Sujaya Maiyya — Research Statement

Individuals and enterprises produce over 2.5 exabytes (10^{18} bytes) of data everyday. Much of this data - including sensitive and private information - is stored with and managed by third parties, such as Amazon Web Services or Google Cloud. These companies can lose millions to billions of dollars in sales if their data access latencies increase by only a few hundred milliseconds [2]. Hence, reducing data access latency to improve performance received the highest priority while designing cloud data management systems. But the ever growing number and sophistication of cyber attacks on the cloud [3] coupled with increases in legal requirements for data privacy and security (e.g., GDPR or HIPAA) [1] have forced cloud providers to re-evaluate their priorities. However, there exists a fundamental trade off between security and efficiency in data management systems. While resolving this tension is challenging, it has **fostered the growth of a deep field at the intersection of cryptography and database research**. In Springer, for example, the number of published papers in this intersection grew by 50.1% since 2015 and 91.8% since 2000[1].

My PhD research encompasses **designing, prototyping, and evaluating data management protocols that strike a balance between efficiency and security in both trusted and untrusted environments**. Before being able to solve security challenges in database systems, I first had to fully understand existing system designs. My extensive study of distributed data management, where data is partitioned and/or replicated across geo-distributed locations, led me to identify open problems in traditional, trusted cloud databases. The systems I built in trusted clouds perform 16x-18x better than the state of the art [6, 9]. As the next step towards building secure data systems, I developed a fault-tolerant data system for hybrid clouds that consist of a small trusted private cloud and a larger untrusted public cloud [5]. This system tolerates malicious failures in the public cloud and crash failures in the private cloud, and by leveraging the locality of crash and malicious failures, our system performs 37.5% better in throughput than the state-of-the-art baseline systems. Finally, to better understand the interplay between various security guarantees and performance, I conducted a tutorial on blockchain systems [11] – an ideal example where untrusted geo-distributed entities manage critical data. Equipped with blockchain techniques that protect data, I built secure database systems that guarantee data integrity [8] and data privacy [7, 10]. *I developed the first distributed transaction commitment protocol [8] that tolerates up to $n-1$ maliciously failing servers (out of n servers) without using expensive data replication and the first fault tolerant and highly available oblivious datastore [10] – one that hides access patterns of users along with the contents of data.*

Moving forward, **my research vision is to bring feature-rich databases that are secure and high performing into reality**. Many existing oblivious database systems trade useful database features such as concurrency control or query optimizations in favor of performance; this leads to a limited backend database, which supports only simple Get and Put operations. Building feature-rich private databases that are also practically usable requires experience designing database features in trusted settings, and the ability to identify security weaknesses and choose the best cryptographic tool to transform a non-private feature into a private one. While I have already established my ability to develop private and secure database features such as transaction commitment [8] and fault tolerance [10], I am interested in furthering this research area by creating private versions of other features such as concurrency control and query optimization. Having demonstrated a track record of building high performing secure databases, I am the ideal candidate to realize the vision of integrating vital features into these secure systems.

Trusted Infrastructure

Before delving into untrusted settings, I started my research career in traditional cloud settings, which assume trust. Cloud enterprises typically *replicate* their data to provide fault-tolerance and *shard* (or partition) their data and store the shards on multiple servers to provide scalability. State of the art databases, such as Google's Spanner, use atomic commit protocols (e.g., Two Phase Commit) to allow scalability and consensus protocols (e.g., Paxos) to achieve replication. Spanner treats atomic commitment and consensus disjointedly, where it hides the consensus logic from commitment logic and vice versa. Motivated by the co-existence of sharding and replication in most real-world databases, our work [9] unifies the two seemingly disparate paradigms into a single framework called *Consensus and Commitment* (C&C). The C&C framework is a great pedagogical tool as it can model many established data management protocols, while also providing insights to propose new ones. To highlight its advantages, we propose a novel commit protocol, *G-PAC*, which merges consensus with commitment. As a key feature, unlike many existing protocols that separate failure recovery logic from failure free execution logic, G-PAC executes the same set of instructions for both scenarios, easing a developer's job. The unified approach of G-PAC reduces one (out of three) round of cross-datacenter communication compared with Google's Spanner; this allows **G-PAC to perform between 27-88% better than Spanner-like protocols in terms of throughput**.

While G-PAC extended the C&C framework to the context of distributed transactions for generalized workloads, we developed Samya [6] to extend the C&C framework to manage *high contention, hotspot* data. Samya, a geo-distributed data management system, stores and manages hotspot aggregate data (e.g., number of cloud resources or number of items in stock).

[1] I searched for conference papers with keywords database AND security OR privacy OR encryption on https://link.springer.com/search.

State-of-the-art geo-distributed databases such as Google's Spanner take a centralized approach where a server acting as a leader processes all client requests to a specific data item sequentially, which thwarts throughput and leaves the data replicas underutilized. To utilize all replicas or sites that store a data item, Samya splits or partitions the aggregate data, modeled as tokens (e.g., resource tokens), and stores individual partitions on different servers. This design choice allows servers in Samya to independently and concurrently serve client requests. A site serves requests locally as long as it has locally available tokens. Samya employs predictive models to predict if a site will exhaust its tokens; if so, the sites execute a novel synchronization protocol, Avantan, extended from the abstractions of the C&C framework, to redistribute the spare tokens in the system. Compared with the centralized solution, **Samya's parallelism reduces the 99th percentile latency by 76% and allows Samya to commit 16x to 18x more transactions**.

Hybrid Infrastructure

To accomplish my goal of building trusted databases using untrusted infrastructure, I started examining *hybrid* cloud infrastructures encompassing a small trusted private cloud and a larger untrusted public cloud – a logical middle-ground between trusted and untrusted settings commonly deployed today. The default protocol choice to replicate data in such hybrid clouds has been PBFT, a Byzantine fault-tolerance protocol, which does not distinguish between crash or malicious failures and is unaware of the locality of the failures. In our work, SeeMoRe [5], we propose replication protocols for hybrid cloud environments that leverage the trust of a private cloud and the scalability of a public cloud to achieve better performance compared with PBFT. SeeMoRe tolerates crash failures in the private cloud and malicious failures in the public cloud. This work highlights how we can develop efficient protocols by being aware of where different types of failures (both crash or malicious) occur in hybrid cloud environments. Specifically, compared with PBFT, SeeMoRe reduces the number of communication phases and messages exchanged and also requires fewer replicas to tolerate a given number of malicious and crash failures, reducing an application's overall deployment cost. We developed SeeMoRe to dynamically switch between three different modes of operation depending upon the load on the private cloud: higher load on the private cloud leads to availing more servers from the public cloud and vice versa. Our extensive evaluation revealed that **SeeMoRe performs 37.5% better than PBFT at peak throughput and only 8% worse than the state of the art protocols that tolerate only crash failures**.

Untrusted Infrastructure

After working with fully trusted infrastructures and hybrid environments, I became intrigued by the question, *"What if I host my data on completely untrusted infrastructure?"*. To answer this question, we propose *Trust-Free Commit* (TFCommit), a trust-free commitment protocol that executes transactions across multiple untrusted servers [8]. To our knowledge, ***TFCommit is the first atomic commitment protocol to handle malicious failures without using expensive Byzantine replication.*** Byzantine replication protocols tolerate less that $n/3$ malicious failures, where n represents the total number of servers, whereas TFCommit tolerates up to $n-1$ malicious failures. TFCommit combines Two Phase Commit with a collective signature scheme, CoSi, to commit transactions across multiple untrusted servers. Similar to blockchain, committing each transaction (or a set of transactions) produces a tamper-resistant log entry, which each server appends to its log. The tamper-resistance stems from collective signature created during commitment that cannot be modified by a single server. An external auditor then audits this log to detect any faulty behavior. TFCommit's failure detection mechanism precisely identifies the point in the execution history at which a fault occurred as well as the servers that failed. These guarantees provide two fold benefits: (i) An auditor always detects both a malicious fault and the misbehaving database server, and (ii) A benign server can always defend itself against false accusations. Many scenarios can benefit from TFCommit, such as blockchain systems or supply-chain management where the participating entities span different administrative domains that do not trust each other. Compared to executing a transaction in trusted infrastructures, **executing TFCommit has only 1.8x overhead in latency and 2.1x in throughput - an acceptable overhead for many use cases given the additional security guarantees of TFCommit**.

While TFCommit ensured data integrity, the growing number of data leaks in the recent past inspired me to work on data privacy. Recent attacks revealed the inefficiency of mere data encryption to protect data privacy; an attacker can uncover non-trivial information either about the data or its users by observing the users' access patterns. Oblivious RAM, or ORAM, is one of the cryptographic techniques that prevents access pattern attacks. Although the database literature consists of many ORAM-based systems, to-date no ORAM-based datastore supports fault tolerance – an important primitive in database systems. To this end, our work, *QuORAM*, proposes a quorum based replicated ORAM datastore that tolerates crash failures while preserving obliviousness [10]. QuORAM consists of multiple untrusted and potentially colluding storage servers, each accessed via a separate trusted proxy. QuORAM guarantees linearizable semantics – all operations on a data item appear to be linear – using *a lock-free replication protocol* where a client always reads from a quorum (e.g., majority) of replicas and writes to the same quorum, for both read and write requests. If proxies treat the reads and writes as separate ORAM requests, then they fetch the path twice sequentially from the storage server, which leads to prohibitive latencies. To avoid double-fetching, proxies in QuORAM maintain an *incompleteMap* to store request mappings of requests that are read but not yet written back. Our evaluations of QuORAM reveal the advantages of geo-replication in ORAM systems: reduced latency for geo-distributed clients and reduced load on a single proxy. **QuORAM reduces the average data access latency by 54.4% and improves the throughput by 51.2% compared to a non-replicated ORAM system, while providing fault tolerance**.

Future research interests

The use of smart devices – wearable health monitors, home assistants, bio-metric security appliances – is on the rise. These smart devices typically communicate with the cloud, either directly or indirectly, to store and retrieve an end user's data. Such outsourcing of data necessitates building trustworthy data management systems where the data can reside atop of potentially untrusted infrastructure. While trusted cloud infrastructures are feature-rich and relatively mature, advanced features in untrusted cloud infrastructure remains a burgeoning technology that places a high burden on application developers and has expensive deployment costs. For my future research, I will continue to design efficient and secure protocols and systems to manage data, and pioneer new research on building cost effective and feature-rich private databases catering to specialized domains such as Internet of Things (IoT) devices. Specifically, I have three focus areas for my future research:

Serverless Paradigms Serverless computing is a paradigm where the cloud provider dynamically allocates and manages resources, allowing a developer to focus solely on application development. Serverless computing executes computations in bursts without retaining any execution state in memory for long. Many applications find it appealing due to its cost effectiveness and low overhead for the developers. I envision many existing applications will switch to serverless paradigms, and in fact, recent data forecasts the serverless market to grow from $7.6 Billion in 2020 to $21.1 Billion by 2026 [4]. Assuming trusted settings, existing transactional applications cannot be easily migrated due to the inherent stateless nature of serverless architectures. Transactional workloads require maintaining state, typically distributed across servers, and the servers that maintain the state need to coordinate to ensure the state's consistency. While some recent works provide transactional consistency and isolation in serverless settings, the space of ACID-compliant (Atomicity, Consistency, Isolation, and Durability) datastores remains widely unexplored.

I have started exploring ways to incorporate ACID guarantees in a serverless platform. As a research intern at IBM Research, I worked on building a transactional framework on top of a serverless runtime, KAR[2], developed at IBM. The transactional framework[3] provides **a**tomicity, strong **c**onsistency, serializable **i**solation, and **d**urability (ACID) guarantees. While analyzing the performance of KAR's transactional framework using the TPC-C benchmark, I realized that serializable and strongly consistent data impose high performance costs, indicating a serverless approach may not be correct for these data types. In my future research, I plan to answer the question: *what consistency and isolation guarantees best suit serverless architectures*? Specifically, I plan to develop an ACID-compliant transactional data system with snapshot isolation (an isolation guarantee more relaxed than serializablity) and eventual consistency (weaker than strong consistency) and compare the performances with their stricter counterparts. Both snapshot isolation and eventual consistency are practical guarantees that database systems such as Amazon's DynamoDB and Microsoft's CosmosDB provide.

Bridging the Functionality vs. Security Gaps in Oblivious Datastores Database researchers, and people in computing in general, are grappling with the increasing challenge of data privacy. Laws such as GDPR and HIPAA make data privacy not merely a desired property but a necessary property that database systems must provide. Merely encrypted data is susceptible to privacy attacks based on access patterns, and Oblivious RAM or ORAM, a cryptographic technique, protects data from access pattern attacks. Although database systems have evolved significantly over the years to provide a rich set of features such as concurrency control, transactional ACID guarantees, and query optimizations, almost all currently existing privacy preserving oblivious datastores strip away these features and downgrade the untrusted backend server to a simple system that supports single item Gets and Puts. This simplifying of the backend database servers is usually necessary to uphold the strict definitions of obliviousness, as different database features may reveal non-trivial information about the data. For example: allowing the untrusted database server to handle concurrency control may reveal high contention in application workloads, as multiple concurrent requests might be accessing the same data item. Today, a trusted proxy server provides much of these features such as concurrency control and the backend server merely Gets and Puts individual data items.

I am interested in developing feature-rich databases, where the features themselves preserve privacy, and to remove the trusted proxy, allowing the clients to access the database servers directly. To achieve this, I aim to consider each database feature separately and build oblivious versions of those features. For example, concurrency control can be made oblivious by creating fake accesses to data items and mixing them with real access such that from the server's perspective, all data items have equal concurrency. I believe such a feature-by-feature approach to integrate obliviousness into database systems can help bridge the gaps between theoretical oblivious datastore constructions and existing practical databases.

Fault-tolerance and Security of IoT data The number of IoT devices in use is speculated to reach 27 billion by 2025, while we use 12.3 billion IoT devices today. Users find IoT devices highly appealing as they allow them to access a myriad of applications and services with a single touch, such as paying with a tap of a smartwatch or phone. Since IoT devices can span

[2] https://github.com/IBM/kar

[3] https://github.com/IBM/kar/tree/main/examples/actors-transactions

sensitive domains such as health monitoring or military systems, data produced by certain IoT devices must be fault tolerant and the entire data collection and processing pipeline must be secure, without overly degrading performance.

Edge computing or edge cloud, a relatively new paradigm, enables quick storage and compute access to edge devices such as IoT devices. Typically, edge devices push their data to *edge nodes* – servers physically close to edge devices – and these nodes then push the data to a persistent and fault-tolerant cloud storage. While the cloud has fault tolerant storage, the edge nodes generally do not. Fault-tolerance techniques such as data replication using Paxos are unfit for edge cloud due to their large communication overheads. As a future research goal, I will develop novel fault tolerance techniques for edge cloud and compare them with state of the art techniques such as replication using Paxos. Specifically, I aim to explore eventually consistent fault tolerance techniques where the main thread of a primary node continues to process client updates and a background thread updates the secondary (and potentially other) node(s).

With regard to privacy and security of IoT data, the literature consists of many works on securing the communication layer (i.e., networks) and solutions to encrypt the IoT data for privacy. In the data processing pipeline of IoT devices, encrypted data eventually migrates to the cloud. As described earlier, mere data encryption is insufficient to achieve data privacy due to access pattern attacks. Presently, no current work has studied the cloud storage access patters unique to edge devices. Existing solutions to mitigate access pattern attacks, such as ORAM, are impractical for IoT devices due to their prohibitive latencies. In the future, my research group will study the access patterns unique to IoT/edge devices, identify specific attacks plausible due to such patterns, and devise novel solutions to mitigate access pattern attacks while retaining the low access latency necessary for IoT devices.

Research Funding

I understand the importance of raising funds to support my future students and to materialize my future research goals. I showcased my ability to obtain funding for my research by receiving two fellowships to fund my PhD: IBM PhD Fellowship totalling $95,000 over 2 years and Google PhD Fellowship (declined) with net worth $159,000 over 3 years. I also secured funded conference travels to attend Rising Stars (awarded to selected women PhD students and postdoctoral scholars interested in academia), CRA-W Women's Grad Cohort, Grace Hopper Conference, and a top database conference, VLDB (2018). At UCSB, I contributed to writing a NSF CNS grants that partially funded my PhD. As a faculty, I plan to submit grant applications to NSF CNS, IIS as well as OAC for my projects on data security and privacy. I have also been forging industry connections with IBM, Facebook, and Google, and I plan to apply for industrial faculty research or early career awards.

References

[1] Growth of Global Data Privacy Laws.
`https://stealthbits.com/blog/growth-of-global-data-privacy-laws/`. Accessed: 2021-10-25.

[2] Impact of slow page load time.
`https://medium.com/@vikigreen/impact-of-slow-page-load-time-on-website-performance-40d5c9ce568a`. Accessed: 2021-06-6.

[3] Increasing Cyber-attacks on Cloud Services.
`https://compliancy-group.com/cyber-attacks-on-cloud-services-rise-630/`. Accessed: 2021-10-25.

[4] Serverless Architecture Market Value.
`https://www.marketsandmarkets.com/PressReleases/serverless-architecture.asp`. Accessed: 2021-10-25.

[5] M. J. Amiri, **Sujaya Maiyya**, D. Agrawal, and A. El Abbadi. Seemore: A fault-tolerant protocol for hybrid cloud environments. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1345–1356. IEEE, 2020.

[6] **Sujaya Maiyya**, I. Ahmad, D. Agrawal, and A. El Abbadi. Samya: A geo-distributed data system for high contention aggregate data. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 1440–1451. IEEE, 2021.

[7] **Sujaya Maiyya**, P. Ananth, D. Agrawal, and A. El Abbadi. ORTOA: One Round Trip Oblivious Access *(In progress)*.
`https://sites.cs.ucsb.edu/~sujaya_maiyya/assets/papers/ORTOA.pdf`. Accessed: 2021-11-11.

[8] **Sujaya Maiyya**, D. H. B. Cho, D. Agrawal, and A. El Abbadi. Fides: Managing data on untrusted infrastructure. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 344–354. IEEE, 2020.

[9] **Sujaya Maiyya**, F. Nawab, D. Agrawal, and A. E. Abbadi. Unifying consensus and atomic commitment for effective cloud data management. *Proceedings of the VLDB Endowment*, 12(5):611–623, 2019.

[10] **Sujaya Maiyya**, C. Scarberry, S. Ibrahim, D. Agrawal, A. El Abbadi, H. Lin, and S. Tessaro. QuORAM: A Quorum-based Replicated ORAM Datastore *(In progress)*.
`https://sites.cs.ucsb.edu/~sujaya_maiyya/assets/papers/QuORAM.pdf`. Accessed: 2021-11-11.

[11] **Sujaya Maiyya**, V. Zakhary, D. Agrawal, and A. E. Abbadi. Database and distributed computing fundamentals for scalable, fault-tolerant, and consistent maintenance of blockchains. *Proceedings of the VLDB Endowment*, 11(12), 2018.