Statement of Career Goals

Elaine (Runting) Shi

1 Research Statement

My research focuses on designing secure and privacy-preserving algorithms and systems, with a focus on distributed computing environments such as cloud computing and blockchains. Specifically, I design and build systems that are not only provably secure, but also easy to program by non-experts. To enable this goal, I blend system building, design of new cryptography and randomized algorithms, as well as programming language techniques. For example, as I will explain in more detail later, my research on Oblivious RAM has not only resolved long-standing open questions and closed the theoretical gap in our understanding, but also resulted in simple and practical constructions that have had impact in multiple research communities including security, cryptography, programming languages, architecture, and algorithms. My recent research on decentralized cryptocurrencies has helped to build a new theoretical foundation for large-scale distributed consensus, and influenced the design of major cryptocurrency systems such as Ethereum.

1.1 Oblivious Computation

Imagine that a client would like to outsource sensitive, encrypted data to an untrusted cloud server. Even though the data is encrypted, it is well-known that access patterns to the data alone can leak sensitive information[1]. For example, a recent attack showed that even by leveraging coarse-grained access pattern leakage, one can reconstruct sensitive images being processed by widely-deployed application libraries [33].

Oblivious RAM (ORAM), first proposed by Goldreich and Ostrovsky [12, 13], is a cryptographic technique that provably obfuscates a program's access patterns to sensitive data. Shortly put, ORAM allows us to "encrypt" access patterns of a program. This might seem magical at first sight — how can one "encrypt" the side effects of a program? Amazingly, Goldreich and Ostrovsky showed that ORAM schemes can be constructed incurring only polylogarithmic overhead; and typically ORAM algorithms work by permuting the data in memory and periodically reshuffling the data as they are being accessed.

Simple and practical ORAM algorithms. Despite Goldreich and Ostrovsky's ground-breaking contribution, their original construction would have incurred prohibitive concrete overhead. In the next 30 years, a few works improved Goldreich and Ostrovsky's schemes but all these algorithms follow Goldreich and Ostrovsky's original algorithmic framework, resulting in complex constructions with large concrete overhead.

In 2011, I devised a new, tree-based paradigm for constructing ORAM [29]. This new paradigm departs from Goldreich and Ostrovsky's approach and results in conceptually simple constructions without relying on any cryptographic assumptions. In subsequent works, my collaborators and I further improved our tree-based ORAM algorithms, resulting in the Path ORAM [31] and Circuit ORAM [32] algorithms — the former remains the scheme of choice for cloud outsourcing and secure processors whereas the latter remains the scheme of choice for secure multi-party computation. Our schemes are 10^5x to 10^6x faster than the original constructions by Goldreich and Ostrovsky in terms of concrete performance.

Because our work provided the much needed progress in an area that had been somewhat stagnant, it helped to revive the community's interest in ORAM. Since then, our schemes have been implemented in

[1] Here access patterns refer to the sequence of addresses accessed, *not* including the data contents.

various application settings, and have had impact in multiple research communities, including the security, cryptography, programming languages, architecture, and database communities. Notably, hardware architects at MIT implemented a variant of Path ORAM in trusted hardware and taped out a secure processor called Ascend running an ORAM controller [27]. They were able to show an average of 2x performance overhead on standard benchmarks (assuming that ORAM is applied only to sensitive parts of the computation). Besides Ascend, our ORAM schemes have been implemented in the context of secure cloud computing [6, 20], secure databases [7, 11], blockchains, and other domains. Moreover, our ORAM constructions have inspired works [9, 18, 21] that developed type systems for verifying a program's memory-trace oblivious properties (including our own works [18, 21]). Numerous open source projects have implemented our ORAM schemes, including our own and those of others. Our ORAM schemes have also been taught in various universities in undergraduate and graduate-level courses.

Theoretical significance. Our research on ORAM has resolved several long-standing open questions. For more than 30 years, the biggest open question in this line of work is whether there exists an optimal ORAM scheme matching the $\log n$ lower bound first proven by Goldreich and Ostrovsky. Our work has closed this gap in our understanding. In a sequence of works we showed optimal ORAM constructions: specifically Path ORAM and Circuit ORAM achieved optimal (i.e., logarithmic) bandwidth overhead assuming that memory blocks are reasonably large. Very recently, with my students and collaborators, we showed, in a work called OptORAMa [1], how to remove the assumption on memory block size and achieve optimality not only in terms of bandwidth but also in terms of runtime.

Interestingly, our most recent OptORAMa construction is based on non-trivial algorithmic techniques intimately connected to the classical algorithms line of work on sorting. In fact, developing these techniques a little further, we can solve additional algorithms challenges that have long been open. Recall that the famous AKS work showed that we can construct sorting circuits of $O(nw \log n)$ size for sorting n elements each w bits long. We were able to show a non-trivial, *non-comparison-based* generalization of this classical result [2]: we show that sorting n elements each with a k-bit key and a $(w - k)$-bit payload requires a circuit only of $O(nwk)$ boolean gates (ignoring $\mathsf{poly} \log^*$ terms) which asymptotically outperforms AKS if $k = o(\log n)$. Due to the 0-1 principle for sorting (see Knuth's book [15]), it is long known that a result of such nature is impossible in the comparison-based model even if $k = 1$ — for this reason our construction in fact fundamentally departs from AKS and we are the first to show non-trivial results using non-comparison-based techniques in a circuit model of computation.

1.2 Blockchains and Cryptocurrencies

I got fascinated by blockchains and cryptocurrencies back in 2011, and I did some early work that helped to bring cryptocurrencies to the attention of the scientific community [3]. Since then, blockchains and cryptocurrencies have been a primary area of my research. In the past few years, my work has focused on building a new theoretical foundation as well as practical constructions for large-scale distributed consensus (also called blockchains). The goal of consensus is to allow distributed nodes to reach agreement on an ever-growing, linearly-ordered log of transactions, thus enabling a public ledger for cryptocurrencies. Although distributed consensus is a 30-year-old problem, decentralized environments such as Bitcoin imposed new challenges and pushed us to fundamentally rethink consensus. There are at least three significant challenges, *scalability*, *incentives*, and *new theoretical foundations*.

Scalability. In a decentralized environment, we would like to scale up blockchain constructions to thousands or even millions of nodes, and yet be able to provide fast confirmation and high throughput. It is well-understood that the Nakamoto consensus protocol that underlies Bitcoin is unlikely the longer-term desired solution due to its slowness, low throughput, and energy waste.

With my collaborators, we showed how to remove the proof-of-work used in Nakamoto's consensus, and

replace it with proof-of-stake type assumptions. In proof-of-stake, a player's voting power is proportional to its stake in the system. Our work called Snow White [8,25] provided one of the first provably secure proof-of-stake constructions. Essentially we showed how to get rid of the proof-of-work in Nakamoto's consensus while preserving its stochastic properties. Elements of our design were later adopted by Ethereum's proof-of-stake design and (later versions of) Algorand's consensus protocol.

In a sequence of works culminating in the Thunderella protocol [26], my collaborator and I showed how to combine techniques from classical consensus and modern blockchains to achieve scalable consensus in a permissionless environment, and moreover our construction is conceptually simple which makes it desirable for practical deployment. I then helped to co-found a company and worked with the engineers to implement and open source our consensus protocol.

Thunderella also suggests a theoretically novel approach for making *synchronous* consensus fast. Classically most synchronous consensus protocols are considered slow because the protocol's performance typically depends on an a-priori determined network delay parameter. For example, if the network's average delay is expected to be 1 second, one might want to conservatively set this delay parameter to be 10 seconds — since consistency can be violated if the network violates synchrony assumptions. We say that a protocol is *responsive* if it confirms transactions as fast as the network makes progress, independent of any a-priori configured delay parameter. Classically, most partially synchronous or asynchronous protocols are responsive; unfortunately they can defend only against 1/3 corruptions due to a well-known lower bound by Dwork et al. [10]. In Thunderella we proposed a new notion of performance called *optimistic responsiveness* which aims to achieve responsiveness almost all the time in practice, without being subject to the 1/3 lower bound pertaining to partial synchrony or asynchrony. For example, as one typical instantiation of the Thunderella paradigm, we have a protocol that guarantees consistency and (slower) liveness as long as the majority of nodes are honest, but assuming that a designated leader and 3/4 of the nodes are honest and online, the protocol confirms transactions at raw network speed.

Incentives. Classically consensus protocols are often deployed by a single organization (e.g., Google or Facebook) in a small-scale and closed environment. The primary purpose there was fault tolerance, and incentive for participation was a non-issue. Incentives, however, arise as an exciting challenge in a decentralized environment. Since participation in the protocol incurs cost, the protocol must provide rewards to incentivize players to participate. Not only so, since all players are selfish and mutually distrustful, even "honest majority", an assumption that we classically took for granted in distributed systems and cryptography, should no longer be taken for granted in a decentralized environment. Instead, we would like to design protocols that not only incentivize participation but also incentivize honest participation.

A high-profile and oft-debated incentive attack in Bitcoin is called the selfish-mining attack: it was shown that with Nakamoto's consensus protocol, a 1/3 coalition can deviate from the honest protocol in a specific way to gain about 1/2 of the block rewards (assuming that the adversary also has a certain but limited degree of control over the network routers). In a work called Fruitchain [24], my collaborator and I showed how to add a small tweak to Nakamoto's consensus protocol to provably defend against the selfish mining attack, resulting in a game theoretically secure blockchain construction. We proved that our protocol satisfies an approximate, collision-resistant Nash equilibrium. With such a game theoretically secure blockchain, players, upon seeing others being honest, will not have incentives to deviate — thus honest participation is an equilibrium state.

New theoretical foundations. In a few recent works [14,25], we showed that classical modeling techniques are insufficient for capturing the robustness requirements of distributed consensus in a decentralized environment. Specifically, classical modeling techniques are typically draconian and assume that honest nodes participate in the protocol from beginning to end. In a decentralized environment, however, nodes may come and go, join the protocol late, or suffer from temporary outages and later want to rejoin the protocol. Our recent works [14,25] explored theoretically how to model such sporadic participation in a decentralized

environment, and proved several lower- and upper-bounds.

1.3 Designing Programming Languages for Cryptography

As cryptography becomes more important in practical applications such as decentralized cryptocurrencies, an important challenge is to design new programming languages that help ordinary developers (who are not necessarily cryptography experts) develop secure protocols and systems.

Cryptographic constructions for multi-party computation and zero-knowledge proofs often adopt the circuit model of computation. Real-world programs, however, are written for a RAM model which fits the prevalent von Neumann architecture. It is not only unnatural and time-consuming for ordinary programmers to encode computation tasks as circuits, they also often end up constructing sub-optimal circuits.

With my students and collaborators, we have built and open sourced the ObliVM framework [19] (for multi-party computation) and the xjSNARK framework [16] (for zero-knowledge proofs) that allow a programmer to express computation tasks as normal programs; we then adopt an optimizing compiler to convert it to a circuit or a sequence of circuits. The enabling algorithmic technique for this RAM-to-circuit conversion is exactly Oblivious RAM (ORAM); however brute-force compilation of a program using ORAM will result in high overhead. Instead, we rely on static analysis to partition the program's data into parts whose access patterns leak information and parts whose access patterns are safe to reveal — our compiler places only the former type of data in ORAMs. Furthermore, we take an algorithms-compiler co-design approach: we observe that for several common programming abstractions such as MapReduce, GraphLab, and inductive data structures, there exist asymptotically more efficient oblivious algorithms than generic ORAM. Therefore, we introduced these constructs (and several others) into our language design and compile them to the efficient oblivious algorithms rather than generic ORAM.

A surprising outcome of our research is that when my graduate students were using our ObliVM framework to implement secure computation for certain graph algorithms, we realized that the resulting implementation led to asymptotically better oblivious graph algorithms than what was theoretically known — we accidentally discovered novel algorithms while building ObliVM.

Finally, we also developed novel, memory-trace oblivious type systems that can type check the source or the target programs and guarantee that they indeed satisfy memory-trace obliviousness. Our work as well as efforts by others in the community have helped to shape subsequent government-funded research programs that called for more synergy between cryptography and programming languages (e.g., the ongoing IARPA HECTOR program).

1.4 Other Directions

Trusted hardware. Besides the directions mentioned before, I have also worked on trusted hardware and remote attested execution. During my Ph.D. I worked on a project called Binding Instruction aNd Data (BIND) which was among the first to suggest how to use trusted hardware to provide *verified computation* on an untrusted server. Our BIND work [30] inspired and influenced several follow-up works including Flicker [23] and TrustVisor [22]. I have also worked on the design of secure processors that enforce oblivious computations [17, 21], thus resolving certain high-profile side-channel attacks pertaining to commodity secure processors such as SGX.

Applied cryptography and privacy. I have worked on various directions in cryptography and privacy, including multi-party computation, game theory, and encryption systems with controlled disclosure (often called predicate encryption and functional encryption). Our work on practical differentially private data aggregation [5, 28] later *inspired the development of privacy-preserving federated learning systems at Google*.

1.5 Ongoing and Future Work

In ongoing and future work, I am excited about the following directions:

Alternative notions of access pattern privacy. Although our ORAM constructions are simple and practically efficient, we cannot avoid the logarithmic lower bound. In a recent work, my collaborators and I suggested a new, relaxed notion of access pattern privacy called differential obliviousness [4], i.e., we require that the access patterns of the program satisfy (approximate) differential privacy. Our work showed that for some computational tasks, there exist differential oblivious algorithms that incur only $\log \log n$ overhead where n is the data size; we also showed that for these specific tasks there is a logarithmic barrier if full obliviousness is required. This work helps to lay the groundwork for the study of differential obliviousness: as we point out in the paper, there are in fact more questions that we do not understand than what we understand regarding this new notion, e.g., can we generalize our results to a broader class of useful algorithms? Can we make our current theoretical constructions practical? What are other alternative and meaningful notions of access pattern privacy? I am excited to explore this general direction, and hopefully our efforts and those of others can allow access pattern privacy techniques to have wide-spread adoption.

Incentives and mechanism design for cryptocurrencies. Our own recent efforts and those of others have led to scalable large-scale consensus protocols. However, the community still does not understand how to formally reason about incentive mechanisms in blockchains and cryptocurrencies; in fact, more often than not, we do not even understand how to model incentives questions in these systems. Many questions can be asked: how to model and design fee mechanisms, how to disincentivize free-riding, how to reason about the economic robustness of cryptocurrency designs in terms of the cost of attacking these systems. I believe that this is a very exciting direction that requires more synergy and conversations between the economics, cryptography, and distributed systems communities.

Programming languages for cryptography. Despite our prior works and efforts of others, a lot more work is needed for "programming languages for cryptography" to be practical and widely-adopted. I have been working with my PL collaborators on using distributed information-flow techniques to allow a programmer to express the functional and security requirements of a distributed program (e.g., a smart contract protocol involving a blockchain) without worrying about which cryptographic primitives to use. We then rely on program partitioning techniques to synthesize distributed cryptographic protocols that can select among a set of available primitives including but not limited to commitments, multi-party computation, and zero-knowledge proofs. These primitives will be composed and woven together to realize the source specification. We will work on proving the compilation secure, and moreover we would like to express the security of the compiled program in cryptographers' language of Universal Composition (UC).

2 Teaching Statement

As an instructor, I believe in not just transferring knowledge, but also in stimulating the students' interest and passion in learning, as well as encouraging them to ask questions and to think creatively and critically about any new problem they might encounter.

Undergraduate/master-level teaching. I am excited about teaching both systems and theory courses at the undergraduate and master levels. I have taught undergraduate-level Computer and Network Security (CMCS 414), Cryptography (CS 4830/5830), and Introduction to Analysis of Algorithms (CS 4820).

Recently I have developed a keen interest in scaling up the impact of my undergraduate teaching. Partly this new interest stems from my latest experience in teaching the algorithms course at Cornell which had about 200 students. Because of Covid-19, we switched to online teaching in the middle, and I used this opportunity to develop techniques for virtual instruction. Using a combination of technologies including

instant messaging, online whiteboard, and video conferencing, we supported both synchronous online lectures and asynchronous viewing for students in different time zones. We developed methods for simulating i-Clicker experiences in remote instruction, and for encouraging students to interact and ask questions during synchronous online lectures. We had offline quizzes after each lecture, to help make sure that even students who are not in the same time zone stay engaged. It was a most rewarding experience for me when my students expressed appreciation for our efforts.

This experience also prompted me to think how to allow our undergraduate education to have greater impact and outreach, e.g., through a combination of online lectures while providing the remote students an opportunity to interact with the instructor and course staff.

In the near future, I would love to continue to teach large undergraduate- and master-level courses, either in systems or theory. In the medium term, I would like to make course videos and lecture notes available online, allowing greater outreach and impact. I am also excited to create and run online courses (potentially as a remote extension of a local course at CMU), where we offer not just remote lectures, but also an interactive experience between the students and course staff, as well as among the students themselves.

Graduate-level courses. In the past, I have designed several new graduate-level classes, including "Privacy-Enhancing Technologies" (Fall 2012), "Securing and Monetizing the Internet" (Spring 2013), "Designing Secure Systems with Cryptography" (Fall 2013), "Science of Crypto-Currency" (Spring 2015), and "Blockchains and Distributed Consensus" (Fall 2017). Among these, two are extra overload courses that I volunteered to teach. Like my research, my graduate-level courses often have a blend of theory and systems.

Open-source course materials and textbooks. In the past, I have created the first open-source course materials on smart contract programming (`http://mc2-umd.github.io/ethereumlab/`), as well as open-source course modules on applied cryptography (`https://pathoram.jimdo.com/`). In the near term, I plan to write a textbook on distributed consensus and blockchains; in the medium term, I would like to write a textbook on applied cryptography.

3 Mentoring and Diversity Statement

To me, the most rewarding part of being a professor is mentoring students and helping them succeed. I would like to train a next generation of researchers who can bridge theory and practice, and who can innovate and make an impact across disciplines. To this end, I have created a unique research group where security, cryptography, and programming languages come together. Several of my students and postdocs have continued onto tenure-track positions at institutions such as UIUC, Duke, USC, NJIT, Hebrew University, University of Utah, Illinois Institute of Technology, and so on (see my CV for details).

Last but not the least, I am eager to help promote and retain underrepresented minority groups in computer science. In the past few years I served on the committee of the Scholarships for Women Studying Information Security (SWSIS) under CRA-WP. I was a co-advisor for the Association of Women for Computing (AWC) back at Maryland, where I helped to organize lunches with successful women role models, and to recruit female graduate students. I have also mentored numerous female students at all levels. I have been invited to speak with female graduate students at other universities and at workshops that promote women in computer science. In the future, I would like to organize workshops for female undergraduate students (e.g., co-located with a major venue in security), to encourage them to apply to graduate school and explore security and cryptography as a research direction.

References

[1] G. Asharov, I. Komargodski, W.-K. Lin, K. Nayak, E. Peserico, and E. Shi. OptORAMa: Optimal oblivious RAM. In *Eurocrypt*, 2020.

[2] G. Asharov, W.-K. Lin, and E. Shi. Can we beat the AKS sorting network? Manuscript, 2019.

[3] S. Barber, X. Boyen, E. Shi, and E. Uzun. Bitter to better — how to make bitcoin a better currency. In *Financial cryptography and data security*, pages 399–414. Springer, 2012.

[4] T. H. Chan, K. Chung, B. M. Maggs, and E. Shi. Foundations of differentially oblivious algorithms. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2448–2467, 2019.

[5] T.-H. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. In *Proceedings of the 37th international colloquium conference on Automata, languages and programming: Part II*, ICALP'10, pages 405–417, Berlin, Heidelberg, 2010. Springer-Verlag.

[6] W. Chen and R. A. Popa. Metal: A metadata-hiding file-sharing system. In *NDSS*, 2020.

[7] N. Crooks, M. Burke, E. Cecchetti, S. Harel, R. Agarwal, and L. Alvisi. Obladi: Oblivious serializable transactions in the cloud. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 727–743, Carlsbad, CA, Oct. 2018. USENIX Association.

[8] P. Daian, R. Pass, and E. Shi. Snow white: Provably secure proofs of stake. Cryptology ePrint Archive, Report 2016/919, 2016.

[9] D. Darais, C. Liu, I. Sweet, and M. Hicks. A language for probabilistically oblivious computation. In *ACM Conference on Principles of Programming Languages (POPL)*, 2020.

[10] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 1988.

[11] S. Eskandarian and M. Zaharia. Oblidb: Oblivious query processing for secure databases. *Proc. VLDB Endow.*, 13(2):169183, Oct. 2019.

[12] O. Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In *ACM Symposium on Theory of Computing (STOC)*, 1987.

[13] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. *J. ACM*, 1996.

[14] Y. Guo, R. Pass, and E. Shi. Synchronous, with a chance of partition tolerance. In *CRYPTO*, 2019.

[15] D. E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching*. Addison-Wesley, 1973.

[16] A. E. Kosba, C. Papamanthou, and E. Shi. xjsnark: A framework for efficient verifiable computation. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 944–961, 2018.

[17] C. Liu, A. Harris, M. Maas, M. Hicks, M. Tiwari, and E. Shi. Ghostrider: A hardware-software system for memory trace oblivious computation. *SIGPLAN Not.*, 50(4):87–101, Mar. 2015.

[18] C. Liu, M. Hicks, and E. Shi. Memory trace oblivious program execution. In *Proceedings of the 2013 IEEE 26th Computer Security Foundations Symposium*, CSF '13, pages 51–65, 2013.

[19] C. Liu, X. S. Wang, K. Nayak, Y. Huang, and E. Shi. ObliVM: A programming framework for secure computation. In *IEEE Symposium on Security and Privacy*, 2015.

[20] J. R. Lorch, B. Parno, J. W. Mickens, M. Raykova, and J. Schiffman. Shroud: ensuring private access to large-scale data in the data center. In *Proceedings of the 11th USENIX conference on File and Storage Technologies, FAST 2013, San Jose, CA, USA, February 12-15, 2013*, pages 199–214, 2013.

[21] M. Maas, E. Love, E. Stefanov, M. Tiwari, E. Shi, K. Asanovic, J. Kubiatowicz, and D. Song. Phantom: Practical oblivious computation in a secure processor. In *ACM Conference on Computer and Communications Security (CCS)*, 2013.

[22] J. M. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. D. Gligor, and A. Perrig. Trustvisor: Efficient TCB reduction and attestation. In *IEEE Symposium on Security and Privacy*, 2010.

[23] J. M. McCune, B. Parno, A. Perrig, M. K. Reiter, and H. Isozaki. Flicker: An execution infrastructure for TCB minimization. In *EuroSys*, 2008.

[24] R. Pass and E. Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, PODC '17, pages 315–324, New York, NY, USA, 2017. ACM.

[25] R. Pass and E. Shi. The sleepy model of consensus. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, pages 380–409, 2017.

[26] R. Pass and E. Shi. Thunderella: Blockchains with optimistic instant confirmation. In *Eurocrypt*, 2018.

[27] L. Ren, C. Fletcher, A. Kwon, M. van Dijk, and S. Devadas. Design and implementation of the ascend secure processor. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 03 2017.

[28] E. Shi, T.-H. H. Chan, E. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *Network and Distributed System Security Symposium (NDSS)*, 2011.

[29] E. Shi, T.-H. H. Chan, E. Stefanov, and M. Li. Oblivious RAM with $O((\log N)^3)$ worst-case cost. In *ASIACRYPT*, pages 197–214, 2011.

[30] E. Shi, A. Perrig, and L. V. Doorn. BIND: A fine-grained attestation service for secure distributed systems. In *IEEE Symposium on Security and Privacy*, 2005.

[31] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas. Path ORAM – an extremely simple oblivious ram protocol. In *ACM Conference on Computer and Communications Security (CCS)*, 2013.

[32] X. S. Wang, T.-H. H. Chan, and E. Shi. Circuit ORAM: On Tightness of the Goldreich-Ostrovsky Lower Bound. In *CCS*, 2015.

[33] Y. Xu, W. Cui, and M. Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *2015 IEEE Symposium on Security and Privacy*, pages 640–656, May 2015.