Cryptography

Boran Erol

March 2024

1 Motivation and Introduction

Secret-Sharing of ORAM Data Structure. 1997 paper from Rafi. Probably one of his top 10 papers.

In oblivious RAM, we trust the CPU but we don't trust the memory. Encrypting the data you upload isn't enough – the access pattern leaks information about your data. Moreover, identical records might also create issues.

This is useful for cloud computing. It's funny that the original intent was software protection and the idea was to hide the access patterns of CPUs against RAMs.

Notice that one trivial solution is to scan the entire memory at every step of the computation. Therefore, the question is not about feasibility, the question is to reduce the overhead. For example, merge sort is not ORAM, but bubble sort is. Notice that merge sort runs in time $O(n \log n)$ but bubble sort runs in time $O(n^2)$. We'd like to build an ORAM compiler.

We formalize ORAM using a simulator that creates indistinguishable access patterns.

The space overhead is very minimal.

2 Naive Oblivious RAM

3 Lower Bound on ORAM

Goldreich and Ostrovsky in 1996 proved a $\omega(\log n)$ lower bound for statistical security in the balls and bins model.

The lower bound isn't there for the computational model.

They also proved this in the Offline ORAM model This means that the entire logical sequence of the program is known in advance, including all the address and the data. The proof works by a counting argument.

Boyle and Naor in 2016 proved that an $\omega(\log n)$ lower bound for offline ORAM not in the balls and bins model implies an $\omega(n \log n)$ lower bound for sorting circuits.

Larsen and Nielsen in 2018 proved a $\omega(\log n)$ lower bound in the online ORAM model with computational security. They use an information transfer technique.

3.1 Lower Bound of Larsen and Nielsen

It is based on an **information transfer** technique of Patrascu and Demaine in '06. It uses the cell probe model of Yao'81. In other words, computation is free and we only charge for probes.

4 Sorting Networks

These allow you to shuffle the data in the cloud using random keys.

4.1 Batcher's Sorting Network

Batcher's sorting network uses $O(n \log^2 n)$ gates and is $O(log^2 n)$ depth.

4.2 AKS Network

The AKS network uses $O(n \log n)$ gates and $O(\log n)$ depth.

The constant is in the billions, so even though the asympototic behavior is better, it's not usable.

5 Hierarchical ORAM

Ostrovsky's observation was the following: why should we keep shuffling everything if we're only accessing a subset of the data? Using this, he devised a scheme where we didn't keep wastefully shuffling everything.

6 Tree Based ORAM

The main invariant we preserve is

How do we prevent the root bucket from overflowing?

Every time we access a block, we access a random path.