Understanding the Security Challenges of Oblivious Cloud Storage with Asynchronous Accesses

Cetin Sahin, Aaron Magat, Victor Zakhary, Amr El Abbadi, Huijia (Rachel) Lin, Stefano Tessaro
University of California, Santa Barbara
{cetin, ajmagat, victorzakhary, amr, rachel.lin, tessaro}@cs.ucsb.edu

***Abstract*—This demonstration introduces the database community to state-of-the-art cryptographic methods that ensure efficient oblivious access to cloud data. In particular, we explore oblivious storage systems which hide both the content of data and data access patterns from an untrusted cloud provider. The demo considers the popular and realistic setting where multiple users from a trusted group asynchronously access and edit potentially overlapping data sets through a trusted proxy. We present a detailed implementation of TaoStore (Sahin et al., S&P 2016), a new tree-based ORAM scheme that processes client requests concurrently and asynchronously in a non-blocking fashion, resulting in substantial gains in throughput, simplicity, and flexibility over previous systems. The demo is presented in the context of a pedagogical game, Guess the Access, which allows participants to play as an adversary trying to guess queries against TaoStore or ObliviStore (Stefanov and Shi, S&P 2013), a recent oblivious storage system which has been shown to leak access patterns. The proposed game will highlight the subtleties and intricacies that underlie the cryptographic methods used to design oblivious storage systems.**

Video: https://youtu.be/1p1dPGoRUjc

I. Introduction

For many potential users skeptical about joining the cloud, confidentiality remains the main concern. In this context, encryption alone is not sufficient to solve all privacy challenges. This is because *access patterns* are typically *not* hidden from the cloud provider, allowing the provider to, for example, detect whether and when the same data item is accessed more than once. Even with encryption of the actual data, these data access patterns can leak sensitive information when combined with prior knowledge, as shown in [2].

Oblivious RAM (ORAM) is the standard approach to make access patterns *oblivious*. However, most ORAM solutions are still not applicable in real-world scenarios, as they both experience low throughput under concurrent loads due to handling operation requests *sequentially* and fail to exploit network asynchronicity. A typical oblivious cloud storage scenario considers multiple users from a trusted group (e.g., employees of the same company) who need to privately access data sets which may overlap. The question that needs to be answered is how to exploit asynchrony and concurrency for higher throughput while hiding access patterns.

One attempt to address these concerns is ObliviStore [4], a partition-based ORAM scheme which leverages parallelism to increase throughput. ObliviStore is also the first work to consider the proxy model, in which user accesses are mediated by a shared (trusted) proxy which coordinates accesses and at the same time reduces the amount of information leaked to the cloud. Unfortunately, ObliviStore does not consider information leakage due to the timing of responses to user requests, which has been shown to be a serious threat for access privacy [3]. In a separate study, Bindschaedler et al. [1] show that ObliviStore has a subtle security issue even under its own security definition.

Recently, we proposed TaoStore [3] which introduces a new tree-based ORAM scheme that uses a proxy model similar to ObliviStore. When compared to existing ORAM schemes, TaoStore is able to provide better performance for concurrent, multi-client use by fully leveraging the benefits of asynchrony. In addition, TaoStore is able to fully resist attacks in an asynchronous setting, and is provably secure under our new formal security definition, *adaptive asynchronous obliviousness* (aaob-security) [3], in which an attacker has the power to schedule read/write requests at any point in time, control the scheduling of messages, and learn *when* requests are answered by the ORAM client.

In this demonstration, we propose *Guess the Access*, an educational game which highlights the important security features of oblivious storage systems, and showcases how TaoStore achieves access pattern privacy, even in the presence of an adversary, while still delivering high throughput.

Our overall goal is to provide the database community with an opportunity to appreciate some of the intricate issues involved in the development and understanding of access security, specifically in a distributed cloud-based data management setting. This demonstration should bridge the gap between the security and the database community, and help a database audience recognize the complexity of attacks that can be mounted on oblivious storage as well as the resulting significant overheads that truly secure oblivious stores require.

II. Overview of TaoStore

In this section, we give a brief overview of TaoStore [3]. Building upon the *tree-based* ORAM scheme, TaoStore uses a new scheme, TaORAM, whose basic steps are as follow: a) At the arrival of a request for a block, the path containing that block is fetched immediately from the server. b) Upon the retrieval of the path from the server, the corresponding read/write request is answered, and the path is inserted into a

[1]This work is partly funded by NSF grants CNS-1528178 and CCF-1442966.

local *subtree* data structure before being flushed. c) Immediately after flushing k paths, the paths' re-encrypted contents are written back to the server (and appropriate nodes deleted from the local subtree).

TaORAM manages *multiple* paths fetched concurrently from the server without waiting for on-going flush and write-back operations to complete. To do so while maintaining correctness, TaORAM relies on a number of data structures, such as a *subtree* which is a local subset of the entire server-side tree storage, the *request map* which is responsible for keeping track of when a fake read should be issued (to help with issues of network asynchrony), the *sequencer* which is responsible for making sure all requests are responded to in sequential order (thus preventing the attack against CURIOUS [3] from working against TaORAM), and a number of other structures. In addition to these data structures, TaORAM imposes a number of conditions that must be adhered to in order to ensure correctness, such as the *fresh-subtree* invariant which states that the content of paths in the subtree and stash are always up-to-date, while the server contains the most up-to-date content for the remaining blocks. This invariant ensures that a client always receives the most recent information available, regardless of any concurrent flush or write-back operations. These data structures and conditions allow TaoStore to both operate correctly and securely in an asynchronous environment while providing high throughput.

III. Guess the Access

We first introduce the first fully implemented version of TaoStore, which was created to be useable not only for research purposes, but also real world applications. As such, this version of TaoStore, which will be made open source, has many improvements over the initial implementation introduced in [3], including the ability to deploy storage partitioning. This will help clarify the complexity of not only TaoStore, but oblivious storage in general, and will encourage others to try and use the system in their own applications to see the impact oblivious storage can have on security, throughput, etc.

We use this implementation of TaoStore, along with the ObliviStore code from [1], to propose *Guess the Access*, a game designed to showcase both oblivious storage systems by allowing attendees to observe and gain intuitions about the data access security each provides, and with some thought can be used to reveal an attack on ObliviStore that is ineffective against TaoStore (see [1], [3]). In this game, attendees will play the role of an adversary and attempt to guess the queries of simulated, non-adversarial clients who are using one of the two oblivious cloud storage systems. As an adversary, attendees will be given a number of adversarial powers in the game (e.g. knowing when requests are answered), along with the ability to dictate certain settings of the simulated clients.

Guess the Access simulates two different machines: M_{proxy}, which executes TaORAM or ObliviStore and M_{server}, which acts as the cloud storage. The adversary can monitor the messages exchanged between the two machines, as well as the timing of these messages. Note that these messages are not the same as the query itself, but rather the communication used between the two machines to help fulfill the query.

Playing the Devil: Recall that the attendee is given all the adversarial powers that are outlined in our security definition (see [3]). Once the initial setup of the game is complete and a random query workload is generated, the attendee is given a chance to exercise these adversarial powers. For instance, the attendee will be able to change the order of the queries, as well as control the network delay of each individual message exchanged between M_{proxy} and M_{server}. The adversary will also be allowed to create, insert, and schedule their own queries into the workload before beginning the game.

Monitoring: During the game, the messages exchanged between M_{proxy} and M_{server} will be shown on both machines with corresponding message types, timestamps, and other relevant information. In addition, the attendee will be provided a timing diagram showing the movement of each message to help visualize the exchange of information between machines.

Reveal the Access: After the workload is executed, the attendee will be asked to enter their guesses about the content of the queries using an interface on M_{proxy}. Once all guesses are entered, the game will calculate a score for the attendee based on the amount of information that was correctly guessed. Lastly, the game will reveal the content of the queries and show the attendee a comparison between their score and the score one would expect to achieve through random guessing.

Given a sufficient number of games, attendees should discover that regardless of the approach, their score when using TaoStore is no better than what can be achieved through random guessing. The same can not be said of ObliviStore, as there exists a concrete attack that will allow an attendee to guess which data items are being accessed with 100% accuracy, regardless of data set size (details in [1] and [3]).

IV. Conclusion

The goal of this demonstration is two fold. First, we provide for the first time a complete implementation of TaoStore, an oblivious asynchronous cloud-based storage system. TaoStore is full of subtle details, and our implementation highlights all of these aspects, thus introducing the audience to some of the many intricate aspects of privacy preserving access. Second, through our pedagogical Guess the Access game, we introduce database researchers to many of the cryptographic subtleties when designing oblivious storage systems.

References

[1] V. Bindschaedler, M. Naveed, X. Pan, X. Wang, and Y. Huang. Practicing oblivious access on cloud storage: The gap, the fallacy, and the new way forward. In *CCS*, pages 837–849. ACM, 2015.

[2] M. S. Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *NDSS*, 2012.

[3] C. Sahin, V. Zakhary, A. El Abbadi, H. R. Lin, and S. Tessaro. Taostore: Overcoming asynchronicity in oblivious data storage. In *IEEE SP*, pages 198–217, 2016.

[4] E. Stefanov and E. Shi. Oblivistore: High performance oblivious cloud storage. In *IEEE SP*, pages 253–267, 2013.