A Comparative Study of Privacy-Preserving Computation Techniques

Contrasting ORAM, MPC, TEEs, Structured Encryption, and Homomorphic Encryption

Author:
Sergiu-Nicolae Stancu
s.n.stancu@student.tudelft.nl
Responsible professor and supervisor:
Lilika Markatou

Introduction

- As the world increasingly relies on cloud and outsourced storage, a concern over the security of this practice arises. It was shown that a malicious server can infer up to 80% of the search queries, just by looking at the data access patterns.

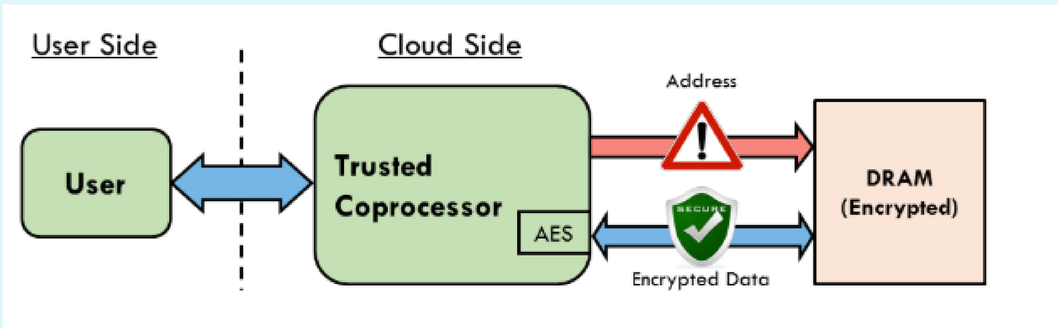

Figure 1: Insecure Model of outsourcing storage **[1]**

- Oblivious Random Access Machines aim to tackle this problem by hiding the data acccess pattern, such that an adversary will not be able to distinguish between a fake and a real program, having the same length.

Background and Motivation

- ORAM was first proposed by Goldreich and Ostrovsky **[2]**, which provided some foundational research, but failed to come up with a feasible implementation, due to high worst-case access costs. However, newer techniques were proposed that made ORAM practical today.
- Homomorphic Encryption, Structured Encryption, Multi-Party Computation and Trusted Execution Environments are other techniques aiming to preserve privacy and allow for computations on encrypted data.
- The literature survey was conducted using the Snowball Sampling Method.

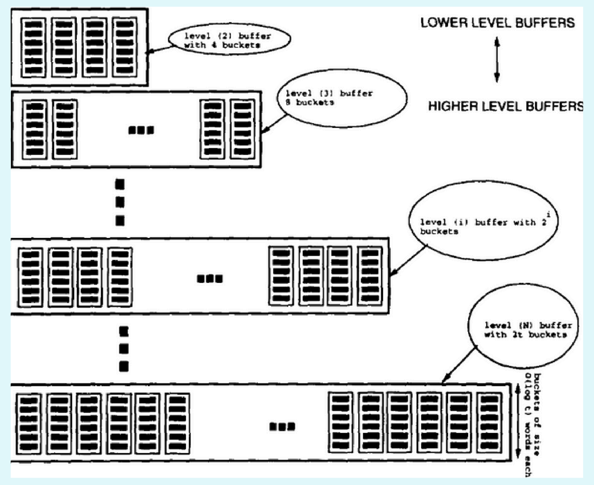

Figure 2: Hierarchical ORAM **[2]**

Research Questions

- How did ORAM evolve and reach the current state it is in?
- Where does ORAM fit in the context of Privacy-Preserving Computation and how does it compare/complement other techniques such as HE, SE, TEE and MPC?

Main ORAM Techniques

- There have been multiple improvements to ORAM. These are some of the most notable:

Scheme	Block Size	Amortized Cost	Worst-case Cost	Client Storage	Server Storage
GO[2]	$\Omega(\log N)$	$O(\log^3 N)$	$\Omega(N)$	$O(1)$	$O(N\log(N))$
SSS [4]	$\Omega(\log N)$	$O(\log^2 N)$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(N)$
Binary Tree ORAM (Shi)[5]	$\Omega(\log N)$	$\tilde{O}(\log^2 N)$	$\tilde{O}(\log^3 N)$	$O(1)$	$O(N)$
Goodrich-Mitzenmacher [6]	$\Theta(1)$	$O(\log^2 N)$	$O(N\log N)$	$O(N^\alpha), \alpha < 1$	$O(N)$
Path ORAM (B = $\Omega(\log^2 N)$)[7]	$\Omega(\log^2 N)$	$O(\log N)$	$O(\log N)$	$O(\log N)\omega(1)$	$O(ZN)$
Path ORAM (B = $O(\log N)$) [7]	$O(\log N)$	$O(\log^2 N)$	$O(\log^2 N)$	$O(\log N)\omega(1)$	$O(ZN)$
Ring ORAM [8]	$\Omega(\log^2 N)$	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(ZN)$
Burst ORAM [9]	$\Omega(\log^2 N)$	$\tilde{O}(\log N)$	$\tilde{O}(\log N)$	$\tilde{O}(\log N)$	$O(ZN)$

Z - parameter referring to the number of blocks in a tree node, Õ hides poly loglog terms
Figure 2: Overview of some significant ORAM constructions

- Newer techniques manage to achieve better asymptotics, but may be less practical due to hidden constants. Other techniques use FHE to minimize bandwidth, but the server becomes a bottle-neck.

Comparison of ORAM with MPC, TEE, StE and FHE

Technique	Computation	Parties	Applicability	Use cases	Threat model	Leakage	Overhead
ORAM	Data access	Client(s)-Server(s)	Used in secure processors + oblivious DBs	SGX integration, ObliDB **[3]**, Signal	Semi-honest or malicious	Through side-channels	Logarithmic
FHE	Any computation	Client-Server	Implemented in open-source libraries	Sensitive data analysis, Recommender systems	IND-CCA2 Adaptive attack	None	High: impractical at the moment
StE	Specific data access on encrypted data structures	Non-interactive Client-server	Practical protocols for specific structures	Encrypted DBMS	Semi-honest	Access patterns and response volumes	Sublinear
MPC	Generalized computation	Distributed parties	Used in practice with limitations	Secure Auctions, DNA comparison	Semi-honest or malicious	Only function output	Constant or Linear
TEE	Any computation	Interactive Client-Server + attestation service	Can be used in cloud deployment	Data analytics, Trusted AI workloads	Malicious OS on the server	Access patterns + plaintext in CPU	Near-native performance

Discussion

- Path ORAM[7] was one of the most influential papers in the field thanks to its simplicity, low amortized and worst-case overhead and the low hidden constants.
- Path ORAM has found multiple other real-world applications. Most notably, it is already being used in the Signal messaging app [10], for private contact discovery, along Intel SGX[11].
- StE can also be complemented by ORAM to hide the data access patterns and minimize its leakage.
- There are also oblivious query processing engines that implemented Path ORAM, such as ObliDB[3].

Conclusion

- ORAM has made huge improvements over the years and has taken many steps forward to become a practical and usable scheme for hiding data access patterns. In addition it can be optimized for different use cases, making it suitable for different scenarios.
- However, it still incurs a significant overhead and can not be adopted by time-sensitive applications. Future advancements in FHE can make ORAM a practical solution for all use-cases.
- Multiple techniques for privacy-preserving computation have been presented, but there is no technique which solves all problems. A user needs to prioritize and decide based on what is most important for them, balancing security, functionality, efficiency and usability for each individual scenario.

References

[1]"Oblivious RAM (ORAM) Research," Secure Computation Lab, University of Connecticut. [Online]. Available: https://scl.engr.uconn.edu/research/oram.php. [Accessed: Jun. 21, 2025]
[2] O. Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In Proceedings of the nineteenth annual ACM conference on Theory of computing- STOC'87, pages 182–194, New York, New York, United States, 1987. ACM Press.
[3] Saba Eskandarian and Matei Zaharia. ObliDB: obliv ious query processing for secure databases. Proceed ings of the VLDB Endowment, 13(2):169–183, October 2019.
[4] Emil Stefanov, Elaine Shi, and Dawn Song. Towards practical oblivious ram. 2012.
[5]Elaine Shi. Oblivious RAM with O((log N)3) Worst Case Cost.
[6] Michael T. Goodrich and Michael Mitzenmacher. Privacy-Preserving Access of Outsourced Data via Oblivious RAM Simulation. In Luca Aceto, Monika Henzinger, and Ji ˇ r ´ ı Sgall, editors, Automata, Lan guages and Programming, volume 6756, pages 576 587. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. Series Title: Lecture Notes in Computer Science.
[7]Emil Stefanov, Marten van Dijk, Elaine Shi, T-H. Hu bert Chan, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path oram: An extremely simple oblivious ram protocol. Journal of the ACM (JACM), 65(4):1–26, 2018.
[8]Ling Ren, Christopher W Fletcher, Albert Kwon, Emil Stefanov, Elaine Shi, Marten van Dijk, and Srinivas Devadas. Ring ORAM: Closing the Gap Between Small and Large Client Storage Oblivious RAM.
[9]Jonathan Dautrich and Elaine Shi. Burst ORAM: Mini mizing ORAM Response Times for Bursty Access Pat terns.
[10] Signal. Signal contact discovery, 2024. Accessed: 2025-06-17.
[11]Victor Costan and Srinivas Devadas. Intel SGX Ex plained, 2016. Publication info: Preprint.

TUDelft