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:
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.

This work is licensed under a Creative Commons Attribution International 4.0 License.
Generality. When accelerating a cryptographic primitive, it is important to target constructs 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 [30] 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 boots this ciphertext in 3.9 ms, 256x 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× 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 Retrieval (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
