A SUMMARY AND CRITICAL REVIEW

TEE.fail: Breaking Trusted Execution Environments via DDR5 Memory Bus Interposition

Alina Abraham, 750538 (Wildcard)

Abstract

Trusted Execution Environments (TEEs) are a commonly used technology to provide strong security guarantees for sensitive data, running on an untrusted environment, which could be a compromised operating system or hypervisor. This security property is especially attractive for cloud environments, as it prevents cloud providers from also accessing sensitive information. Utilizing this advantage for cloud-scale, TEEs have evolved from protecting only small parts of a system to safeguarding whole virtual machines. This required significant architectural changes favoring scalability and performance over strong security guarantees. This paper summarizes the findings of TEE.fail, a study that demonstrates a ciphertext-based side-channel attack on TEEs that, assuming the environment's security guarantees were true, would be impossible. Exploiting deterministic memory encryption combined with weak integrity protection, enables the attacker to steal attestation keys and thereby break the entire trust foundation that TEE implementations rely on. The paper highlights the dangerous mismatch between the assumed threat models and the deployed system design, which violates the security that confidential computing attempts to offer and thus erodes trust in cloud environments.

Keywords: Trusted Execution Environments, Confidential Computing, Memory Encryption, Remote Attestation, Threat Models, Side-Channel Attacks

1. Introduction

Modern cloud environments are shared and multi-tenant infrastructures that aim to provide scalability and elasticity while simultaneously offering high security guarantees. Finding a respectable balance between these two properties poses a though challenge, which has led Trusted Execution Environments (TEEs) to become central building blocks of this domain (*Costan and Devadas 2016; Intel Corporation 2023b*).

Using TEEs, trust is shifted from software to hardware, which resultantly takes away the OS/hypervisor privileges to enforce memory isolation. This has the desired effect, that making use of TEEs in cloud computing promises strong isolation, and thus confidentiality, of sensitive data even in the presence of a malicious host. This feature allows cloud providers to promise strong security guarantees and ensures users that their data is not readable by the service that handles it.

Looking back, TEEs were originally designed to protect small workloads and not to isolate entire virtual machines, which requires a significantly more resource-intensive approach to provide the same level of protection. While this shift does improve usability and adoption, it also introduces new scalability challenges that result from the emerging requirements of scaling up to cloud level. To meet these new requirements, the most impactful modification lies in changing the memory protection mechanism.

The paper *TEE.fail* (*Schlüter 2024*) investigates the security consequences of this transition. In particular, it demonstrates how the architectural changes for scalability can be exploited to re-enable ciphertext-based attacks and thereby impersonate real TEEs even on fully patched systems.

By breaking this trust chain, the paper highlights the importance of aligning threat models with evolving system architectures.

2. Technical Context

2.1 Trusted Execution Environments

To understand the core idea of the paper, it is essential to first introduce Trusted Execution Environment (TEE). A TEE is an isolated execution environment enforced by the processor, that is designed to protect code and data even if the OS were to be compromised, thus shifting trust from software to hardware. The environment runs alongside normal software but has stronger security guarantees.

Core Security Guarantees of TEEs. To provide confidentiality TEEs ensure that data is not readable by the OS, a hypervisor, or other VMs/processes. This is achieved by using hardware memory encryption and isolation enforced by the CPU, which is applied both to data at rest, meaning in memory, and during execution. The second core principle is integrity. TEEs provide this by ensuring that code and data cannot be modified without detection, which protects the data against tampering (unauthorized changes of data in transit or at rest) and replaying old memory contents. Adding these integrity concepts, encryption turns into a robust security guarantee.

SGX enclaves. Intel's implementation of a TEE at the application level is the Software Guard eXtension (SGX). Enclaves only isolate small memory regions within a process, so not the whole OS or VM is protected. They are designed for protecting specific code paths and small amounts of sensitive data. To offer the confidentiality guarantees, in main memory a dedicated region, known as Processor Reserved Memory (PRM), is allocated for enclaves and inaccessible to the kernel (*Intel Corporation 2023a*). Overall, SGX offers precision and strong isolation at small scale.

TDX confidential VMs. Intel's Trusted Domain Extension (TDX) extends TEE concepts to entire virtual machines. This confidential VM (CVM) guarantees that the machine's entire memory and CPU state is protected from the host, meaning that even the hypervisor is considered untrusted (*Intel Corporation 2023b*). Compared to enclave-based TEEs, the amount of protected memory increases significantly, when confidentiality is applied at the scale of a full VM. Providing strong integrity would require the CPU to detect and verify modifications in encrypted memory continuously, which brings on substantial overhead and conflicts with the performance requirements of cloud environments.

2.2 Memory Protection

An important aspect of TEEs is how data is protected in memory. To recap, TEEs try to protect sensitive data while it resides in main memory, ensuring that plaintext data is never exposed on the memory bus. This way secrets are protected after they enter the TEE. Hardware memory encryption is almost always implemented deterministically to support fast, random access and to tolerate crashes (*Schlüter 2024*). Deterministic memory encryption means that the same plaintext written to the same physical address incessantly results in the same ciphertext.

Providing confidentiality alone through encryption is insufficient for secure memory. Integrity and replay protection can be implemented using Merkle trees and freshness counters (*Costan and Devadas 2016*), which make it possible for the CPU to verify integrity metadata on memory access.

Shifting Protection to CVM-Scale. When TEEs were just used at the application level, only relatively small memory regions needed protection, making strong guarantees combining confidentiality, integrity and replay protection feasible. Extending TEE protection to entire VMs, integrity mechanisms based on Merkle trees incur major efficiency loss. As a result, server-scale TEE designs prioritize scalable confidentiality, while integrity and replay protection are reduced in shared memory protection mechanisms used across modern TEE implementations. This means that although this change was motivated by CVMs, the design choices affect hardware and attestation components used by both enclave-based and VM-based TEEs.

2.3 Attestation and Trust Establishment

Because TEEs are often used in cloud environments or remote execution scenarios, the party providing the hardware is not necessarily trusted. Therefore, a remote user must be able to verify that a genuine TEE is running, ensuring that any secrets they provision remain confined within the TEE. This task is addressed by remote attestation. The attestation protocol is executed between a TEE, acting as the prover, and a remote user, acting as the verifier. The TEE produces a statement, called a *report*, which contains information about its identity, configuration and initial state. This statement is then cryptographically signed and transformed into a *quote*, which can be verified by the remote party.

Intel DCAP attestation flow. For SGX and TDX, Intel uses Data Center Attestation Primitives (DCAP), which offers a standardized attestation workflow, certificate chains and verification infrastructure (*Intel Corporation 2022*). SGX as well as TDX share the same attestation ecosystem thereby establishing a common trust root. *Figure 1* shows a diagram, illustrating how the parties interact in Intel's attestation flow.

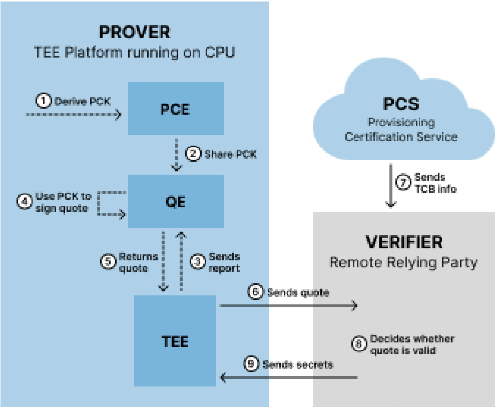

Figure 1. High-level overview of Intel's DCAP attestation flow, showing how reports are transformed into quotes using platform-specific attestation keys and verified by a remote relying party.

Upon the CPU's initialization the Provisioning Certification Enclave (PCE) derives the Intel-signed Provisioning Certification Key (PCK), which is used by the Quoting Enclave (QE) sign attestation evidence. When a TEE produces a report describing its state, the QE transforms this report into a remotely attestable quote by signing it with the PCK. This quote can than be send to a remote verifier to prove that they are interacting with a genuine TEE. The remote verifier checks the quotes validity by verifying the signature chain to Intel's trust root and by evaluating the platform's status using Trusted Compute Base (TCB) information obtained from Intel's Provisioning Certification Service (PCS). If the verification succeeds the verifier trusts the prover's TEE identity and provisions secrets.

Fully trusted status. Remote verifiers often provision secrets only to TEEs in fully trusted state. This state implies that the TEE is running up-to-date microcode, contains no known revoked components and presents valid certification. Whether a TEE is considered fully trusted is a policy decision derived from the attestation results and serves as the basis for trust establishment by the remote verifier.

3. Threat Model

One of the key ideas of the paper is the existence of a mismatch between assumed and actual threat models. A threat model specifies under which assumptions a systems guarantees are expected to hold. This includes naming which attackers are considered, what capabilities they have and which attacks are explicitly out of scope (*OWASP Foundation 2023*). As a result, security guarantees are always conditional to the threat model, meaning that a system can be "secure" while still being vulnerable to attacks that violate its assumed threat model.

TEE Threat Model. For TEEs, guarantees such as confidentiality, integrity and attestation are only valid under specific assumptions. In enclave-era TEEs, the attacker was assumed to be unable to observe or manipulate encrypted memory contents undetected. This assumption was enforced through strong integrity and replay protection mechanisms (explained in *Section 2.2*), which ensured that deterministic memory encryption could not be exploited. When shifting to CVM scale, integrity guarantees are weakened, while the underlying threat model remains unchanged. As a result, a core assumption of the original threat model, namely that encrypted memory cannot be meaningfully manipulated or replayed, no longer holds. This mismatch between the original threat model and the CVM-oriented architecture re-enables ciphertext-based leakage, key extraction, and forging attestation.

4. Attack

4.1 Attacker Setup and Capabilities

Setup. The paper assumes a powerful attacker with control over the host system. The attacker can run arbitrary software on the host and modify kernel components. In addition, using inexpensive equipment and hobbyist-level hardware skills, the attacker is capable of gaining physical access to the memory subsystem (*Schlüter 2024*). This enables the attacker to interpose on the memory bus and observe encrypted memory traffic between CPU and DDR5 DRAM. The system is assumed employ deterministic and address-depended memory encryption and to provide weakened integrity and replay protection. *Figure 2* visualizes the setup.

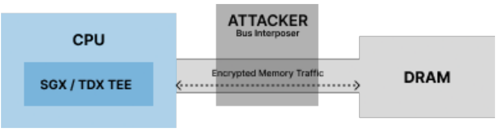

Figure 2. Abstract attacker setup.

Goal. The attacker's goal is to extract the platform attestation key (PCK), which enables the forging of attestation quotes and, consequently, the impersonation TEEs. It is important to note that the attacker doesn't aim to break the specific enclave or CVM but rather to compromise the trust infrastructure.

Threat Model. With these capabilities the attacker clearly violates the original threat model assumptions. However, the attackers actions remain realistic in CVM deployments. The attack does not rely on bugs or misconfiguration but can be attempted at a target system in fully trusted state.

4.2 Control of Memory Placement and Execution

Memory Placement. Because the attacker interposes on only a single DRAM channel, effective exploitation requires control over where sensitive data is placed in physical memory. Modern DRAM architectures, such as DDR5, distribute memory accesses across multiple channels, meaning that only data mapped to the observed channel can be analyzed and secrets that on unobserved channels are missed. To overcome this limitation, the attacker reverse engineers the mapping from physical addresses to DRAM locations. By reverse engineering the mapping of physical memory to virtual location the attacker can choose the channel that covers the most important information. In addition, the kernel privileges of the attacker make it possible to pin specific pages, and thus sensitive TEE data, to the observed memory channel.

Controlling Execution. Because observing ciphertext alone is noise, the attacker must know when a sensitive operation occurs. By leveraging host-level execution control and manipulating page permission, the attacker can pause and resume enclave execution at specific points in time. This enables the attacker to align memory observations with relevant computation phases, and therefor increasing the usefulness of the observed ciphertext.

4.3 PCK Extraction

As shown in *Section 2.3*, the Provisioning Certification Key (PCK) is a central element of the attestation trust chain.

By extracting the PCK, a malicious actor forge valid attestation quotes and impersonate a legitimate TEE, thereby deceiving remote verifiers into provisioning secrets. Consequently, extracting the PCK collapses the root of trust on which DCAP attestation relies.

Signing with the PCK. The Provisioning Certification Key serves as the private key in ECDSA signatures used to produce attestation quotes. Each ECDSA signature depends on a long-term private key d (the PCK) and a fresh per-signature nonce k. If the nonce k used for a single signature becomes known, the private key d can be recovered algebraically. During the signing process, the algorithm iteratively processes digits of k and intermediate values H directly depending on those digits. These values are drawn from a small, known set and are written to memory during computation.

Building a ciphertext dictionary. Deterministic memory encryption, combined with missing integrity and replay protection enables the attacker to write known plaintext values to memory and observe corresponding ciphertext. Exploiting this, the attacker writes all possible values of H to an observable memory location and records the resulting ciphertexts. This produces a dictionary that maps observed ciphertexts to the corresponding H values and, by extension, to the associated digits of k. Such a calibration phase would be infeasible under strong integrity and freshness guarantees.

Recovering k. By triggering a legitimate attestation operation, for example QE certification, the attacker causes the Provisioning Certification Enclave to perform a real ECDSA signature using the secret PCK. Pausing execution at carefully chosen points, the attacker observes the encrypted memory accesses corresponding to H and matches the observed ciphertexts against the previously constructed dictionary. Each observation reveals one digit of k, allowing the attacker to recover the entire nonce bit by bit.

Recovering PCK. Once the attacker has obtained the ECDSA signature and the corresponding nonce k, the private key d, being the Provisioning Certification Key, can be computed directly. At this point, the attacker possesses the platform's attestation key and can generate fully valid SGX and TDX attestation quotes without running a genuine Trusted Execution Environment.

5. Impact

5.1 Relevance of Breaking Attestation

As discussed in earlier sections, attestation enables remote parties to decide whether a TEE can be trusted. Breaking attestation therefore removes the verifier's ability to distinguish between a genuine TEE and an attacker-controlled environment. This is particularly critical because forged attestation quotes are cryptographically valid and pass all standard verification checks. As a result, remote verifiers may provision secrets under false assumptions about the execution environment. Even fully patched systems operating in a *fully trusted* state can no longer provide reliable attestation guarantees. Consequently, the compromise affects not only individual TEEs but the entire trust infrastructure on which confidential computing relies.

5.2 Case Studies

The paper presents several case studies to demonstrate the practical consequences of broken attestation and the range of systems potentially affected.

BuilderNet. BuilderNet is a trusted smart contract execution framework within the Ethereum ecosystem that relies on TDX attestation to guarantee fairness and confidentiality. By exploiting TEE.fail, an attacker can impersonate a TEE, enabling secret leakage, front-running, and economic manipulation. This case study illustrates that broken attestation can have direct financial and economic consequences, rather than remaining a purely theoretical or cryptographic concern.

DSTACK. DSTACK is an SDK for deploying containerized applications to TEEs and relies directly on attestation APIs for trust decisions. By forging attestation quotes, an attacker can relay GPU attestations from other machines, causing supposedly confidential AI workloads to execute without effective TEE protection. In this scenario, cloud users are cryptographically misled into trusting insecure execution environments.

AMD SEV-SNP. SEV-SNP is AMD's implementation of CVMs. While it incorporates mitigations against certain ciphertext-based attacks (Amd sev-snp: strengthening vm isolation with integrity protection *2020*), the paper demonstrates that physical memory bus interposition can bypass these defenses. This shows that the underlying problem is architectural rather than vendor-specific and affects multiple confidential computing implementations.

5.3 Direct Secret Extraction without Attestation

Beyond forging attestation, the paper also demonstrates that the same architectural weaknesses enable direct secret extraction. While forged attestation allows attackers to obtain secrets outside of any TEE, this case study shows that even secrets legitimately provisioned into a TEE can be extracted at runtime. Deterministic memory encryption without integrity protection enables ciphertext-based leakage of sensitive data, highlighting that the vulnerability is not limited to trust establishment but also undermines runtime confidentiality guarantees.

5.4 Implications for Cloud Trust

Taken together, the case studies highlight that forged trust poses a serious problem in cloud environments. Cloud customers rely on TEEs to protect sensitive data and on remote attestation to verify trustworthiness. Cloud providers advertise hardware-backed confidentiality guarantees, yet these guarantees implicitly depend on architectural trade-offs that are not always reflected in the stated threat model. The presented attacks demonstrate that trust does not automatically scale with performance-oriented design decisions and that security guarantees must be reassessed in light of realistic attacker capabilities.

6. Critical Discussion

Architectural Trade-Offs. It is important to emphasize that the vulnerability exploited by TEE.fail is not caused by an implementation bug or flawed cryptography, but instead results from an intentional architectural compromise. To achieve the scalability and performance required for server-grade CPUs and confidential virtual machines, strong integrity and replay protection mechanisms based on Merkle trees got sacrificed. This decision directly reintroduces ciphertext-based attacks that were previously out of scope in early TEE designs. Architectural choices that are driven by performance therefore have to be reflected in the security claims so users don't rely on claims that no longer hold.

Threat Model Mismatch. TEEs are only secure relative to their threat model, making the unchanged threat model a central issue in this context. A correct threat model serves as a contract between system designers and users, making transparent which guarantees can realistically be expected and thus implying how the user can securely interact with the system. Maintaining an outdated threat model allows vulnerabilities to exist unnoticed while systems are still considered secure by definition.

Limitations of the Attack. The paper introduces a very powerful attacker that doesn't just perform a remote attack but is physically invasive. The attack requires hardware access to the memory subsystem as well as control on kernel level (*Schlüter 2024*). Nevertheless, the paper establishes that the required setup is feasible with inexpensive equiment and realistic for well-resourced or insider attackers. Especially because cloud environments consolidate many tenants this increases the attackers incentives. The need for precise timing and memory placement control makes the attack scalable across all deployments. Overall, the attack is not universal, but definitely credible and repeatable within the assumed cloud threat landscape.

Implications for Future TEE Design. *TEE.fail* highlights that confidentiality alone is insufficient and that deterministic memory encryption requires strong integrity and freshness guarantees to remain secure. While the paper emphasizes that the demonstrated weaknesses are rooted in hardware design and cannot be addressed through software patches alone, it underscores the need for rethinking future TEE architectures. Possible directions could involve the reintroduction of integrity protection for sensitive regions only and, more importantly, clearly communicating and updating security guarantees for TEEs.

7. Conclusion

This summary reviewed *TEE.fail*, a study that exposes fundamental security weaknesses in modern Trustes Execution Environments. Exploiting these vulnerabilities it demonstrates how deterministic encryption with weak integrity enable an attacker to extract platform attestation keys and impersonate legitimate TEEs, fully breaking the trust guarantees of cloud environments. The problem arises from the transition of TEEs from enclave-scale design to cloud-scale TEEs resulting in significant changes in the platforms architecture and security design to keep up with the higher performance requirements of cloud systems. In return, the system sacrifices strong integrity mechanisms that couldn't perform at the new scale. Importantly, these attacks do not rely on software bugs or outdated systems, but instead exploit a mismatch between unchanged threat models and altered architectural assumptions. As a key lesson *TEE.fail* shows that hardware-backed security guarantees are only meaningful when their underlying threat models accurately reflect the capabilities of a real-world attacker and that these assumptions have to be revisited when changes to the system are implemented. As confidential computing continues to evolve, it becomes of even higher relevance to clearly scope and transparently communicate security guarantees. Without this alignment, TEEs risk to provide a false sense of security in precisely the environments they are meant to be protecting.

References

Amd sev-snp: strengthening vm isolation with integrity protection. 2020. Technical report. Advanced Micro Devices.

Costan, Victor, and Srinivas Devadas. 2016. Intel sgx explained. *IACR Cryptology ePrint Archive.*

Intel Corporation. 2022. *Intel sgx data center attestation primitives orientation guide.* Intel Corporation.

———. 2023a. *Intel 64 and ia-32 architectures software developer's manual.* Intel Corporation.

———. 2023b. *Intel trusted domain extensions (tdx) module base specification.* Intel Corporation.

OWASP Foundation. 2023. *Owasp threat modeling cheat sheet.* https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html.

Schlüter, et al. 2024. Tee.fail: breaking trusted execution environments via ddr5 memory bus interposition. In *Usenix security symposium.*