Confidential Computing—a brave new world

Dominic P. Mulligan, Gustavo Petri, Nick Spinale, Gareth Stockwell and Hugo J. M. Vincent
Arm Ltd. Cambridge, UK
forename.surname@arm.com

***Abstract*—The semiconductor industry is witnessing a nascent security paradigm shift in the rise of Confidential Computing. Driven by the need to protect computations delegated to co-tenanted machines operated by Cloud Computing services, mainstream instruction set architectures are gradually introducing novel features that can be used to establish protected *isolates* offering strong integrity and confidentiality guarantees to code and data contained within. Coupled with a Remote Attestation protocol, a third-party may request the launch of an isolate on an otherwise untrusted machine and know—with a high degree of assurance—that a payload of code and data was indeed loaded into a legitimate isolate with a particular configuration.**

We argue that this ability to reliably establish a safe "beach-head" on an untrusted third-party's machine has far-reaching consequences with applications beyond protecting workloads delegated to Cloud Computing services. In a future world where facilities for Confidential Computing are widely deployed and used, we imagine a utopia where inadvertent data leakage is a curiosity of a bygone age, with encrypted data moving from isolate to isolate and never resting in plaintext. Moreover, data is only released in explicitly delimited ways for processing, with systems and individuals exhibiting fine-grained control over data.

We report on recent activities within Arm in attempting to realize this vision, and hope that this paper acts as a "call to arms" to others to join with us in fully exploring the potential of these emerging technologies.

***Index Terms*—Confidential Computing, Arm® Confidential Compute Architecture (Arm CCA), Remote Attestation, Veracruz, IceCap**

I. Introduction

By and large, protecting secrets when they are *at rest*—stored on disk for example—is a solved problem. Likewise, we know how to protect secrets *in transit*, when being communicated device-to-device, using transport-layer security protocols such as TLS. What we have yet to fully grapple with is how to efficiently and ergonomically protect secrets when *in use*—for instance, how to protect potentially collaborative computations over secret inputs, or how to protect computations in environments not under the full control of a single party.

Ordinarily, this knowledge gap would not be a huge problem if not for a wildly-popular industrial trend: *Cloud Computing*. Increasingly, sensitive computations—finance, health, even computations and data related to national security—are being deployed on shared pay-per-use infrastructures owned and operated by Cloud services that are able to leverage economies of scale and comparative advantage to drive down costs. While traditional isolation and virtualization technologies (like hypervisors and OSes) have served us well, the prevalence of the Cloud has exposed software systems to new security vulnerabilities and heightened the risk of old ones [1], [2]. Academia and industry have acknowledged this challenge, and *Confidential Computing* architectures and platforms have emerged as a result. Some proposed solutions are based purely on cryptographic primitives, others rely on strong security and isolation guarantees through the use of formal methods and verification [3], [4], and yet others are enabled by hardware-backed *strong isolation* mechanisms.

Each offers a trade-off in terms of threat-model, performance, maintenance, and usability. We will concentrate on the latter two for their performance advantages, and remain optimistic about cryptographic techniques like Fully Homomorphic Encryption (FHE) [5], [6] for their future promise. Note that solutions for *first-party* (for lack of a better term) Confidential Compute have existed for over a decade in the client-device space, represented primarily by Arm's TrustZone® [7]. Here, a small collection of trusted vendors supply privileged services—cryptographic key management, for example—which needs protecting from malicious or untrustworthy software and users. However, driven by the security challenges inherent to Cloud and multi-tenant computing, the semiconductor industry is witnessing a paradigm shift in security as most major instruction set architectures add support for *third-party* Confidential Compute—witness AMD's Secure Encrypted Virtualization (SEV), Arm's TrustZone and new Confidential Compute Architecture, and Intel's Software Guard Extensions (SGX) and Trust Domain Extensions (TDX), for example. Whilst each of these technologies is unique in its own way, we may identify commonalities:

a) Hardware support for isolation against a privileged attacker: each technology introduces what we shall call *isolates*, a new kind of architectural primitive—variously called a Secure Enclave, Trusted Execution Environment, Realm, Protected Virtual Machine, Trusted Application, amongst other names, depending on the technology—that provides strong *integrity* and *confidentiality* guarantees to code and data against a wide class of attackers. Depending on the technology, this class of attacker may commonly be assumed to include attackers able to co-opt the capabilities of privileged system software—such as the operating system, hypervisor, and BIOS—and potentially even a class of physical attacker capable of inspecting and manipulating off-chip memory.

b) Small Trusted Computing Base: each technology places an emphasis on keeping the *Trusted Computing Base* (or TCB, henceforth) as minimal as possible, encompassing only the content of the isolate and any essential hardware or firmware needed to implement the technology in focus.

c) Root of trust, measurement, and Remote Attestation: most technologies provide a hardware root of trust and Remote Attestation protocol[1] through which a skeptical challenger can obtain strong, cryptographic evidence that a legitimate isolate has been initialized on a remote (or local) machine, containing software with a known hash and configuration parameters.

Taken together, this combination of features allows one to establish a protected "beach-head" on an untrusted system with a known good configuration, safe from spying and interference—exactly what is needed to protect computations delegated to the Cloud or any other untrusted server.

Many discussions of Confidential Compute naturally fixate on Cloud Computing use-cases. However, there are also natural applications wherever compute happens, whether that be on client-devices—mobile phones, smart watches, tablets, PCs, home assistants, and similar—or on server-class hardware sitting outside of a traditional Cloud data centre, for example at the "network edge" or even in supercomputing facilities. Moreover, Confidential Compute can (and will, in time) be extended to cover accelerators and other programmable devices—for example machine learning accelerators, GPUs, and Smart NICs—by providing strongly-partitioned isolated execution environments on these devices, and providing associated attestation mechanisms. The future for Confidential Compute is bright.

Yet, what we hope to do here is look beyond discussion of the immediate applications, and extensions, of these new Confidential Computing technologies. Indeed, in the short term we see most discussion and potential application of strong isolation technologies, in both infrastructure- and client-class devices, as simply moving existing workloads—virtual machine images, databases, and similar—inside an isolate as a form of defence in depth. Whilst this is commercially important, we are particularly interested in exploring a longer term *(utopian) vision* for what the rise of Confidential Compute could mean for industry and wider society.

II. Confidential Compute: our vision

Imagine a world where strong isolation technology is widely deployed and used. What could this world look like?

In this world computations can be freely moved from device to device without a privacy risk, provided the device is capable of launching an isolate. Computations are "mobile", and can be handed-off to devices as an individual moves around a building, or down the street—scheduled onto devices with dedicated accelerators, or those sitting idle with spare computing capacity. Computations on sensitive data can now be crowd-sourced—medical images are processed *en masse* in private grid computing networks, for example. Data is now released in delimited ways, as mutually mistrusting individuals enroll in collaborative computations which use isolates as a safe "neutral ground" within which the computation takes place, essentially delivering on the promise of Secure Multiparty Computations.[2] Enhanced, pervasive privacy acts as an economic lubricant as data once deemed too risky to share is freely released——in strongly controlled ways. More generally, data is now pervasively encrypted, and only temporarily decrypted for use within an isolate, before being encrypted again for storage—inadvertent data leaks are now a curiosity of the past, and data sets once deemed too sensitive to outsource or share given the risk of data leaks are now moved around freely. The concepts of control and possession of data are no longer conflated.

Indeed, once the capability of delegating computations safely to untrusted third-parties is developed, it no longer really matters, from a privacy point-of-view, where compute happens. Computations can be scheduled wherever it is most convenient, and wherever is best placed to carry out the computation. Moreover, isolates can be used as virtual strongboxes, used to address problems ordinarily deemed within the domain of cryptographers, albeit in more efficient and ergonomic ways.

Within Arm, we have initiated a program exploring this utopian vision through concrete point solutions—exploring what new ways of collaborating, and working with data, emerge in a world where strong isolation technologies are widely deployed. We describe some of our activities in the remainder, and aside from a précis of our current activity, we hope this position paper serves as a "call to arms". Before doing that, we outline some important cross-cutting aspects and challenges that we believe will help in realizing our vision.

Reliable hardware and software stacks: Establishing trust in the hardware and software that underlies strong isolation technology is a foundational aspect in delivering on the promises of Confidential Compute.

The means available to underpin and establish trust vary depending on whether we talk about hardware, software, or networking protocols and distributed systems. At the lowest level, in hardware, trust is gained by having reliable root-of-trust flows that guarantee that systems are booted into known and valid states, from trusted read-only images (possibly signed by a known and trusted provider). Moreover, for isolates, the hardware, in tandem with firmware, must provide services that enable the measurement and cryptographic authentication of their initial contents and state.

In the case of software, trust is enabled by source code which is: i) available and auditable, ii) built into binaries with bit-exact reproducibility, iii) signed by a trusted party, iv) formally verified, v) equipped with verifiable certificates (e.g. in the form of a proof witness). These are but a few examples of practices, ordered by increasing impact on trustworthiness, that we believe should be adopted as standard in Confidential Computing's critical software.

Note that many principles for increasing trust in software might appear to be applicable to hardware as well, however there are crucial differences. Underlying them is the capa-

[1] Arm TrustZone as an *architecture* provides no Remote Attestation, though *deployments* could define one, using measured boot and hardware root of trust.

[2] Despite the great algorithmic progress in recent years, we see Secure Multiparty Computations as still too slow for common practical application.

bility to reproduce, measure (hash) and compare artifacts; this enables signatures, and makes auditing of source code useful—otherwise there is no way to know if what you audited is manifested in the hardware you are using. At present, there are no practical ways to measure a physical hardware implementation to assert that it correctly implements a given piece of hardware source (e.g. Verilog/RTL code), and indeed solutions to this problem still seem far fetched (for example, state of the art nondestructive die imaging requires the use of a particle accelerator [8]).

Standardized protocols and policies: Some protocols required for Confidential Computing have been the subject of study, with multiple extant implementations—notable examples include remote attestation protocols [9], [10], and protocols for firmware update and trusted execution environment setup [11], [12]. However, building distributed systems using Confidential Computing will often require more than one participant, with different trust relationships, and with different risks. It is therefore important to design protocols and policies to mediate these trust relationships as well as the provenance and manipulation of data and results. In the best case we would expect these policies to be attestable as well. Arm's Veracruz project, discussed later, incorporates a runtime system and policy format to establish policies dictating ownership of software, data, and results of computations. While Veracruz policies are relatively simple, given the use cases of Veracruz, we anticipate that this is an important topic of research as more use-cases for distributed confidential systems emerge.

New use-cases enabled by Confidential Compute: Once Confidential Compute support becomes ubiquitous across client and infrastructure devices, it can form the foundation for new use-cases, building trust in everyday interactions. One such promising use-case is end-to-end provenance and cryptographic authenticity of images, video and other content in online news and social media. An isolate could be used at the point of capture to execute the image signal processing computations and append signature and attestation material to the resulting image (enabling relying parties to have confidence that the processing was faithful, without compromising the privacy of the user). Downstream, journalists and newsroom editors could run their image or video editing operations in isolates, again allowing relying parties to understand what operations were performed. Such a use-case is being developed and standardized by the Coalition for Content Provenance and Authenticity (C2PA) [13], where Arm is a member. We hope that this technology may begin to stem the harmful spread of misinformation and disinformation in social media.

Realizing the vision outlined above requires changes to existing computing stacks, starting at the hardware level and extending to programming languages, libraries, and protocols. As is typical in security, each element in this stack is subject to distinct threats, and their composition is only as robust as the weakest link in the stack. We therefore think that principled and robust design principles have to be enacted to deliver on the security promises of Confidential Compute.

III. Towards realizing the vision

We now describe some point solutions we are working on in attempting to deliver upon our utopian vision. While each solution described relates to a specific layer—working from hardware upwards—we expect that in future each will contribute toward an end-to-end Confidential Compute stack.

A. Hardware foundations with Arm CCA

The recently-announced Arm Confidential Compute Architecture (or Arm CCA, henceforth) [14] introduces the Realm Management Extension (or RME, henceforth), and with it introduces a form of isolate to the Armv9-A [15] architecture profile, called a *Realm*. A Realm is an isolate protected from privileged—and other non-privileged, but unrelated—software surrounding it, including the operating system, hypervisor, and TrustZone firmware. An untrusted operating system manages the memory and CPU resources of a Realm but cannot access nor interfere with its content or state.

RME extends the Arm architecture with both a new physical address space, the Realm physical address space, and a new execution state called the Realm security state, which are protected from physical attacks through memory encryption, and from both untrusted operating systems and applications running in the Non-secure security state and software running in the Secure security state of Arm TrustZone, by access controls. RME also introduces a mechanism for securely and dynamically partitioning the system's memory resources between the Realm and non-Realm address spaces.

Trusted firmware, called the Realm Management Monitor (or RMM, henceforth), acts as a *separation kernel* in the Realm security state and isolates Realms from each other. The RMM co-operates defensively with an untrusted operating system, executing in the Non-secure world, to manage opaque memory and CPU resources of Realms, and can attest to a Realm's confidentiality and initial state, providing the foundations for a Remote Attestation mechanism for Realms.

As part of our work towards a reliable and trustworthy implementation of RME we are using software *model checking* techniques to ensure that Arm's implementation of the RMM upholds its ABI specification. Moreover, we are utilizing *formal models and methods* to study some of the emergent security guarantees that can be derived from the specification. For now, we have concentrated on functional correctness, and coarse-grained notions of confidentiality and integrity.

Note that one peculiarity of Arm's architecture is that the RMM and TrustZone firmware are both mutually distrusting, and while the mechanisms protecting each are similar, they differ in purpose. The Realm security state is dedicated to supporting Confidential Compute, wherein Realms distrust the operating system but are less privileged than the operating system, and can be dynamically created. The Secure security state, which hosts TrustZone firmware, is dedicated to supporting first-party trusted services which are similarly protected from the operating system, but, in contrast, trusted by the operating system—being potentially more privileged than the operating system itself—and also tend to exist in static “carve outs” and pre-loaded on devices.

The primitives provided by RME, coupled with the RMM ABI, are general enough to support a broad range of Arm platforms and their applications. In the Cloud, the unit of isolation may be as large as a virtual machine. In this case, the RMM is used as a *blind hypervisor* running in the Non-secure state, which orchestrates, manages, and schedules virtual machines running in Realms without access to their state or content. On a client device, resource constraints may not permit an entire virtual machine for each confidential component, in which case a Realm may contain just a single process, or part of a process running in the Non-secure state.

B. Strong isolation on existing platforms with IceCap

Arm CCA is Arm’s Confidential Compute architecture for Armv9-A platforms, yet billions of Armv8-A powered-devices will continue to be widely used far into the foreseeable future. Can we provide a *pragmatic* Confidential Compute solution within the limitations of these devices?

IceCap [16] is our open-source project tackling this problem, supporting a form of isolate on compatible Armv8-A devices. IceCap is a hypervisor with a minimal, attestable TCB based on the high-assurance seL4 microkernel [3]. Conceptually similar to the RMM in Arm CCA, the IceCap hypervisor co-operates defensively with an untrusted operating system to manage opaque memory and CPU resources of isolates, with minimal overhead. The extensive security and functional correctness proofs of seL4 provide a high degree of assurance that IceCap correctly protects isolates from software attacks [17], [18], [19]. IceCap works within the limitations of existing hardware platforms, and does not depend on any new hardware features introduced by Arm CCA, relying on memory access controls provided by the Armv8-A Memory Management Units (MMUs) and System MMUs (SMMUs) to protect isolates. The IceCap threat model for isolates is weaker and more variable across hardware platforms than Arm CCA Realms, especially in the presence of an attacker with physical access, due to the lack of memory encryption.

C. Deployment and collaboration with Veracruz

Consider the following scenario. Alice and Bob are each in possession of private data sets and wish to collaborate to obtain a machine learning model that was trained over their joint data sets. Neither wishes to divulge their data set to the other, nor to anybody else for that matter, and the output from this computation should only be available to Alice and Bob.

Abstracting a little, we observe that a group of *data providers* with data sets D_i for $1 \leq i \leq m$ and a *program provider* with program π wish to collaborate together to compute the result $\pi(D_1, \ldots, D_m)$. Most generally, each data provider wishes to keep their data input, D_i, a secret, contributing it to the collaborative computation without divulging it to anybody else (though any principal may choose to intentionally declassify one of their secrets, perhaps as a means of inducing another to take part in a collaborative computation). Moreover, once identified, this pattern of $m + 1$ mutually-distrusting principals computing a joint result with strong privacy guarantees appears over-and-over again—further examples include privacy-preserving auctions, polls and surveys, private graph analytics and set-intersection, and privacy-preserving sensor fusion, amongst many others.

Veracruz [20], an open-source project adopted by the Confidential Compute Consortium, captures this pattern, and uses strong isolation technology—including Arm TrustZone Trusted Applications, AWS Nitro Enclaves, Intel SGX Secure Enclaves, seL4/IceCap virtual machines, and soon Arm Confidential Compute Architecture Realms—to provide a neutral ground within which collaborative computations take place. Veracruz provides protocols, tools, and importantly a *trusted runtime* component that is loaded into an isolate, and which hosts a computation. Working with the isolate, the runtime acts as a two-way sandbox, preventing the untrusted host from spying or interfering, but also limiting the program π to computing only a pure partial function of its inputs, modulo the ability to sample random data. Participants provision secrets into the runtime via TLS after using Remote Attestation to authenticate the isolate and its Veracruz runtime content.

To abstract over different isolation technologies, we use WebAssembly [21] as an executable format, and the WebAssembly System Interface [22] as a programming environment. Computations are written in familiar programming languages (including Rust and C) and their standard libraries without needing to resort to custom tools. Moreover, different attestation protocols are also abstracted over using a method of *proxy attestation*, which sits in front of each “native” attestation service and exposes the Arm PSA attestation protocol [12]. With this, client code interacting with an isolate need only understand PSA attestation, not a zoo of different protocols.

Veracruz, strictly speaking, is a framework for designing and deploying collaborative privacy-preserving computations. Every Veracruz computation is parameterized by a public *policy* file, capturing the “topology” of a computation—describing who supplies inputs, programs, and importantly who obtains results. By varying this policy file, and the program π, a host of privacy-preserving collaborative computations can be designed and made to fit a particular set of existing trust relationships.

Veracruz’s platform abstraction also means it is an effective way of *deploying* computations across a host of different isolation technologies, and makes Veracruz a central pillar in realizing our vision: providing both a means for a group of mutually mistrusting individuals to collaborate, and a substrate through which computations can be safely moved around: *write once, isolate anywhere*.

IV. Gaps and open problems

We have so-far outlined our vision and some ongoing efforts. We now discuss some open challenges towards realizing this vision. We don’t intend for this list to be exhaustive; there are also hard problems to solve in supply chain security, system-level optimization, developer productivity, and so on, that we do not have space to address.

A. Unprotected leakage

The leakage of confidential information through side channels is a common problem for strong isolation technologies. Whilst hardware plays a crucial role in the extent of leakage, it cannot address all leakage alone. Computations whose execution time, data access patterns, and control flow depends on confidential data will be subject to side channels that are potentially observable. Unfortunately, almost any program not specifically written and compiled to avoid side channels will have secret-dependent control flow or data access patterns.

The threat model is relevant here; most Confidential Computing solutions support cryptographic memory protections aimed at protecting against an attacker with physical access and capable of monitoring and manipulating communication between the main SoC and off-chip memory (e.g. DRAM). This is typically modelled as either an attacker replacing a DIMM with a malicious one that can leak or replay data, or exfiltrate a DIMM for later analysis without losing the volatile contents (eg. by freezing it); the solution is to encrypt and perhaps cryptographically integrity-protect DRAM contents. However, an attacker may also be capable of attaching a fine grained power monitoring or fault injection apparatus to the system. There are techniques available to hardware designers to reduce power side channel leakage below practically observable levels for certain operations, but these require cooperation with software actuating the hardware.

Even without physical access confidential information can leak through timing, and designing hardware and software such that it doesn't leak secrets through timing is a well-established discipline in cryptography—the "constant time" implementation style. Correctly writing constant time software is widely regarded as difficult even within the cryptography community, and generally comes at a performance cost. An interesting area to explore is the combination of novel compilation technology and hardware features to reduce leakage, to help developers understand leakage, and to make informed performance/leakage trade-offs. We are encouraged by recent progress in *data-oblivious* data structures and algorithms (e.g. [23]). Relatedly, *Oblivious RAM* (ORAM) is a known approach to masking memory access patterns, and hardware-based ORAMs have been reported (e.g. [24]). Currently, all such schemes have overheads that are too high to be applied to all memory [25], but can be applied to smaller regions. Use of small hardware-enabled ORAMs, to protect critical parts of programs, is also an interesting area for future work.

B. Behavioural measurement and attestation

Remote attestation, used correctly, enables a relying party to establish that the software running in an isolate is as expected. However, it does not enable the relying party to trust the software, by itself, and must be combined with a separate, out-of-band establishment of trust in the software—typically by access to source code, manual audit, reproducible build, and comparing hashes. Some applications for Confidential Compute are not conducive to out-of-band trust establishment, for example because the computation is proprietary or confidential, or because it is bespoke or one-off and the relying party does not have the resources or expertise to manually audit the code. The next logical step is to enable a relying party to establish that isolated software acts in certain ways they care about, for example that no information flows to defined outputs exist, or that the program is functionally safe (i.e. no Heartbleed-style [26] vulnerabilities are present) to mention but two. *Behavioural attestation* allows a relying party to establish trust in behavioural properties of programs while keeping the programs themselves secret. In the general case, concrete and practical ways of doing this do not yet exist, however we are encouraged by recent progress towards similar goals in the cryptocurrency and smart contracts community (for instance using *Zero-Knowledge Proofs*), and hope the next generation of Confidential Computing systems adopt similar techniques.

C. Sandboxing and software compartmentalization

Isolates protect their contents from the host system, but the converse—protecting the host system from the isolate contents—is not necessarily provided. To facilitate pervasive usage of strong isolation, this protection is necessary, as an isolate prevents a host system from using conventional anti-malware protections to detect viruses, for example.

In *Veracruz* we use WebAssembly as a sandbox for guest programs. Because WebAssembly is primarily intended for running code in a web browser both the specification and implementations are designed with a high level of robustness. WebAssembly also has additional advantages as a sandbox, such as providing program portability and providing a widely supported compiler target; however it also has a number of disadvantages, including: i) reduced performance compared to native code, ii) interpretation and JIT compilation make it very difficult to write code that avoids side channel leakage (see CT-Wasm [27] for promising proposals), iii) it currently presents programs with a single contiguous memory, complicating common system interface patterns (`mmap` etc), and iv) it is not a good compilation target for all languages due to lack of direct support for arbitrary control flow graphs.

While WebAssembly is currently the right choice for Veracruz, we anticipate interest in other sandboxing approaches, and especially in sandboxing native code. We are excited by the potential of CHERI and the Morello project [28] which both experimentally extends the Arm AArch64 architecture with CHERI [29], and implements that experimental architecture in a high performance server-class processor. CHERI adds a hardware capability system to the instruction set wherein, with compiler assistance, pointers are replaced by *capabilities*—tagged fat pointers carrying metadata such as bounds and permissions——allowing a large portion of common memory safety vulnerabilities, for example buffer overflows, to be avoided. Empirical historical evidence from industrial codebases from Microsoft and Google Chromium

[30], [31] suggest about 70% of patched vulnerabilities are due to memory safety problems; CHERI promises to address many such vulnerabilities, so the potential dividend is high.

Whilst the Morello project's experimental processor does not support Arm CCA, the possibility of hardware combining CHERI and Confidential Compute support is interesting. CHERI significantly reduces the number of exploitable vulnerabilities present in the isolate software or in the system TCB, and the hardware capability system can also be used for efficient and robust sandboxing and compartmentalization of native code, with access to all ISA features (such as SIMD) as well as controlled access to other system resources such as memory mapped accelerators, with appropriate OS support.

D. High-level policies

As mentioned in Section III, Veracruz is parameterized by a *policy*, providing all participants with information about the provenance of data, software, and who will have access to the results. Whilst sufficient for Veracruz's use-cases, we anticipate that as distributed, and potentially autonomous, systems at a larger scale utilize Confidential Computing infrastructures, the coordination and agreement of capabilities, trust relationships, and attestation evidence will become increasingly more complex. Thus, the design of policy languages, protocols, and mechanisms for scalable establishment of mutual trust between independent agents, is required to realize our vision.

E. Next-generation attestation

Remote attestation is a central part of Confidential Compute, but in its current form it is not perfect. Current protocols only give the relying party confidence in the initial state of an isolate, relying on the transitive assumption that known initial state and known code can only produce expected executions. In an ideal world, with perfect software in the isolate, this is a good assumption. However real software inevitably contains bugs and vulnerabilities, and especially for long running or interactive computations such as stateful network servers, the relying party should regard the isolate with less confidence. It is of course possible to measure (hash) the isolate's state during a computation, but it is very hard for a relying party to interpret those measurements in all but the very simplest cases. Further work on attestation techniques that offer more flexibility and higher assurance, especially to long running and interactive computations, and to isolates that go beyond purely CPU-bound computations—that is to say, use accelerators or heterogeneous compute fabrics—is needed. While we participate in several efforts towards standardization of attestation protocols and implementation of attestation infrastructure [10], [12], we believe this is still an open problem.

F. End-to-end verification

The security posture of Confidential Compute demands high-quality assurances for hardware, software and communication protocols. While the formal methods community has made great progress in verifying complex and critical systems (see [3], [4], [32], [33] for a few examples), we believe that a full end-to-end application of formal methods to a Confidential Compute stack cannot be expected in a short-to-mid term. However, we believe that wherever practical, existing techniques should be used in lieu of future tools that will extend strong guarantees across layers of the stack—the inability to have a completely verified system should not be an excuse for not verifying as much as is possible.

G. Cryptographic Confidential Compute

With future efficiency improvements, cryptographic approaches to Confidential Compute—Fully Homomorphic Encryption (FHE) and Secure Multi-Party Computations (MPC), for example—will inevitably become attractive. The DARPA Data Protection in Virtual Environments (DPRIVE) program aims to improve performance of FHE using large scale hardware acceleration to within one order of magnitude of the performance of running the computation unprotected [34]; that is, the program aims to accelerate FHE computations by five orders of magnitude or more. FHE and MPC also restrict the computational model in various algorithm-dependent ways, making it more or less applicable to different computations. But, assuming such improvements are successful, further development and commercialization of hardware accelerators and algorithms may enable FHE to eventually displace isolate-based Confidential Compute in many applications. Cryptographic Confidential Compute is, in principle, desirable since it eliminates the need for trusting hardware, performing attestation or (in some cases) addressing side channels, reducing security arguments to purely cryptographic ones. Even with today's performance, FHE and MPC are starting to find application today, especially where the computation is small and the number of participants is high, such as aggregating survey results or attributing online advertising spend. In such cases, multiple attestations, isolate set up and teardown, and other overheads can make cryptographic approaches performance competitive with today's notions of Confidential Compute.

V. Conclusions

Arm has taken a number of small steps toward realizing the vision outlined in Section II, for example by introducing a new architecture for Confidential Compute, called Arm CCA. Building on top of, and alongside Arm CCA and other related isolation and security technologies, we are investigating new ways of protecting computation wherever it happens, potentially in collaborative settings between mutually distrustful parties, with projects such as IceCap and Veracruz.

With this position paper, we hope to show that emerging technologies like Arm CCA are more than mechanisms for protecting existing workloads but represent interesting new primitive building blocks through which new ways of controlling data, on a fine-grained basis, can be built.

Responsible use of these new primitives will, we hope, significantly improve privacy and security, enable a next generation of trustworthy, respectful systems and services, and perhaps even begin to tackle some of the social and political problems caused by insecurity and inauthenticity online.

References

[1] T. Anderson – The Register, "Hyper-V bug that could crash 'big portions of Azure cloud infrastructure': Code published," https://www.theregister.com/2021/06/02/hyperv_bug_that_until_recently/, 2021.

[2] Xen Security Advisory, "Improper MSR range used for x2APIC emulation," http://xenbits.xen.org/xsa/advisory-108.html, 2021.

[3] G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood, "seL4: Formal verification of an OS kernel," in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, ser. SOSP '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 207–220. [Online]. Available: https://doi.org/10.1145/1629575.1629596

[4] R. Gu, Z. Shao, H. Chen, J. Kim, J. Koenig, X. N. Wu, V. Sjöberg, and D. Costanzo, "Building certified concurrent os kernels," *Commun. ACM*, vol. 62, no. 10, p. 89–99, Sep. 2019.

[5] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, 2009, pp. 169–178. [Online]. Available: https://www.cs.cmu.edu/~odonnell/hits09/gentry-homomorphic-encryption.pdf

[6] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, 2009. [Online]. Available: https://crypto.stanford.edu/craig/craig-thesis.pdf

[7] Arm Ltd., "Arm TrustZone Technology," https://developer.arm.com/ip-products/security-ip/trustzone, 2021.

[8] M. Holler, M. Odstrcil, M. Guizar-Sicairos, M. Lebugle, E. Müller, S. Finizio, G. Tinti, C. David, J. Zusman, W. Unglaub, O. Bunk, J. Raabe, A. F. J. Levi, and G. Aeppli, "Three-dimensional imaging of integrated circuits with macro- to nanoscale zoom," *Nature Electronics*, vol. 2, no. 10, pp. 464–470, 2019. [Online]. Available: https://doi.org/10.1038/s41928-019-0309-z

[9] Intel, "Attestation service for Intel® Software Guard Extensions (Intel® SGX)," https://software.intel.com/content/www/us/en/develop/download/intel-sgx-intel-epid-provisioning-and-attestation-services.html.

[10] Arm Ltd., "Project VERAISON: VERificAtIon of atteStatiON," https://github.com/veraison/veraison, 2021.

[11] Brendan Moran, Hannes Tschofenig, David Brown, Milosch Meriac, "A Firmware Update Architecture for Internet of Things – IETF Draft 24 April 2021," https://tools.ietf.org/id/draft-ietf-suit-architecture-14.html, 2021.

[12] H. Tschofenig, S. Frost, M. Brossard, A. Shaw, and T. Fossati, "Arm's Platform Security Architecture (PSA) Attestation Token – IETF Draft 24 March 2021," https://tools.ietf.org/id/draft-tschofenig-rats-psa-token-00.html, 2021.

[13] C2PA, "Coalition for Content Provenance and Authenticity," https://c2pa.org, 2021.

[14] Arm Ltd., "Arm Confidential Compute Architecture," https://developer.arm.com/architectures/architecture-security-features/confidential-computing, 2021.

[15] Arm Ltd., "Arm's solution to the future needs of AI, security and specialized computing is v9," https://www.arm.com/company/news/2021/03/arms-answer-to-the-future-of-ai-armv9-architecture, 2021.

[16] Arm Ltd., "IceCap: Trustworthy virtualization based on seL4, the formally verified microkernel," https://gitlab.com/arm-research/security/icecap/icecap/, 2021.

[17] T. Sewell, S. Winwood, P. Gammie, T. C. Murray, J. Andronick, and G. Klein, "sel4 enforces integrity," in *Interactive Theorem Proving - Second International Conference, ITP 2011, Berg en Dal, The Netherlands, August 22-25, 2011. Proceedings*, 2011, pp. 325–340. [Online]. Available: https://doi.org/10.1007/978-3-642-22863-6_24

[18] T. C. Murray, D. Matichuk, M. Brassil, P. Gammie, and G. Klein, "Noninterference for operating system kernels," in *Certified Programs and Proofs - Second International Conference, CPP 2012, Kyoto, Japan, December 13-15, 2012. Proceedings*, 2012, pp. 126–142. [Online]. Available: https://doi.org/10.1007/978-3-642-35308-6_12

[19] T. C. Murray, D. Matichuk, M. Brassil, P. Gammie, T. Bourke, S. Seefried, C. Lewis, X. Gao, and G. Klein, "sel4: From general purpose to a proof of information flow enforcement," in *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, 2013, pp. 415–429. [Online]. Available: https://doi.org/10.1109/SP.2013.35

[20] Confidential Computing Consortium – Arm Ltd., "Veracruz: privacy-preserving collaborative compute," https://github.com/veracruz-project/veracruz, 2021.

[21] A. Haas, A. Rossberg, D. L. Schuff, B. L. Titzer, M. Holman, D. Gohman, L. Wagner, A. Zakai, and J. F. Bastien, "Bringing the web up to speed with webassembly," in *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2017*, A. Cohen and M. T. Vechev, Eds. ACM, 2017, pp. 185–200.

[22] The WebAssembly working group, "The WebAssembly system interface (WASI)," https://wasi.dev, 2021.

[23] V. Ramachandran and E. Shi, "Data oblivious algorithms for multicores," in *SPAA '21: 33rd ACM Symposium on Parallelism in Algorithms and Architectures, Virtual Event, USA, 6-8 July, 2021*, K. Agrawal and Y. Azar, Eds. ACM, 2021, pp. 373–384. [Online]. Available: https://doi.org/10.1145/3409964.3461783

[24] C. W. Fletcher, L. Ren, A. Kwon, M. v. Dijk, E. Stefanov, D. Serpanos, and S. Devadas, "A low-latency, low-area hardware oblivious ram controller," in *2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines*, 2015, pp. 215–222.

[25] G. Asharov, I. Komargodski, W.-K. Lin, K. Nayak, E. Peserico, and E. Shi, "Optorama: Optimal oblivious ram," Cryptology ePrint Archive, Report 2018/892, 2018, https://eprint.iacr.org/2018/892.

[26] F. Schwartzenburg, W. Oates, J. Park, D. Johnson, M. Stutzman, M. Bailey, and S. Youngblood, "Verification, validation, and accreditation (vv&a): one voice — unified, common & cross-cutting," in *Proceedings of the 2007 Summer Computer Simulation Conference, SCSC 2007, San Diego, California, USA, July 16-19, 2007*, G. A. Wainer, Ed. Simulation Councils, Inc., 2007, pp. 429–436.

[27] C. Watt, J. Renner, N. Popescu, S. Cauligi, and D. Stefan, "CT-Wasm: type-driven secure cryptography for the web ecosystem," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, p. 1–29, Jan 2019. [Online]. Available: http://dx.doi.org/10.1145/3290390

[28] Arm Ltd., "Morello – Arm Developer," https://developer.arm.com/architectures/cpu-architecture/a-profile/morello, 2021.

[29] R. N. M. Watson, P. G. Neumann, et al., "UCAM-CL-TR-951: Capability hardware enhanced RISC instructions: CHERI instruction-set architecture (version 8)," https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-951.pdf, University of Cambridge Computer Laboratory, Tech. Rep., 2020.

[30] MSRC Team, Microsoft Inc., "A proactive approach to more secure code," https://msrc-blog.microsoft.com/2019/07/16/a-proactive-approach-to-more-secure-code/, 2019.

[31] Google Inc., "Memory safety - The Chromium Projects," https://www.chromium.org/Home/chromium-security/memory-safety, 2021.

[32] M. Polubelova, K. Bhargavan, J. Protzenko, B. Beurdouche, A. Fromherz, N. Kulatova, and S. Z. Béguelin, "Haclxn: Verified generic SIMD crypto (for all your favourite platforms)," in *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, J. Ligatti, X. Ou, J. Katz, and G. Vigna, Eds. ACM, 2020, pp. 899–918. [Online]. Available: https://doi.org/10.1145/3372297.3423352

[33] S. Li, X. Li, R. Gu, J. Nieh, and J. Hui, "A secure and formally verified linux kvm hypervisor," in *2021 2021 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2021, pp. 1782–1799.

[34] DARPA, "Data Protection in Virtual Environments (DPRIVE)," https://www.darpa.mil/program/data-protection-in-virtual-environments, 2020.