Research Statement

Summary. My work lies at the intersection of systems and theory. I am interested in practical system designs and implementations but also believe in the importance of writing proofs and grounding builds in core fundamentals. I have built kernel drivers, written security proofs for applied crypto protocols, and designed secure hardware and processors while in IBM. I find this systems-theory spot extremely interesting and I intend to continue exploring it in the years to come. My dissertation primarily focuses on provably securing data stored on clouds and local storage devices using efficient access-privacy mechanisms.

1 Dissertation Overview

The massive amounts of data stored and processed in shared (often public) online spaces raise essential security and privacy concerns.

Encryption of data is usually the first step towards ensuring some form of privacy. But access patterns leak significant amounts of information even for encrypted data, primarily because application accesses are data-dependent.

To this end, oblivious RAMs (ORAM) have been proposed as a solution. Yet, ORAMs are not considered viable for practical deployments due to their prohibitive overheads.

In my dissertation, I design, secure and implement access privacy mechanisms addressing important practical requirements: i) enabling shared parallel multi-client access [5], ii) high-throughput mechanisms for traditional storage media [2], and iii) securing application-specific requirements [3, 8]. In the following, I briefly describe some of these results.

Multi-Client Access Privacy. Sharing data among multiple clients is extremely common in clouds. In this setting, it is straightforward to provide access privacy by deploying existing oblivious RAM schemes to support multiple clients by sharing credentials and storing on the server data structures that are normally maintained client-side, to ensure state consistency across multiple clients. However, in such a setup, to maintain access privacy, only one client can be allowed to access the server-hosted data structures at any one time. This reduces overall throughput and significantly increases query latency.

To solve this problem, two independent approaches have been proposed in prior works. The first approach leverages inter-client communication for synchronizing parallel accesses. However, constant inter-client connectivity poses barriers that are often times difficult to handle in real-life scenarios.

The second approach leverages a trusted proxy to eliminate the need for inter-client awareness. All client requests are routed to the proxy, which deploys multiple threads to fetch data from the server-hosted data structures and satisfy
numerous client requests at once. The need for a trusted third-party component however severely limits applicability. Moreover, a trusted proxy is difficult to deploy in reality and also constitutes a single point of failure/compromise.

In [5], I designed and implemented the first parallel oblivious access mechanism for non-interacting stateless clients that does not require any additional components or assumptions. A major insight behind the construction is the fact that during data access, only a subset of the server-hosted data structures require parallel access with privacy guarantees. Everything else can be implemented as efficient oblivious data structures that are concurrently accessed.

Further, since a major contributor to ORAM latency is the expensive “eviction” step in which client-resident data is reshuffled and reinserted back encrypted into the main server database, the construction also employs a parallel eviction scheme that allows multiple clients to coordinate indirectly and perform evictions in parallel, and in the background without blocking queries.

The resulting ORAM construction (ConcurORAM) has small upper-bounded query times and scales extremely well with increasing number of clients. For example, about 65 queries per second can be executed in parallel by 30 concurrent clients with only a 2x increase in query access time over a single-client deployment. Importantly, this is a 2x speed-up over the state-of-the-art operating under impractical trusted proxy assumptions.

**Locality-Preserving Access Privacy.** One important measure, so far largely overlooked while evaluating access privacy mechanisms, is data locality. Due to caching effects at all levels of the memory hierarchy, it has long been understood that taking advantage of locality can have significant performance benefits. Notably, for example, a single cache miss overhead is more costly than executing 100 instructions. Disk seek overhead costs (e.g., time) often exceed 10000 times the bandwidth cost of reading a single sequential block from that disk.

Access privacy solutions however often rely on underlying access pattern randomization. Particularly, in the case of oblivious RAMs, the randomization necessary to ensure privacy seems to be in direct conflict with data locality. Even for a single access, a typical ORAM requires many non-sequential accesses to the untrusted data store. Even worse, the upper-layer (e.g., file systems) generating optimized accesses with high degree of locality to the underlying storage, gains no benefit when using a standard ORAM to interface with a physical store. This is because in ORAMs, the physical locations have no correlation with their logical addresses.

To overcome this, in [2], I proposed two new techniques for generic tree-based ORAM constructions: i) a locality-aware storage layout for efficient locality-preserving writes, and ii) a locality-sensitive block-placement strategy for locality-preserving reads. As a by-product, I also solved an orthogonal problem – efficient and secure synchronization in distributed settings where data is replicated across multiple ORAMs for fault tolerance.

These techniques are judiciously combined in a locality-preserving oblivious RAM construction, namely rORAM, optimized for sequential access (e.g. range query, file systems etc). Experimental results show that an rORAM prototype is
30-50x times faster than the state-of-the-art ORAM without locality-preserving access for similar range-query workloads on local HDDs, 30x faster for local SSDs, and 10x faster for network block devices. For real world workloads, rORAM is up to 5x faster running a file server and up to 11x faster running a range-query intensive video server workload compared to state-of-the-art traditional ORAMs. Further, these techniques can also speed-up traditional ORAMs by a factor of 2x.

In ongoing work [6], I am exploring how locality-preserving accesses can benefit write-only oblivious RAMs for applications where only write protection is required. Write-only access privacy has found use in several applications e.g., secure data backup and plausible deniability, and are generally more efficient that full ORAM constructions. While previous works optimize write operations, reads (which are often much more frequent than writes) are still expensive. Our new construction ensures locality-preserving accesses for both read and write operations, and performs significantly better for real-world workloads.

Access Privacy for Plausible Deniability. Plausible deniability allows a party to plausibly claim to a coercive adversary that a certain information is not in their possession. It constitutes a powerful tool against censorship and oppressive regimes. Unfortunately, existing solutions, such as (the now defunct) TrueCrypt, defend only against an adversary that can access a user’s device at most once (single-snapshot adversary). However, real adversaries often have “multi-snapshot” capabilities. For example, officers at border crossings can inspect a user’s device multiple times – saving and comparing the device snapshots during entries and departures. In fact, an oppressive government can easily collude with a hotel maid for cheap multi-snapshot capabilities. Obviously, a plausible deniability scheme should be resilient to such realistic externalities.

Recent solutions have traded significant performance overheads for the ability to defend against a powerful multi-snapshot adversary.

In [3], I first explored the relationship between access privacy and plausible deniability against strong multi-snapshot adversaries. I designed and implemented DataLair, an efficient block storage Linux kernel device with plausible deniability. A key component in DataLair is a new, efficient write-only oblivious RAM construction. DataLair is orders of magnitude faster than the state-of-the-art plausible deniability solution capable of defending against multi-snapshot adversaries, especially for read operations.

Subsequently, I jointly worked on the design of a locality-preserving access privacy mechanism suited for plausible deniability applications in [8]. A key insight here is that under a more relaxed practically-relevant security requirement, oblivious RAMs can be eliminated. Instead, for access privacy, we can use a canonical form that permits most of the writes to be done sequentially, thus increasing I/O throughput by orders of magnitude over existing work.

This mechanism is incorporated in PD-DM, a Linux kernel block device with strong plausible deniability. Due to the locality-preserving accesses, in typical setups, PD-DM throughputs are orders of magnitude (10 - 100x) faster than existing approaches.
In more recent work [7], I jointly designed the first plausibly-deniable file system that can defend against multi-snapshot adversaries and deny the existence of data as well as its own existence. This is an important property since even the existence of a special system to hide data will be suspicious to an adversary. The solution leverages special physical properties of NAND flash chips.

**Integrity-Preserving Storage.** Data integrity is another essential, often overlooked property in outsourced remote storage. This is significantly more critical for cloud services where the storage server can be malicious, compromised, or simply not as diligent in ensuring regulatory compliance or security best practices. In fact, secure hybrid cloud strategies based on Amazon’s Virtual Private Cloud (VPC) (which provides strong security guarantees through network isolation) store data on untrusted storage devices residing in the public cloud. For example, Amazon VPC file systems, object stores, and databases reside on virtual block devices in the Amazon Elastic Block Storage (EBS).

This allows numerous attack vectors for a malicious cloud provider/untrusted software running on the server. For instance, without integrity of the storage device, a malicious cloud storage service could remove logs of a coordinated attack. By determining the blocks that store the logs, a malicious cloud service can selectively replace these blocks with old, innocuous values. Such an attack on the logs can go unnoticed, potentially indefinitely.

While several custom file systems handle parts of this problem, they are not well suited for scenarios where users may prefer to deploy standard file systems on remote storage. Therefore, a block device level solution is more suitable for cloud storage scenarios. Currently available block device integrity solutions either ensure block-level integrity (dm-integrity) and are prone to replay attacks, or can only support read-only integrity verification (dm-verity).

In [4], I designed and implemented dm-x, the first Linux kernel block device that transparently interfaces with standard file systems and provides volume-level integrity for writable volumes.

## 2 Future Work

I thoroughly enjoy exploring this spot at the intersection of security, systems and theory. In the immediate future, I will continue to design and build practical secure systems for which I can prove strong security properties. Here are some of the projects I am working on.

**Oblivious data access.**

- **Multi-layered storage hierarchies.** To achieve really fast systems, it is essential to carefully consider and deeply understand the underlying technology (e.g., storage, DRAMs, NVRAMs) instead of blindly designing protocols. This is especially important as faster (and more expensive) storage is being developed (e.g., NVRAMs) and deployed in multi-layered hierarchies. I believe interesting constructs can be achieved by coupling effective and secure caching of data in the faster storage components with
lazy oblivious prefetching (and synchronizing) data from the slower components.

- **Application Semantics.** I want to explore efficient oblivious access system designs driven by application-specific semantics. Applications may require access privacy for only a subset of the data items. Further, the desired level of privacy for a particular data item can often be context-specific. The first task in analyzing this is identifying and defining privacy contexts, either through automated techniques (e.g., symbolic execution, static analysis) or explicit user hints. Subsequently, these context-specific privacy definitions can allow us to build systems where expensive components such as oblivious RAMs function as “secure enclaves”, holding only the data that requires privacy in the current context. This will dramatically speed-up overall application execution while also providing strong privacy guarantees on demand.

**Practical History Independence.** I also want to continue my work on important and related security primitives such as history independent data structures. History independence is essential in many scenarios involving sensitive data or actions (e.g., secure e-voting). In [1], I jointly proposed a theoretical framework for secure history independence and would like to continue my work in the context of two applications.

**Secure Analytics.** Another broad area of interest I intend to pursue further is the intersection of security, privacy, information retrieval and natural language processing. With an increasing number of cloud services now supporting end-to-end encryption, the availability of plaintext data for applying data analysis techniques is becoming limited. On the other hand, these techniques have proved very useful in increasing usability such as by ranking queries and returning more meaningful results. While generic cryptographic techniques like homomorphic encryption can alleviate the problem, they also impose significant overheads. I will collaborate with my colleagues in data science and explore tailor-made privacy-preserving information retrieval and NLP techniques that can be applied efficiently to encrypted data.

**Physical Properties and Untapped Information Channels.** Information channels that can be found in unexpected physical properties are very interesting and I will explore such bandwidth further, e.g., in the design of information hiding and plausible deniability systems. There are at least two independent ways to hide information in flash chips that are incredibly hard to detect. We leveraged one such technique to design a plausibly deniable filesystem [7]. Such opportunities certainly exist in other emerging technologies such as IoT devices. With misuse, these technologies can pose a threat to individual privacy and national security. This is especially concerning since these devices can enable new information channels that are either unknown or not considered viable. I intend to explore and analyze the IoT ecosystem in more detail for information hiding opportunities.
**Covert and Side Channels.** I will continue my initial explorations into covert and side channels in modern heterogeneous memory systems. Heterogeneous architecture is increasingly being adapted to build more efficient smart phones, tablets etc., where e.g., GPUs are used to speed-up calculations. While this design reduces the burden on the programmers, sharing memory between different components often comes with powerful covert and side channel attacks that need to be mitigated.

**Hardware-based Security.** After my internship at IBM Research, I have developed an interest in hardware-backed security (e.g., secure processor architectures, and secure virtualization etc.). At IBM Research, I have worked on an effective malware defense mechanism leveraging secure hardware-backed virtual machines. Instead of attempting to analyze and detect malware, we isolate the malware execution in a secure virtual machine which can be later used for forensics. A key advantage of this technique is protection against unknown malware/ransomware. This is ongoing work that I will expand on in the years to come.

**Secure Hardware Architecture.** At IBM Research, I also explored processor tracing technologies for debugging and control flow attestation. I will continue this work and collaborate with colleagues in architecture while also studying more to enhance my architecture skills.

Overall I find the prospect of research both exciting and rewarding as a career choice. Spawning and leading an idea from conceptualization to a real system build with provable security properties, is an enriching experience with exposure to all kinds of unknowns and new concepts to be explored or designed. So far, I have collaborated with researchers from many different areas and I intend to continue doing so.

Anrin Chakraborti

December, 2018

References


