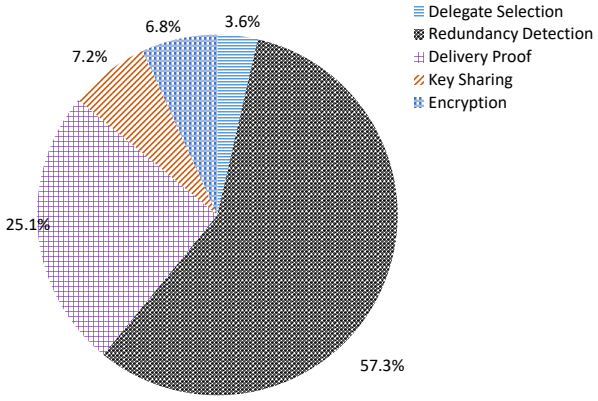

Figure 7: Proportion Time for Core Processes in PEO-Store

We further statistic the proportion of each scheme for average time cost in PEO-Store, as shown in Figure 7, such as the secure randomized delegate selection (Delegate Selection), non-interactive zero-knowledge based redundancy detecting (Redundancy Detection), secure time-based delivery proof (Delivery Proof), and the encryption for each chunk (Encryption). Figure 7 shows that the redundancy detection and delivery proof are the time-consuming schemes in PEO-Store. For example, there are at least one or two hops to look up or transfer the metadata of redundancy detection and delivery proof among peer nodes. And the latency for each hop is affected by the network quality, detailed in the following experiments. Key sharing is almost 8× faster than redundancy detection compared to encryption and delegate selection. For each duplicate chunk, it can avoid unnecessary delivery proofs.

We also evaluate PEO-Store for workloads with a different skew in YCSB-A, as shown in Figure 8. The PEO-Store's throughput scales linearly regardless of the skew but is

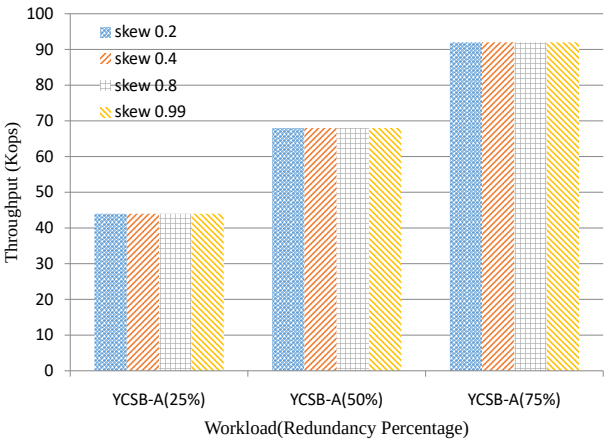

Figure 8: PEO-Store Throughput is unaffected by access skew with redundancy percentage in workloads

related to the workload's redundancy proportion. The reason is that the consistent hash can ensure that each peer node is responsible for an equal portion of the distributed hash table content. In other words, the load of the PEO-Store is evenly distributed, even though the heavily skewed workload.

5.2 Latency Overhead

To qualify the PEO-Store's latency overheads, we simulate the scenario that peer nodes continuously join and leave, which is reasonable and common in reality. Figure 8 shows the latency corresponds to one node joining and leaving every T seconds on average in Fslhome, where T is from 20 seconds to 80 seconds. PEO-Store increases latency by an additional 20ms compared to Shortstack. This growth is due to the lookup in the distributed hash table. However, the distributed hash table balances the load since each peer node receives roughly the same metadata and requires relatively little metadata movement when nodes join and leave the system. And we only show Fslhome workload results, as YCSB-A and YCSB-C results are similar. Nevertheless, these overheads are masked by the significantly larger WAN access latency.

5.3 Storage Overhead

Finally, we statistic the overall storage costs in the real workload, Fslhome, as shown in Table 1. Table 1 shows the normalized overall storage costs. We denote the ideal storage cost of Fslhome as one after completely removing the redundancy. Then, Encryption-only and Shortstack are 5.1× and 5.4× compared with the idea storage cost. Since the Pseudo-Random encryption and fake request mechanism in conventional oblivious schemes, it is hardly recognized and removes the duplicate data, which is critical for the cloud storage to improve the storage utilization and save the network bandwidth. PEO-Store can effectively remove the duplicate and requires only 37% more storage overhead than Ideal storage. The metadata causes these extra storage costs in PEO-Store.

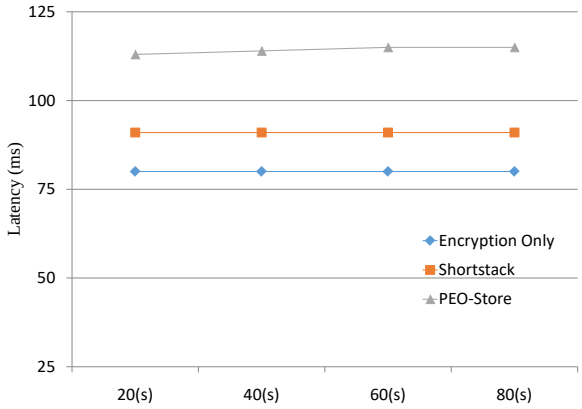

Figure 9: Latency with One Node Join or Leave for Every T seconds

Table 1: Normalized Storage Cost for Fslhome Workload

	Normalized Storage Cost
Ideal Storage	1
Encryption Only	5.1×
Shortstack	5.4×
PEO-Store	1.37×

6 Related Work

In this section, we mainly summarize relevant existing work, focusing on (1) the ORAM algorithm's optimization, (2) the ORAM with hardware enclave, and (3) the trusted proxy-based oblivious store.

ORAM Algorithm's Optimization: Oblivious RAM, first proposed by Golden Rich [Gol87], to resist the privacy leakage from both read and write operations. Path ORAM [SvDS+13] further organizes the data in tree-like structures. Each tree node is a bucket, including several blocks. The same block in different nodes has variance ciphertexts, and each request should constantly shuffle and re-encrypt the data. A simple request suffers additional unrelated block transmissions and heavy computations. Ring ORAM [RFK+15], Bucket ORAM [FNR+15] were proposed to reduce the bandwidth overhead on the memory bus by using different bucket organization and more complicated access flow control. Ren et al. [YHR+15] has improved the performance of ORAM by introducing the static super block structure and the dynamical adaptive algorithm. It saves the bandwidth in ORAM based on spatial locality property.

Unfortunately, the ORAM is still far from practical usage, and the situation will worsen in storage level. To alleviate this problem, Elaine Shi et al. [AKL+22] further propose the oblivious parallel RAM (OPRAM), which is a natural extension of ORAM to the (more realistic) parallel setting where several processors make concurrent accesses to a shared memory. Snoopy [DFD+21] comprises the load balancers and subORAM to securely achieve horizontal scaling by storing data partitions.

ORAM with Hardware Enclave: The hardware enclaves, such as Intel SGX, can effectively maintain the oblivious access and public states in the cloud scenario. ZeroTrace [SGF18] proposes several oblivious memory primitives from Intel SGX, which includes an efficient and flexible block-level memory controller. Oblix [MPC+18a], Opaque [ZDB+17a],

and ObliDB [EZ17] improve the oblivious query in the cloud by introducing new oblivious physical operators in hardware enclaves. Pro-ORAM [TJS19] improves the throughput using multi-threading Melbourne Shuffle with SGX enclaves. However, hardware enclaves do not entirely solve the problem of hiding access patterns for oblivious storage: the enclave side channels allow attackers to exploit data-dependent memory accesses to extract enclave secrets [BMD+17, VBWK+17].

Trusted Proxy-based Oblivious Store: CURIOUS [BNP+15] and TaoStore [SZA+16] employ a centralized proxy model with parallelism to break down the performance limitation by adapting ORAM with the cloud storage. The trusted proxy server maintains the oblivious state on behalf of the clients. Loco-Store further attempts to combine the locality with oblivious property and proposes a locality-based eviction algorithm to reduce the access time. Moreover, Pancake [GKL+20a] leverages a trusted proxy to transform a set of plaintext accesses to a uniformly distributed set of encrypted accesses that can be forwarded directly to an encrypted, non-oblivious storage server. While this approach achieves high throughput, the proxy remains a bottleneck as it must maintain a dynamic state about the request distribution. Thus, Shortstack [VBKA22] could batch and parallel the calculation to improve the throughput. While both of these techniques provide a powerful oblivious data access guarantee, additional trusted independent servers are a strong assumption that is very difficult to meet in commercial contexts. How to arrange for trusted third parties is still an unsolved open problem [ZLK11].

Moreover, obfuscating the access pattern in probability encryption and the fake access setting will seriously damage the benefits of removing redundancy in the cloud scenario, which has been adopted and deployed in many cloud storage products. However, existing secure deduplication schemes still disclose users' privacy from access patterns, even in an encrypted state. The PEO-Store is the first step to securely remove the redundancy while keeping the obliviousness property to achieve high throughput and storage utilization in a cloud storage scenario.

7 Conclusion

In conclusion, this paper has provided an overview of the challenges associated with applying Oblivious RAM (ORAM) to cloud environments, particularly in achieving practical usage of Oblivious Stores. We have identified two key issues: the unacceptable performance caused by obfuscating access patterns in probability encryption and the conflict between the requirement for obliviousness and the need to identify and remove redundant data. To address these challenges, we have proposed a practical Oblivious Store, PEO-Store, that leverages various techniques such as randomized delegate selection, non-interactive zero-knowledge-based redundancy detection, and discrete logarithm problem-based key sharing. Our theoretical analysis and experimental results demonstrate that PEO-Store outperforms existing state-of-the-art solutions, achieving up to 3× greater throughput and saving up to 74% storage space on average. We believe that PEO-Store provides a promising solution for achieving practical usage of Oblivious Stores in cloud scenarios while maintaining security and efficiency. Future work can explore the scalability and robustness of PEO-Store, as well as its application to other cloud storage scenarios. Overall, this paper highlights the importance of addressing the challenges associated with applying ORAM to cloud environments and provides a practical solution to this ongoing issue.

Acknowledgments

We would like to express our sincere gratitude to the anonymous reviewers for their valuable comments and suggestions on our poster paper. Their constructive feedback and insights

have helped us to improve the quality and clarity of our work.

References

[AKL+22] Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Enoch Peserico, and Elaine Shi. Optimal oblivious parallel RAM. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 2459–2521. SIAM, 2022.

[BFV19] Alex Biryukov, Daniel Feher, and Giuseppe Vitto. Privacy aspects and subliminal channels in zcash. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1813–1830, 2019.

[BKR13a] Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. Dupless: Server-aided encryption for deduplicated storage. *Cryptology ePrint Archive*, 2013.

[BKR13b] Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. Message-locked encryption and secure deduplication. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 296–312. Springer, 2013.

[BMD+17] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. Software grand exposure:{SGX} cache attacks are practical. In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, 2017.

[BNP+15] Vincent Bindschaedler, Muhammad Naveed, Xiaorui Pan, XiaoFeng Wang, and Yan Huang. Practicing oblivious access on cloud storage: the gap, the fallacy, and the new way forward. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 837–849. ACM, 2015.

[CBC+18] Natacha Crooks, Matthew Burke, Ethan Cecchetti, Sitar Harel, Rachit Agarwal, and Lorenzo Alvisi. Obladi: Oblivious serializable transactions in the cloud. In Andrea C. Arpaci-Dusseau and Geoff Voelker, editors, *13th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2018, Carlsbad, CA, USA, October 8-10, 2018*, pages 727–743. USENIX Association, 2018.

[CCR19] Hao Chen, Ilaria Chillotti, and Ling Ren. Onion ring ORAM: efficient constant bandwidth oblivious RAM from (leveled) TFHE. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 345–360. ACM, 2019.

[CHK22] Henry Corrigan-Gibbs, Alexandra Henzinger, and Dmitry Kogan. Single-server private information retrieval with sublinear amortized time. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2022.

[CS19] Anrin Chakraborti and Radu Sion. Concuroram: High-throughput stateless parallel multi-client ORAM. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

[CST+10] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with YCSB. In Joseph M. Hellerstein, Surajit Chaudhuri, and Mendel Rosenblum, editors, *Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC 2010, Indianapolis, Indiana, USA, June 10-11, 2010*, pages 143–154. ACM, 2010.

[DFD+21] Emma Dauterman, Vivian Fang, Ioannis Demertzis, Natacha Crooks, and Raluca Ada Popa. Snoopy: Surpassing the scalability bottleneck of oblivious storage. In Robbert van Renesse and Nickolai Zeldovich, editors, *SOSP '21: ACM SIGOPS 28th Symposium on Operating Systems Principles, Virtual Event / Koblenz, Germany, October 26-29, 2021*, pages 655–671. ACM, 2021.

[ES18] Emir Erdem and Mehmet Tahir Sandıkkaya. Otpaas—one time password as a service. *IEEE Transactions on Information Forensics and Security*, 14(3):743–756, 2018.

[EZ17] Saba Eskandarian and Matei Zaharia. Oblidb: Oblivious query processing using hardware enclaves. *arXiv preprint arXiv:1710.00458*, 2017.

[FNR+15] Christopher Fletcher, Muhammad Naveed, Ling Ren, Elaine Shi, and Emil Stefanov. Bucket oram: single online roundtrip, constant bandwidth oblivious ram. *Cryptology ePrint Archive*, 2015.

[GKL+20a] Paul Grubbs, Anurag Khandelwal, Marie-Sarah Lacharité, Lloyd Brown, Lucy Li, Rachit Agarwal, and Thomas Ristenpart. Pancake: Frequency smoothing for encrypted data stores. In Srdjan Capkun and Franziska Roesner, editors, *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pages 2451–2468. USENIX Association, 2020.

[GKL+20b] Paul Grubbs, Anurag Khandelwal, Marie-Sarah Lacharité, Lloyd Brown, Lucy Li, Rachit Agarwal, and Thomas Ristenpart. Pancake: Frequency smoothing for encrypted data stores. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2451–2468, 2020.

[Gol87] Oded Goldreich. Towards a theory of software protection and simulation by oblivious rams. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 182–194. ACM, 1987.

[GSW21] Daniel Günther, Thomas Schneider, and Felix Wiegand. Revisiting hybrid private information retrieval. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 2408–2410. ACM, 2021.

[IKK12] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society, 2012.

[IRR90] Kenneth Ireland, Michael Ira Rosen, and Michael Rosen. *A classical introduction to modern number theory*, volume 84. Springer Science & Business Media, 1990.

[JMTS16] Yaoqi Jia, Tarik Moataz, Shruti Tople, and Prateek Saxena. Oblivp2p: An oblivious peer-to-peer content sharing system. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pages 945–962. USENIX Association, 2016.

[JR15] Jonathan L. Dautrich Jr. and Chinya V. Ravishankar. Combining ORAM with PIR to minimize bandwidth costs. In Jaehong Park and Anna Cinzia Squicciarini, editors, *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy, CODASPY 2015, San Antonio, TX, USA, March 2-4, 2015*, pages 289–296. ACM, 2015.

[LAP15] Jian Liu, Nadarajah Asokan, and Benny Pinkas. Secure deduplication of encrypted data without additional independent servers. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 874–885, 2015.

[LPM+13] Jacob R. Lorch, Bryan Parno, James W. Mickens, Mariana Raykova, and Joshua Schiffman. Shroud: ensuring private access to large-scale data in the data center. In Keith A. Smith and Yuanyuan Zhou, editors, *Proceedings of the 11th USENIX conference on File and Storage Technologies, FAST 2013, San Jose, CA, USA, February 12-15, 2013*, pages 199–214. USENIX, 2013.

[LYG+15] Fangfei Liu, Yuval Yarom, Qian Ge, Gernot Heiser, and Ruby B Lee. Last-level cache side-channel attacks are practical. In *2015 IEEE symposium on security and privacy*, pages 605–622. IEEE, 2015.

[MJWT13] Bo Mao, Hong Jiang, Suzhen Wu, and Lei Tian. Leveraging data deduplication to improve the performance of primary storage systems in the cloud. In Guy M. Lohman, editor, *ACM Symposium on Cloud Computing, SOCC '13, Santa Clara, CA, USA, October 1-3, 2013*, pages 24:1–24:2. ACM, 2013.

[MPC+18a] Pratyush Mishra, Rishabh Poddar, Jerry Chen, Alessandro Chiesa, and Raluca Ada Popa. Oblix: An efficient oblivious search index. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 279–296. IEEE Computer Society, 2018.

[MPC+18b] Pratyush Mishra, Rishabh Poddar, Jerry Chen, Alessandro Chiesa, and Raluca Ada Popa. Oblix: An efficient oblivious search index. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 279–296. IEEE, 2018.

[OSF+16] Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. Oblivious {Multi-Party} machine learning on trusted processors. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 619–636, 2016.

[PKK+22] Jisung Park, Jeonggyun Kim, Yeseong Kim, Sungjin Lee, and Onur Mutlu. Deepsketch: A new machine learning-based reference search technique for post-deduplication delta compression. In Dean Hildebrand and Donald E. Porter, editors, *20th USENIX Conference on File and Storage Technologies, FAST 2022, Santa Clara, CA, USA, February 22-24, 2022*, pages 247–264. USENIX Association, 2022.

[PP14] João Paulo and José Pereira. A survey and classification of storage deduplication systems. *ACM Comput. Surv.*, 47(1):11:1–11:30, 2014.

[RFK+15] Ling Ren, Christopher W. Fletcher, Albert Kwon, Emil Stefanov, Elaine Shi, Marten van Dijk, and Srinivas Devadas. Constants count: Practical improvements to oblivious RAM. In Jaeyeon Jung and Thorsten Holz, editors, *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*, pages 415–430. USENIX Association, 2015.

[RRM20] Maan Haj Rachid, Ryan D. Riley, and Qutaibah M. Malluhi. Enclave-based oblivious RAM using intel's SGX. *Comput. Secur.*, 91:101711, 2020.

[Sch13] Bruce Schneier. The us government has betrayed the internet. we need to take it back. *The Guardian*, 2013.

[SGF18] Sajin Sasy, Sergey Gorbunov, and Christopher W. Fletcher. Zerotrace : Oblivious memory primitives from intel SGX. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.

[SJG22] Sajin Sasy, Aaron Johnson, and Ian Goldberg. Fast fully oblivious compaction and shuffling. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2565–2579, 2022.

[SKH17] Youngjoo Shin, Dongyoung Koo, and Junbeom Hur. A survey of secure data deduplication schemes for cloud storage systems. *ACM computing surveys (CSUR)*, 49(4):1–38, 2017.

[SMK+01] Ion Stoica, Robert Tappan Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Rene L. Cruz and George Varghese, editors, *Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 27-31, 2001, San Diego, CA, USA*, pages 149–160. ACM, 2001.

[SS13] Emil Stefanov and Elaine Shi. Oblivistore: High performance oblivious distributed cloud data store. In *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*. The Internet Society, 2013.

[SvDS+13] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: an extremely simple oblivious RAM protocol. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 299–310. ACM, 2013.

[SZA+16] Cetin Sahin, Victor Zakhary, Amr El Abbadi, Huijia Lin, and Stefano Tessaro. Taostore: Overcoming asynchronicity in oblivious data storage. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 198–217. IEEE Computer Society, 2016.

[TJS19] Shruti Tople, Yaoqi Jia, and Prateek Saxena. PRO-ORAM: practical read-only oblivious RAM. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses, RAID 2019, Chaoyang District, Beijing, China, September 23-25, 2019*, pages 197–211. USENIX Association, 2019.

[TLXX18] Wenlong Tian, Ruixuan Li, Weijun Xiao, and Zhiyong Xu. Pts-dep: A high-performance two-party secure deduplication for cloud storage. In *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 700–707. IEEE, 2018.

[TLXX22] Wenlong Tian, Ruixuan Li, Zhiyong Xu, and Weijun Xiao. Loco-store: Locality-based oblivious data storage. *IEEE Trans. Dependable Secur. Comput.*, 19(2):1395–1406, 2022.

[TMB+12] Vasily Tarasov, Amar Mudrankit, Will Buik, Philip Shilane, Geoff Kuenning, and Erez Zadok. Generating realistic datasets for deduplication analysis. In Gernot Heiser and Wilson C. Hsieh, editors, *2012 USENIX Annual Technical Conference, Boston, MA, USA, June 13-15*, 2012.

[TMPD19] Santhosh Kumar T, Debadatta Mishra, Biswabandan Panda, and Nayan Deshmukh. Cowlight: Hardware assisted copy-on-write fault handling for secure deduplication. In *Proceedings of the 8th International Workshop on Hardware and Architectural Support for Security and Privacy, HASP@ISCA 2019, June 23, 2019*, pages 3:1–3:8. ACM, 2019.

[VBKA22] Midhul Vuppalapati, Kushal Babel, Anurag Khandelwal, and Rachit Agarwal. SHORTSTACK: distributed, fault-tolerant, oblivious data access. In Marcos K. Aguilera and Hakim Weatherspoon, editors, *16th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2022, Carlsbad, CA, USA, July 11-13, 2022*, pages 719–734. USENIX Association, 2022.

[VBMW+18] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient {Out-of-Order} execution. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 991–1008, 2018.

[VBWK+17] Jo Van Bulck, Nico Weichbrodt, Rüdiger Kapitza, Frank Piessens, and Raoul Strackx. Telling your secrets without page faults: Stealthy page {Table-Based} attacks on enclaved execution. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1041–1056, 2017.

[Wik22a] Wikipedia contributors. Consistent hashing — Wikipedia, the free encyclopedia, 2022. [Online; accessed 23-December-2022].

[Wik22b] Wikipedia contributors. Diginotar — Wikipedia, the free encyclopedia, 2022. [Online; accessed 1-December-2022].

[Wik22c] Wikipedia contributors. Negligible function — Wikipedia, the free encyclopedia, 2022. [Online; accessed 7-January-2023].

[WST12] Peter Williams, Radu Sion, and Alin Tomescu. Privatefs: a parallel oblivious file system. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 977–988. ACM, 2012.

[XCP15] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *2015 IEEE Symposium on Security and Privacy*, pages 640–656. IEEE, 2015.

[YHR+15] Xiangyao Yu, Syed Kamran Haider, Ling Ren, Christopher Fletcher, Albert Kwon, Marten Van Dijk, and Srinivas Devadas. Proram: dynamic prefetcher for oblivious ram. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, pages 616–628. IEEE, 2015.

[YLL22] Zuoru Yang, Jingwei Li, and Patrick P. C. Lee. Secure and lightweight deduplicated storage via shielded deduplication-before-encryption. In Jiri Schindler and Noa Zilberman, editors, *2022 USENIX Annual Technical Conference, USENIX ATC 2022, Carlsbad, CA, USA, July 11-13, 2022*, pages 37–52. USENIX Association, 2022.

[YXT+22] Xuming Ye, Xiaoye Xue, Wenlong Tian, Ruixuan Li, Weijun Xiao, Zhiyong Xu, and Yaping Wan. Chunk content is not enough: Chunk-context aware resemblance detection for deduplication delta compression. In Ali Bilgin, Michael W. Marcellin, Joan Serra-Sagristà, and James A. Storer, editors, *Data Compression Conference, DCC 2022, Snowbird, UT, USA, March 22-25, 2022*, page 492. IEEE, 2022.

[ZDB+17a] Wenting Zheng, Ankur Dave, Jethro G. Beekman, Raluca Ada Popa, Joseph E. Gonzalez, and Ion Stoica. Opaque: An oblivious and encrypted distributed analytics platform. In Aditya Akella and Jon Howell, editors, *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, pages 283–298. USENIX Association, 2017.

[ZDB+17b] Wenting Zheng, Ankur Dave, Jethro G Beekman, Raluca Ada Popa, Joseph E Gonzalez, and Ion Stoica. Opaque: An oblivious and encrypted distributed analytics platform. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 283–298, 2017.

[ZLK11] Dimitrios Zissis, Dimitrios Lekkas, and Panayiotis Koutsabasis. Cryptographic dysfunctionality-a survey on user perceptions of digital certificates. In *Global Security, Safety and Sustainability & e-Democracy*, pages 80–87. Springer, 2011.