---

slug: ZTEE title: Zero Trust Execution Environments authors: [quintus, sbellem, jopasserat, amiller, friends] tags: [TEE, secure hardware, Zero Trust, hardware, trojan, side channel] hide_table_of_contents: false forum_link: https://collective.flashbots.net/t/zero-trust-execution-environments/3966?u=quintus

---

Recent months have seen an awakening of several large use cases of TEEs, fromOpenAI calling for new and improved TEEs and Apple announcing a hardware-based private cloud, to Ethereum transactions being privately processed in TDX and other blockchains leveraging SGX for integrity. Having worked quite closely with several use cases that are at the center of this excitement, we are aware of both the promise which secure hardware holds, and of the current insufficiency of today's hardware to service these use cases. TEE solutions currently on the market simply do not offer sufficient security guarantees while alternative secure hardware families like TPMs, secure elements and smart cards offer better (although not perfect) security guarantees at the cost of needed performance and functionality. For the first time, there is substantial industry demand for high-power, high-performance hardware that is secure under a comprehensive threat model.

This post is the first in a series that aims to unlock these use cases by getting the right people designing hardware that is fit for this new generation of applications. Our first objective is to convey to hardware security experts with the right skillsets but little context on the use cases that this is a project worth undertaking. Our second objective is to provide the details required to arrive at the necessary hardware. This involves defining what "good enough" means in terms of security, performance and functionality. It also involves pointing out tradeoffs, open problems and leading approaches to these problems. Realising the vision we lay out requires inputs from many different experts both on the software and hardware side, we try to give enough context to accommodate wide backgrounds.

This post will introduce some of the use cases that stand to benefit most from improvements in secure hardware, explain why secure hardware is an essential component in serving these use cases and give the 100,000ft explanation of why the secure hardware we have today doesn't satisfy our needs.

Here we summarise some reasons you ought to care about this project. We expand on these later.

- Many of the requirements of this new hardware are novel and consequently presen**interesting new challenges** with new affordances (e.g. power and cost are much less of a concern), new constraints (e.g. higher performance requirements) and new adversarial models. They also provide motivation for past research that has been left dormant.
- The industries and use cases we are pointing to are large and present**a good business opportunity**. Furthermore, the technology we are calling for is fundamental so unanticipated use cases will likely rear their head. Most secure hardware products have been aimed at low-cost, low-power use cases like sim and credit cards while higher cost and higher power chips have proliferated but without providing similar levels of security.
- This work can be very **good for the world**. Improving security or reducing its cost makes it easier for distrusting parties to interact. This has implications that are technological (e.g. you train your AI on my data), economic (e.g. lower barriers to entry or costs to form agreements) and political (e.g. identifying deepfakes, obviate legal enforcement). Much of it also extends past the use cases we mention, to the most basic functionalities of simply storing cryptographic material.[^1]

A note on definitions

Throughout this series we will refer broadly to secure hardware (SH) to avoid the distinctions between TEEs, TPMs and the likes as this categorisation is not directly relevant for most discussions. **The key functionality that we are interested in is publicly verifiable remote attestation (RA)**. To quote from Intel SGX Explained:

> *Secure remote computation is the problem of executing software on a remote computer owned and maintained by an untrusted party, with some integrity and confidentiality guarantees.*

As is often the case, we require this attestation to be accompanied by public key cryptographic functionality so that remote users can interact with the attested program through a secure channel.

Use Cases

Before getting into the hardware details, lets first look at why anyone would want to use this new secure hardware in the first place.

While we are most familiar with the "crypto"/web3 setting, we highlight example use cases across several industries. Many of the fundamental security improvements required serve all of these use cases. We recognise that there are important differences between these use cases and we will discuss these in future blog posts.

Financial Exchange

Venues for financial exchange process trillions of dollars of economic activity on a monthly basis. The operational details vary by venue, but the overall structure remains the same: multiple users input offers to buy or sell an asset to a single system with one user's outcome being highly dependent on the inputs and ordering of inputs from other users. Even slight

influence over the operation of these markets or insight into confidential trading information can be highly profitable for those who benefit and highly costly for those who don't [1, 2, 3]. For basic intuition, consider a seller who is able to raise a price once they know that there is a willing purchaser of a large quantity of an asset because this seller also operates the exchange venue.

Traditional approaches to securing these markets have been based on legal enforcement, but legal enforcement techniques are limited by our ability to detect misbehaviour, provide friction for international trade that crosses regulatory boundaries, and increase barriers to entry (e.g. licensing fees and reporting overheads), reducing market efficiency. In recent years, the "decentralised finance" (DeFi) industry has developed techniques for enforcing faithful operation of markets through technological means. While these efforts have shown promise - such as success in smoothing frictions to international trade with blockchain-based methods [1, 2, 3] - some critical problems remain unsolved with participants in leading decentralised exchanges losing upwards of hundreds of millions of dollars due to insufficient privacy measures (e.g. sandwich attacks). These problems have lead to several proposed architectures and live systems based on TEEs.

Verifiable inference and model routing

Machine learning model inference currently relies on deploying models behind an API hosted on a public cloud computing service, with usage billed per query. Users can select from multiple models with varying performance levels and data retention policies, and are charged accordingly. Expensive models often offer superior prediction and generation capabilities, but end-users struggle to discern differences between models. Malicious providers could exploit this information asymmetry by charging high prices for supposedly superior models and conservative data policies that are, in reality, degraded versions costing less to run. Hosting API services in TEEs allows users to audit the routing algorithm implemented by the remote service, selecting the appropriate model and verifying that the advertised data collection policy is enforced. If the router is proprietary and the provider wants to protect their intellectual property, logging a model fingerprint via the TEE application controlling the routing could be a middle ground. This would enable users to verify model consistency over time or compare models across accounts to ensure fairness and non-discrimination against a specific population.

Outsourced model training

As AI increasingly relies on raw processing power to build larger models, the limited availability of the latest Graphics Processing Units (GPUs) has driven the usage of 3rd party cloud infrastructure to train larger and more complex artificial intelligence (AI) models. This practice introduce significant security concerns, particularly when dealing with sensitive user data and highly valuable model weights. Secure Hardware (SH) offers a solution by isolating the training process and the raw data, mitigating the risks associated with untrusted 3rd party cloud providers, and providing assurances to both the model owner and data providers.

Proposed solutions in this direction include the further development ofGPU-TEEs to meet the performance requirements of large model training, as well as the combined use with existing techniques like federated learning.

Beyond their increased security guarantees, SH solutions have the potential to unlock previously inaccessible datasets. For example, in the healthcare space, strict regulatory requirements make it difficult for hospitals to share raw data. SH can enable secure processing of sensitive data within enclaves located beyond primary IT infrastructure, facilitating collaboration on AI projects while protecting raw data behind firewalls.

A useful case study is highly-sensitive genomics data. As early as 2017,SGX-based whole genome variation searches were proposed, later including genome imputations and privacy-preserving machine learning. Commercial solutions, use secure hardware not only to confidentially compute over this sensitive data, but also as part of solutions to provide secure storage, ensuring data sovereignty and uniqueness.

Content Authenticity and Provenance

The Coalition for Content Provenance and Authenticity (C2PA) standard is being promoted as a solution to combat fake news and deep fakes in an era of inexpensive generative AI services and a surge in news providers. C2PA equips cameras with signing oracles that bind an image's original version to its originating device, allowing for tamper-proof image capture. However, further processing, as often required by online news publishers to at least provide a downscaled version of the image, will break the cryptographic bind.

Images may also need to go through rescaling, cropping, exposure adjustments and the likes, even without malicious intent. Zero-knowledge proof (ZKP) solutions have been proposed to address this issue but will incur a high overhead for proof generation and are currently limited to simple post-processing. TEEs can support more elaborate processing pipelines, non-interactively applying transformations based on common image processing software, while maintaining the integrity of the pipeline via the attestation mechanism.

Secure Delegation

Account delegation is a powerful primitive: it's used to give web services limited access to accounts, such as "Login with Github", or for users to share accounts without having to give up their authentication secrets (e.g. password). Most digital systems offer only limited options for delegation. For example, Twitter has two authorization levels: Read-only, and Read/Write. An app that requests Read-only can't post at all, but an app with Read/Write can post indefinitely… as well as

change the username and profile photo. What if you only wanted to allow a service to post just once?

Using TEEs to store passwords, session cookies, and other account credentials can allow users to delegate access to these. The idea was first proposed in 2018 for law enforcement use cases, and can be seen more broadly as a way to empower users with fine grained control over their data and digital capabilities without sacrificing the efficiency of the platform.

Why secure hardware?

Mom, can we have some secure hardware?

We have listed some high-level use cases that benefit from secure hardware and only briefly mentioned why secure hardware and not some other security tool should be employed. In this section, we tackle the latter question directly.

Explicit calls from leaders in the AI and decentralised finance communities for the improvement and use of secure hardware may be justification enough for some, but we feel it necessary to expound upon the differences (and complementarities) with related technologies to more clearly highlight the most salient properties of secure hardware.

The technologies most fit for comparison are:

- Multi-Party Computation protocols (MPC)
- Fully Homomorphic Encryption (FHE)
- SNARKs and STARKs which we classify together under Zero-Knowledge proof schemes (ZKPs)[^4]
- Blockchains

SH is differentiated among these technologies in performance and unique security guarantees, in some cases, constituting a unique solution and in other cases unlocking a novel hybrid approach.

Performance

Some use cases demand efficiency that only secure hardware can provide either in cost or latency. Naturally, much nuance is lost in comparing these different technologies to each other, especially without fixing a specific use case, so the following metrics should be considered loose estimates:

- The ZKP state of the art (for a SHA256 hash) can be estimated to be5,000x slower than Intel TDX (0.5 MHz vs 3 GHz). [^2]
- For FHE, latency overhead ranges in the 4 to 6 orders of magnitude depending on whether you're customising your application to fit FHE's constraints, or whether you employ a more naive approach. The expectation is for hardware acceleration to shave another 2 orders of magnitude off that in the next few years.
- For MPC, benchmarks are particularly challenging because latency depends on heavily on the number of parties, round trip time between those parties and available bandwidth. Choosing the number and dispersion of participating parties is a factor in determining system security. A rough estimate would be 1 to 4 orders of magnitude, also depending heavily on application specificity. We are not aware of MPC implementations that are of comparable complexity to some of the SH use cases we are interested in

The curious reader can have a look at some of the (unoptimised) work we have done applyingFHE and MPC to the decentralised finance setting or these benchmarks[^6].

Security Guarantees

Some use cases benefit from SH guarantees that are not attainable using only the listed software-based techniques. This has been well studied from a theoretical perspective. For example, one-shot signatures (or more generallyone-time programs) are known to only be realizable through hardware-based guarantees or quantum cryptography[^7].

The unique security properties of SH can also be practically understood with the observation that we don't currently have any way of controlling how a party uses cryptographic secrets they have access to. For instance, we might want a system which takes encrypted inputs from many users and then computes some joint function over the inputs with only the output being learnt by anyone (a concrete example). We could use homomorphic encryption, but this leaves the question of who holds the decryption key. If we leave one party with control over the decryption key, what stops them from peeking at the inputs or intermediate computation? The standard answer is to split the decryption key between many parties, but this in turn relies on the assumption that a certain number of members of the committee are acting honestly.

Enforcing the privacy of the system through secure hardware can remove the need to make this threshold-honesty assumption. Of course, SH also introduces its own set of assumptions. These cannot be ignored and we will address them in depth. For now, let it suffice to say that these assumptions are appealing because (1) they can be weakened through technical progress, (2) a combination of SH with other techniques presents a very robust security model (as is already being done).

This kind of honesty-threshold assumption can also be found in public blockchains networks and MPC schemes and can similarly be obviated by SH. For instance, digital currencies, the current largest use case of blockchains, can be

implemented without blockchains given some SH or quantum crypto. Blockchains can also make use of SH that enables certified deletion to address "long range attacks." The idea is that by requiring provable deletion of keys, we can prevent adversaries from using old cryptographic material to fool new participants of a system by faking a system history.

As for ZKPs, typically these do not require honesty assumptions but on their own cannot provide privacy guarantees for functions involving inputs from multiple users as the prover must have knowledge of multiple inputs.

None of this is to say that secure hardware should replace the listed techniques or that any of these approaches are mutually exclusive. Instead, SH can be combined with other techniques to offer additional security through defense in depth, or to allow more granular performance/security tradeoffs. In fact we already see the combined use of TEEs and other forms of cryptography in the wild (1, 2, 3, 4). An idea we will explore in future posts is the potential for software cryptography to be used to accelerate SH and not the other way around as is typically discussed.

Why current offerings don't cut it

The secure hardware we have at home…

Why not simply use the secure hardware on offer? Let's look at a score card with three broad classes of adversaries:

- **Remote software adversary**: This is an adversary who does not have physical access to the device. For a large class of SH (typically targeted at key management use cases), this adversary is not particularly threatening because sensitive computations are isolated from untrusted code at the hardware level. However, the story for the most performant SH, server-side TEEs, is different. With SGX as the canonical example[^8], a defining characteristic of these TEEs is that trusted and untrusted code share the same processor and cache. This comes with the additional implication that the processor is not optimised with only secure execution in mind. Over recent years, there have been many attacks that have exploited these architectural features, which should come as no surprise since SGX and other similar models do not even consider software-based side-channels as part of their threat model either[^9].
- **Physical adversary**: this is anyone with physical access to the completed device. High-performance server-side TEEs do not consider physical adversaries as part of the threat model, effectively making the cloud provider a trusted entity[^5]. Mobile TEEs and other forms of SH like secure elements and smart cards do offer some security guarantees against adversaries of varying sophistication (no specialised tools, non-invasive, invasive etc). Unlike server-side TEEs, these usually include security measures such as tamper-detecting meshes and sensors, redundancy for fault injection detection and masking techniques to address side channels. We will discuss these in greater depth in a future post.
- **Supply chain adversary**: this is anyone involved with the design of the chip or who has access to the chip before it is "complete" (i.e. all security measures are in place). With only one exception, the authors are not aware of any SH considers supply chain adversaries as part of their threat model. The majority of supply chain trust efforts have gone to allow different parts of the supply chain to trust each other, such as the designer trusting the foundry. This is importantly different from our goal of assuring the end user. The implication is that the user must assume that the SH does not contain a hardware trojan and, even more simply, the user must trust a key-injection process (i.e. no one has a kept a copy of the hardware keys that were supposed to be secret) and an attestation service (i.e. the service only provides certificates to legitimate SH). Supply chain attacks are not without precedent, with the most recent attack being executed as part of a military operation.

In one sentence, performant SH falls short on all three measures and other forms of SH perform better against physical and remote adversaries, but still don't address supply chain adversaries and do not offer the performance (or functionality) required as they were developed with simple key management in mind. Future posts will take a more thorough look at the ways in which SH falls short and can be improved, but at a high level we have two challenges:

- Applying existing security measures against remote and physical adversaries without sacrificing too much performance or functionality.
- Developing and implementing techniques for removing trust in supply chain actors. These include trojan detection methodology, decentralised remote attestation protocols and obviating key injection.

Naturally there is a question whether all of these problems need to be or can be addressed by a single design. This too we address in a later post - for now we're just listing the problems.

Why the higher bar for security?

- Previous secure hardware use cases were not of the same magnitude of importance. As AI tools become more powerful and more deeply ingrained in every tool we use, the amount of personal data processed by these models will grow to astonishing heights. In the past, client-side computing was able to address this issue with user data staying on the device, but the size and performance requirements of state of the art models requires that user data be sent to the cloud. This is the observation that motivated Apple's Private Compute Cloud announcement. AI models themselves can also be worth extraordinary amounts of money and increasingly are considered of national security interest.

At the same time, decentralised financial exchanges are already processing billions of dollars of value and the full market for financial trade is of astronomical size. In fact, decentralised financial systems are already being attacked by nation states.

Cryptographic identities (e.g. passkeys) are also becoming a ubiquitous mediating access to large parts of human activity as

part of a global identity system. These keys are all stored and used on unverified hardware, meaning very large portions of this identity system can be completely compromised by a single party. Of course this issue transcends the remote attestation requirement and specifically points to the importance of considering the supply chain adversary. - SLAs and other legal agreements have traditionally been used as a means of achieving security. If a TEE provided by a large cloud provider is physically attacked, the cloud provider can be sued. However, the legal action is slow, expensive, limited geographically and limited to large well-capitalised actors. The web3 industry (within which one can categorise decentralised finance), in particular, has focused on obviating the need for such agreements by removing trusted actors.

As a reference point, we can turn to the Ethereum blockchain which is the platform that plays host to the current largest decentralised exchanges. Ethereum has been explicitly designed and implemented to minimise the impact of the failure or misbehaviour of any single party or system. The underlying distributed protocol supports a very high number of nodes and has been parameterised to tolerate nodes in any geographic/regulatory location with resource requirements sufficiently low so that consumer hardware can be used to avoid dependence on cloud vendors. Moreover, there are 6 maintained implementations of the Ethereum client, each written in its own language.

This context helps to understand the need to remove trusted parties from the threat model. Remote attacks are more likely than cloud providers tampering with the machines they host. Similarly, remote and physical attackers are much more worrisome than any trojan in the near future. However, these points do not take into account the dependencies which these vulnerabilities create. Businesses built around SH that is subject to these attack vectors must use one of a few large trustworthy cloud vendors and can only buy chips from the most trustworthy designers and manufacturers (a weakened business position). Some entity must be prepared to take legal action, an expensive and time consuming undertaking and one that assumes that a legal entity even exists - an assumption which does not hold for many of the systems at play. Consider a decentralised protocol that requires SH attestations to participate.

Even when there is an entity who is able to legal action, the ultimate end user who engaged with this entity on the basis of SH assurances - who we assume is not well resourced or in the same country - does not always have the same recourse. Another challenge with trusting supply chain actors or legal systems is that faith in these actors is not geographically universal. A US, Brazilian and Chinese person have very different views on the trustworthiness of Intel, Huawei and the likes and it is thus hard for them all to participate in the same system if that system is based on trust in entities like this. Technical security guarantees on the other hand can be much more universally agreed upon. - Similarly to the previous point, the status quo in SH in which the security guarantees of a system are in large part assured through legal means caters to a “security through obscurity” approach. For example, the majority of tamper resistance measures are not documented publicly and common criteria - through which the security of most hardware is certified - even allocates points for keeping design details private.[^11] This approach does not carry over well to our setting in which it not only matters that the SH is actually secure, but that a 3rd party can be convinced of its security *on a technological basis* without dependence on a trusted authority. Our setting benefits from openness of as many design details as possible, thus the security model cannot rely on obscurity of these details, even if we were to work under the assumption that obscurity leads to greater security. Our next post addresses this topic in more detail. - Purely from a business perspective, there is clear economic interest in security technology of this type. Enormous amounts of funding have poured into the comparable security technologies mentioned in this article. Network participation incentives for the Ethereum and Bitcoin blockchains (a means to guard against the collusive attacks mentioned before) have exceeded $30B and $65B respectively with Ethereum usage fees exceeding $11B over the last two years. Substantial amounts of funding have been allocated tocompanies improving and utilising ZK, MPC and FHE technology, often including hardware-components.

What now?

Ideally, this post convinced you that this is a project worth undertaking. The next post will be dedicated specifically to the issue of supply chain adversaries, why there is reason to believe that something can reasonably be done about them and the problems that need to be solved to do that. From there we will describe the issues of physical and remote attackers and the tradeoffs involved in addressing them. That will have set the scene to look in more detail at existing solutions and some use cases to plot the path(s) towards a more secure internet.

A special thanks to Leo Arias, Zach Belateche, Dan Boneh, the Fabric team, James Ball, the people atSimple Crypto and Bunnie

[^1]: Cory Doctorow’s post makes the counterpoint that security technology is also dangerously a means of control. In some sense, this is a fourth reason to care: to influence the technology we are developing today. [^2]: Alternative benchmarks [^4]: We also consider uses of these techniques that don’t have the ZK property and are only used for succinct proofs of correctness [^5]: one way around this limitation is to put the TEE where no one can reach it - like outer space [^6]: We anticipate the comments section will feature some heated opinions citing useful sources. [^7]: This post is written on the assumption that quantum computers will not be widely adopted for the foreseeable future. [^8]: Intel TXT and AMD SVM are exceptions. [^9]: “Intel SGX is Not the Answer. Unfortunately, … SGX design … fails to offer meaningful software isolation guarantees. The SGX threat model protects against all direct attacks, but excludes “side-channel attacks”, evenif they can be performed via software alone. Alarmingly, cache timing attacks require only unprivileged software running on the victim’s host computer, and do not rely on any physical access to the machine. This is particularly concerning in a cloud computing scenario, where gaining software access to the victim’s computer only requires a credit card” [^11]: To reach CC-EAL-5+, a developer needs to satisfy ALC_DVS.2, which requires security controls to "protect the confidentiality and integrity of the TOE design."