CT Advanced Computing Center (CACC) Security Seminar Series 2024

Speaker: Ananya Appan University of Illinois Urbana-Champaign
Date: November 6, 2024
Time: 11:15 12:30pm
Location: ITE 336
Meeting Link : https://uconn-cmr.webex.com/uconn-cmr/j.php?MTID=m4d36ddf4d36edd727aa04331cdb7d7f6
Meeting number (access code): 2633 368 4613
Meeting password: MqmJMizs533

Oblivious Single Access Machines: A New Model for Oblivious Computation

Oblivious RAM (ORAM) allows a client to securely outsource memory storage to an untrusted server. It has been shown that no ORAM can simultaneously achieve small bandwidth blow-up, small client storage, and a single roundtrip of latency.
We consider a weakening of the RAM model, which we call
the Single Access Machine (SAM) model. In the SAM model, each memory slot can be written to at most once and read from at most once. We adapt existing tree-based ORAM to obtain an oblivious SAM (OSAM) that has O (log n) bandwidth blow-up (which we show is optimal), small client storage, and a single roundtrip.
OSAM unlocks improvements to oblivious data structures/algorithms. For instance, we achieve oblivious unbalanced binary trees (e.g. tries, splay trees). By leveraging splay trees, we obtain a notion of caching ORAM, where an access in the worst case incurs amortized O(log2 n) bandwidth blow-up and O (log n) roundtrips, but in many common cases (e.g. sequential scans) incurs only amortized O (log n) bandwidth blow-up and O (1) roundtrips. We also give new oblivious graph algorithms, including computing minimum spanning trees and single source shortest paths, in which the OSAM
client reads/writes O (E|.| · log E||) words using O (|E|) roundtrips, where |E| is the number of edges. This improves over prior custom solutions by a log factor.
At a higher level, OSAM provides a general model for oblivious computation. We construct a programming interface around OSAM that supports arbitrary pointer-manipulating programs such that dereferencing a pointer to an object incurs O (log d log n) bandwidth blowup and O (log d) roundtrips, where d is the number of pointers to that object. This new interface captures a wide variety of data structures and algorithms (e.g., trees, tries, doubly-linked lists) while matching or exceeding prior best asymptotic results. It both unifies much of our understanding of oblivious computation and allows the programmer to write oblivious algorithms combining various common data structures/ algorithms and beyond.