Designing Hardware for Cryptography and Cryptography for Hardware

Srinivas Devadas*
Simon Langowski
Nikola Samardzic
Sacha Servan-Schreiber
Daniel Sanchez
Massachusetts Institute of Technology
Cambridge, MA, USA

ABSTRACT

There have been few high-impact deployments of hardware implementations of cryptographic primitives. We present the benefits and challenges of hardware acceleration of sophisticated cryptographic primitives and protocols, and briefly describe our recent work. We argue the significant potential for synergistic codesign of cryptography and hardware, where customized hardware accelerates cryptographic protocols that are designed with hardware acceleration in mind.

CCS CONCEPTS

• **Security and privacy** → **Cryptography**; **Security in hardware**;

KEYWORDS

cryptography, hardware acceleration, hardware security

ACM Reference Format:
Srinivas Devadas, Simon Langowski, Nikola Samardzic, Sacha Servan-Schreiber, and Daniel Sanchez. 2022. Designing Hardware for Cryptography and Cryptography for Hardware. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), November 7–11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3548606.3559393

1 INTRODUCTION

The Advanced Encryption Standard (AES) is a standardized scheme for symmetric key encryption [42]. Since the introduction of the `AES-NI` hardware instruction set in the early 2010s, there has been a tremendous growth in cryptographic software taking advantage of hardware-accelerated AES. These applications stem far beyond data encryption. For example, constructing one-way compression functions (i.e., keyed hash functions) from AES offers concrete performance boosts using hardware acceleration compared to software implementations of purpose-built hash functions. Similarly, hardware acceleration has motivated many uses of AES for instantiating other cryptographic primitives such as PRGs and PRFs. Hardware acceleration of these basic primitives has had tremendous downstream effects. Complex cryptographic primitives [9, 11, 29, 41] and applied systems [17, 19, 21, 43, 55] are often designed to use AES (because of the optimization enabled by using `AES-NI`). These cryptographic schemes and protocols were planned around the hardware they would be deployed on—specifically around *one* accelerated cryptographic primitive.

Another example that benefited from hardware acceleration is public-key cryptography. Proposals for hardware acceleration of RSA date back to 1980 [48], and virtually all smart cards in use today include hardware to accelerate public-key operations.

It is conceivable that additional hardware accelerated cryptographic primitives could result in a cambrian explosion of novel applications and systems.

One of the reasons for why AES was chosen by Intel is its wide acceptance by both government and industry entities "and its expected long term importance" [28]. We tease out two properties that we believe made AES especially appealing for hardware acceleration: (1) stability and (2) generality. Stability captures how unlikely a primitive is to change over time and generality measures the impact such a primitive could have on *downstream* applications.

Stability. Building special-purpose hardware requires a large investment (money, time, and resources) from chip makers and security experts. Further, in contrast to software, it is difficult (sometimes even impossible) to update hardware once deployed "in the wild." Therefore, any candidate cryptographic primitive should be vetted, widely accepted by cryptographers and security experts, and be unlikely to change down the road. Implementing hardware acceleration for new cryptographic primitives that have not stood the test of time is risky and may quickly result in obsolete hardware. Such changes can occur fairly frequently in cryptography. For example, recently it was found that the security of the Barreto-Naehrig elliptic curve [6], which was used in many pairing-based verifiable computation schemes and cryptocurrencies, provided only 100 bits of security as opposed to the claimed 128 bits [5, 37]. Another recent example is Supersingular Isogeny Diffie–Hellman (SIDH) [35], which was believed to be secure, even against post-quantum computers. Recently, however, this assumption was shown to be false under certain parameters, requiring, at minimum, updating all security parameters [13, 39, 49]. Tackling a more challenging design problem of parameterizable hardware acceleration can mitigate or avoid these situations, and provide more stability.

---

*Keynote speaker.

CCS '22, November 7–11, 2022, Los Angeles, CA, USA

ACM ISBN 978-1-4503-9450-5/22/11.
https://doi.org/10.1145/3548606.3559393

Generality. When accelerating a cryptographic primitive, it is important to target constructions that are likely to accelerate a large number of *other* primitives and applications using them. For example, it has been known for a long time that AES (or more generally one-way permutations) can be used to realize PRGs, PRFs, one-way compression functions, and a host of other cryptographic tools [7, 26, 33].

When cryptographic accelerators make sense. Most cryptographic operations are used too infrequently to be accelerated. For example, key agreement protocols used ubiquitously on the Internet to establish connections between two machines. However, they are typically executed once per connection and therefore not used frequently enough to merit acceleration. As such, a hardware accelerator for key agreement would be an inefficient use of precious resources. In contrast, digital signatures and encryption are ubiquitous *and* used repeatedly by many devices, possibly making them better candidates for acceleration.

The chicken and the egg. In many instances, hardware accelerators have found usage outside of their original intended purpose. It is difficult to predict which primitives are worthwhile accelerating because *current* usage of the primitive is not always indicative of its *potential* usage if a hardware accelerator were to exist for it. As an example, consider our key agreement example above. If key agreement were cheap, new kinds of protocols that don't require long-lived sessions (e.g., a DNS server doing per-DNS-query key agreement), or systems that avoid complex designs like TLS session resumption may suddenly appear. In many cases, if a hardware accelerator were to be developed for some primitive, then cryptographic protocols could be tweaked or redesigned to use it, as was exemplified with `AES-NI`. Moreover, accelerators designed for one domain may find wide usage in a completely different domain. Graphics Processing Units (GPUs) were originally developed for the purpose of accelerating *computer graphics.* Today, GPUs are heavily used in machine learning and cryptocurrency mining—applications that couldn't be further removed from rendering pixels. The potential impact on cryptographic protocols from new accelerators is difficult to gauge and might only appear obvious in retrospect, similarly to the trajectory of GPUs and the AES instruction set. However, the decision to design and build a hardware accelerator is often contingent on the primitive already being widely deployed and used, resulting in a conundrum.

2 DESIGNING HARDWARE FOR CRYPTO

As cryptographers have worked to solve more complex problems, solutions inevitably require larger overheads. However, as computers and protocols have improved, these technologies have become viable in real-world implementations. For example, Path ORAM has recently been deployed in the Signal messenger [15], zero-knowledge proofs are used widely in cryptocurrencies [23, 31, 51], and private information retrieval can be used for resolving DNS queries anonymously [20] and for certificate transparency [30]. We often imagine these technologies as adding "security" or "privacy" to some baseline functionality, at some cost in performance. By accelerating cryptography with hardware, we can remove the added overheads and avoid the performance costs of building more secure and private systems.

Consider Fully Homomorphic Encryption (FHE) [25], a class of encryption schemes that enables generic computation on encrypted data, allowing secure offloading of computation. Even modern, highly optimized schemes [12, 14] have very significant overheads. To avoid the overheads of FHE, accelerators for private deep learning, Gazelle [36] and Cheetah [45], combine shallow homomorphic encryption (HE) with multi-party computation (MPC). Unfortunately, these systems require very frequent communication with the client, essentially after every single level of multiplication. While they do accelerate private deep learning, they are limited by high client-server communication and client encryption/decryption overheads. Delphi [40] shows that each DNN inference imposes *gigabytes* of traffic, which quickly bottlenecks the performance. CHOCO [54] shows that, even after accelerating client operations, communication costs still dominate.

As we describe below, recent work has resulted in FHE now bordering on practicality, thanks to hardware acceleration [22, 36, 45, 47, 50, 56].

F1 Accelerator for FHE. F1 [22] is the first *programmable* FHE accelerator, i.e., capable of executing full FHE programs. F1 is a wide-vector processor with novel functional units deeply specialized to FHE primitives. This organization provides so much compute throughput that data movement becomes the key bottleneck. Thus, F1 is primarily designed to minimize data movement. It does so by speeding up shallow FHE computations (i.e., those of limited multiplicative depth) by roughly 5,000× over a 4-core CPU. F1 becomes memory-bandwidth bound on deeper computations (e.g., deep neural networks). This is because deep FHE programs require very large ciphertexts (tens of MBs each) and different algorithms, which F1 does not support well.

CraterLake Accelerator for FHE. CraterLake [50] addresses these shortcomings and is the first accelerator to effectively speed up arbitrarily large FHE programs. CraterLake introduces a new hardware architecture that efficiently scales to very large ciphertexts, novel functional units to accelerate key kernels, and new algorithms and compiler techniques to reduce data movement. These advances help CraterLake outperform a 32-core CPU by roughly 4,500× and deliver performance on deep benchmarks that is an order of magnitude better than a scaled-up F1, with the same chip area as CraterLake. These speedups enable new applications for FHE, such as real-time inference using deep neural networks.

In order to perform arbitrary computations, ciphertexts need to be periodically refreshed using a computationally-expensive procedure called *bootstrapping*. By dramatically accelerating bootstrapping, CraterLake avoids the high communication costs of HE-MPC and shallow HE designs. To avoid bootstrapping, these prior approaches require the client to receive, re-encrypt, and resend ciphertexts that have exhausted their multiplicative budgets. In the CraterLake benchmarks [50], avoiding each bootstrapping would require transferring over 13 MB between client and server, which would make client-server communication two orders of magnitude more time-consuming than the actual server computation (assuming a standard 100 Mbps link). Specifically, these transfers correspond to the server sending noisy (intermediate) ciphertext to the client, which the client decrypts, then reencrypts, and resends back to the server. Even if we ignore client computation latency, on a 100 Mbps connection this would require over one second per ciphertext. In contrast, CraterLake bootstraps this ciphertext in 3.9 ms, 256× faster. Bootstrapping therefore greatly reduces encryption and network overheads, and hardware acceleration of bootstrapping reduces its computational overhead.

3 DESIGNING CRYPTO FOR HARDWARE

Many cryptographic protocols require high overheads in terms of communication (bandwidth) and computation. We argue that the design of cryptography for hardware should focus on the communication costs. This is simply because, while computation can be accelerated using parallelism and specialized hardware, accelerating communication between parties is more difficult, requiring more extensive (e.g., physical infrastructure) changes.

Consider a concrete example of Amdahl's law [3] applied to a protocol that spends 50% of its time in computation and 50% of its time in communication. Providing a 100× speedup to computation will only result in a 2× speedup to the protocol. However, if we had a slower protocol that traded communication for computation, then, when hardware acceleration is applied, this protocol will be faster overall. For a protocol that runs in a reasonable time-frame without acceleration, a $1000 - 10000\times$ speedup to computation will make computational costs negligible, and only the communication costs will remain.

We can also compare computation and communication costs from a viewpoint of energy consumption. It takes ~1,000× as much energy to transmit an 8-bit word off-chip (i.e., DRAM access) as to perform word addition on-chip [32, 44]. Further, transmitting the same word over the Internet increases energy consumption by many orders of magnitude compared to a DRAM access [18], resulting in an inherent cost differential.

Few current protocols outside of FHE are designed for this significant skew between communication and computation costs, and we argue that hardware acceleration further exacerbates this skew. As acceleration is able to speed up and reduce energy cost of computation by several orders of magnitude, communication costs become the only bottleneck. Therefore, cryptography designed for hardware should focus on minimizing communication costs. For example, [38] provides a codesign of a Private Information Retreival (PIR) protocol and near-storage compute to reduce communication costs.

Finally, back to the notion of generality, given dedicated accelerators such as F1 and CraterLake [22, 50] that accelerate FHE, can we design other cryptographic schemes that take advantage of the same hardware? For example, verifiable computation (VC) is a cryptographic protocol that enables a (usually computationally limited) entity to verify the correctness of an expensive computation delegated to an untrusted server [16, 27]. FHE and VC share many common characteristics: (1) They convert the computation to be performed into a circuit, (2) an untrusted server evaluates the circuit on inputs to produce outputs, and (3) this evaluation is extremely computationally expensive on CPUs. Can we design and build a shared accelerator for FHE and VC? This motivates designing VC schemes that are lattice-based (whereas current VC schemes are group-based). Lattice-based VC schemes [2, 10, 24, 34] might be amenable to acceleration, but hardware and cryptography will have to meet in the middle. Although both schemes are based on lattices, they differ slightly in the parameters, fields, and other structures required.

4 LOOKING TO THE FUTURE

Hardware acceleration of AES supercharged the deployment of cryptography on the Internet. Are there other primitives useful in multiple cryptographic protocols that would have a similar impact?

The learning with errors (LWE) assumption [46] is assumed to be post-quantum secure, making it a good candidate for hardware acceleration (in comparison to factoring and discrete-logarithm style assumptions, which are broken with quantum computers [7]). This has motivated standardization of post-quantum cryptography [1] which will necessitate *fast* post-quantum cryptography. Efforts along these lines include Amazon Web Services enabling hybrid post-quantum schemes for TLS [52, 53], the development and cryptoanalysis [13, 39, 49] of NIST candidates [1], key agreement protocols based on LWE [8], and many more. Unfortunately, most post-quantum secure cryptographic algorithms are slower than their pre-quantum counterparts; hardware acceleration can help close this gap. Accelerating the underlying operations for LWE, for example, would be applicable to a host of primitives, such as FHE, verifiable computation, and even cryptographic primitives that do not use the LWE assumption. For example, an accelerator for LWE that achieves significant speedups over CPUs (even with SSE and AVX instructions) could impact an extraordinary number of cryptographic primitives, ranging from multi-party computation to obfuscation [4].

We believe a successful marriage of hardware acceleration and cryptography has high potential for impact: by lowering the cost of cryptographic techniques, these techniques will become an essential part of the secure datacenters of the future, saving billions of dollars and enabling security in a sustainable, energy-efficient way.

ACKNOWLEDGEMENTS

We thank Henry Corrigan-Gibbs, Raluca Ada Popa, Ron Rivest, and Nickolai Zeldovich for insightful comments and suggestions.

REFERENCES

[1] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the second round of the nist post-quantum cryptography standardization process. *US Department of Commerce, NIST*, 2020.

[2] Martin R Albrecht, Valerio Cini, Russell WF Lai, Giulio Malavolta, and Sri AravindaKrishnan Thyagarajan. Lattice-based snarks: Publicly verifiable, preprocessing, and recursively composable. *Cryptology ePrint Archive*, 2022.

[3] Gene M Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485, 1967.

[4] Sebastian Banescu, Martín Ochoa, Nils Kunze, and Alexander Pretschner. Idea: benchmarking indistinguishability obfuscation–a candidate implementation. In *International Symposium on Engineering Secure Software and Systems*, pages 149–156. Springer, 2015.

[5] Razvan Barbulescu and Sylvain Duquesne. Updating key size estimations for pairings. *Journal of cryptology*, 32(4):1298–1336, 2019.

[6] Paulo SLM Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *International workshop on selected areas in cryptography*, pages 319–331. Springer, 2005.

[7] Dan Boneh and Victor Shoup. A graduate course in applied cryptography. *Draft 0.5*, 2020.

[8] Joppe W Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the tls protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy*, pages 553–570. IEEE, 2015.

[9] Joppe W Bos, Onur Özen, and Martijn Stam. Efficient hashing using the aes instruction set. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 507–522. Springer, 2011.

[10] Cecilia Boschini, Jan Camenisch, Max Ovsiankin, and Nicholas Spooner. Efficient post-quantum snarks for rsis and rlwe and their applications to privacy. In *International Conference on Post-Quantum Cryptography*, pages 247–267. Springer, 2020.

[11] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1292–1303, 2016.

[12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, New York, NY, USA, 2012. Association for Computing Machinery.

[13] Wouter Castryck and Thomas Decru. An efficient key recovery attack on sidh (preliminary version). *Cryptology ePrint Archive*, 2022. https://eprint.iacr.org/2022/975.

[14] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, 2017.

[15] Graeme Connell. Technology deep dive: Building a faster ORAM layer for enclaves. https://signal.org/blog/building-faster-oram/, 2022.

[16] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 90–112, 2012.

[17] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *2015 IEEE Symposium on Security and Privacy*, pages 321–338. IEEE, 2015.

[18] David Costenaro and Anthony Duer. The megawatts behind your megabytes: going from data-center to desktop. *Proceedings of the 2012 ACEEE Summer Study on Energy Efficiency in Buildings, ACEEE, Washington*, pages 13–65, 2012.

[19] Emma Dauterman, Eric Feng, Ellen Luo, Raluca Ada Popa, and Ion Stoica. DORY: An encrypted search system with distributed trust. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 1101–1119, 2020.

[20] Natnatee Dokmai, L Jean Camp, and Ryan Henry. A proposal for assisted private information retrieval. *Cryptology ePrint Archive*, 2022.

[21] Saba Eskandarian, Henry Corrigan-Gibbs, Matei Zaharia, and Dan Boneh. Express: Lowering the cost of metadata-hiding communication with cryptographic privacy. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1775–1792, 2021.

[22] Axel Feldmann, Nikola Samardzic, Aleksandar Krastev, Srini Devadas, Ron Dreslinski, Christopher Peikert, and Daniel Sanchez. F1: A fast and programmable accelerator for fully homomorphic encryption. In *Proceedings of the 54th annual ACM/IEEE International Symposium on Microarchitecture*, 2021.

[23] Ethereum Foundation. Zero-knowledge rollups. https://ethereum.org/en/developers/docs/scaling/zk-rollups/, 2022.

[24] Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. Lattice-based zk-snarks from square span programs. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 556–573, 2018.

[25] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC '09*, pages 169–178, 2009.

[26] Oded Goldreich. *Foundations of cryptography: a primer*, volume 1. Now Publishers Inc, 2005.

[27] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. *Journal of the ACM (JACM)*, 62(4):1–64, 2015.

[28] Shay Gueron. Intel® advanced encryption standard (aes) new instructions set. https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf, 2010.

[29] Shay Gueron and Fabian Schlieker. Software optimizations of ntruencrypt for modern processor architectures. In *Information Technology: New Generations*, pages 189–199. Springer, 2016.

[30] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. Cryptology ePrint Archive, Paper 2022/949, 2022. https://eprint.iacr.org/2022/949.

[31] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification. *GitHub: San Francisco, CA, USA*, page 1, 2016.

[32] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14. IEEE, 2014.

[33] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 44–61, 1989.

[34] Yuval Ishai, Hang Su, and David J Wu. Shorter and faster post-quantum designated-verifier zksnarks from lattices. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 212–234, 2021.

[35] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *International Workshop on Post-Quantum Cryptography*, pages 19–34. Springer, 2011.

[36] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1651–1669, 2018.

[37] Taechan Kim and Razvan Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In *Annual international cryptology conference*, pages 543–571. Springer, 2016.

[38] Jilan Lin, Ling Liang, Zheng Qu, Ishtiyaque Ahmad, Liu Liu, Fengbin Tu, Trinabh Gupta, Yufei Ding, and Yuan Xie. Inspire: In-storage private information retrieval via protocol and architecture co-design. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, page 102–115, New York, NY, USA, 2022. Association for Computing Machinery.

[39] Luciano Maino and Chloe Martindale. An attack on sidh with arbitrary starting curve. *Cryptology ePrint Archive*, 2022. https://eprint.iacr.org/2022/1026.

[40] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. Delphi: A cryptographic inference service for neural networks. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2505–2522, 2020.

[41] Tilo Müller, Felix C Freiling, and Andreas Dewald. TRESOR runs encryption securely outside RAM. In *20th USENIX Security Symposium (USENIX Security 11)*, 2011.

[42] James Nechvatal, Elaine Barker, Lawrence Bassham, William Burr, Morris Dworkin, James Foti, and Edward Roback. Report on the development of the advanced encryption standard (aes). *Journal of research of the National Institute of Standards and Technology*, 106(3):511, 2001.

[43] Zachary Newman, Sacha Servan-Schreiber, and Srinivas Devadas. Spectrum: High-bandwidth anonymous broadcast. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 229–248, 2022.

[44] Mike O'Connor, Niladrish Chatterjee, Donghyuk Lee, John Wilson, Aditya Agrawal, Stephen W Keckler, and William J Dally. Fine-grained dram: Energy-efficient dram for extreme bandwidth systems. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 41–54, 2017.

[45] Brandon Reagen, Woo-Seok Choi, Yeongil Ko, Vincent T Lee, Hsien-Hsin S Lee, Gu-Yeon Wei, and David Brooks. Cheetah: Optimizing and accelerating homomorphic encryption for private inference. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 26–39. IEEE, 2021.

[46] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[47] M. Sadegh Riazi, Kim Laine, Blake Pelton, and Wei Dai. Heax: An architecture for computing on encrypted data. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, page 1295–1309, New York, NY, USA, 2020. Association for Computing Machinery.

[48] Ronald L Rivest. A description of a single-chip implementation of the rsa cipher. *LAMBDA Fourth Quarter*, 1980.

[49] Damien Robert. Breaking sidh in polynomial time. *Cryptology ePrint Archive*, 2022. https://eprint.iacr.org/2022/1038.pdf.

[50] Nikola Samardzic, Axel Feldmann, Aleksandar Krastev, Nathan Manohar, Nicholas Genise, Srini Devadas, Karim Eldefrawy, Christopher Peikert, and Daniel Sanchez. Craterlake: A hardware accelerator for efficient unbounded computation on encrypted data. In *Proceedings of the 49th annual International Symposium on Computer Architecture (ISCA-49)*, 2022.

[51] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE symposium on security and privacy*, pages 459–474. IEEE, 2014.

[52] Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum tls without handshake signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1461–1480, 2020.

[53] Amazon Web Services. Using hybrid post-quantum tls with aws kms. https://docs.aws.amazon.com/kms/latest/developerguide/pqtls.html, 2022. Accessed: 2022-08-30.

[54] McKenzie van der Hagen and Brandon Lucia. Practical encrypted computing for iot clients. *arXiv preprint arXiv:2103.06743*, 2021.

[55] Frank Wang, Catherine Yun, Shafi Goldwasser, Vinod Vaikuntanathan, and Matei Zaharia. Splinter: Practical private queries on public data. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 299–313, 2017.

[56] Ye Zhang, Shuo Wang, Xian Zhang, Jiangbin Dong, Xingzhong Mao, Fan Long, Cong Wang, Dong Zhou, Mingyu Gao, and Guangyu Sun. Pipezk: Accelerating zero-knowledge proof with a pipelined architecture. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 416–428. IEEE, 2021.