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

Jalen Chuang[1], Alex Seto[2], Nicolas Berrios[1]Stephan van Schaik[3], Christina Garman[2], Daniel Genkin[1]
[1]Georgia Tech, [2]Purdue University, [3]van Schaik, LLC
{jchuang, berrios, genkin}@gatech.edu, {aseto, clg}@purdue.edu, stephan@synkhronix.com

1. Introduction

Trusted execution environments (TEEs) aim to offer strong privacy and integrity guarantees even in the presence of root level attacks capable of arbitrarily modifying the system's software. Recently however, there has been a pivotal shift in TEE deployment, moving TEEs from enclaves running on PC-oriented hardware to confidential virtual machines executing on server-grade CPUs. Under the hood, this change has also resulted in significant modifications to the underlying memory encryption engine, removing integrity guarantees as well as protections against replay attacks.

However, when comparing the TEE threat model to its actual deployments, an unsettling gap arises. While hardware vendors claim that general purpose TEEs do not offer complete mitigation against attackers with physical access [1, 2, 3], Web3-related TEE deployments are often permissionless, meaning any node can join the network regardless of its physical location, by presenting a trusted attestation status. Indeed, real-world deployments often acknowledge TEEs' checkered security record [4, 5], often arguing that these attacks are only possible in carefully-controlled lab settings and requiring expensive hardware equipment [6]. Thus, given the clear gap in TEE threat modeling, in this work we discuss the following questions:

What are the true costs of mounting physical attacks on modern TEE hardware? In particular, can such attacks be mounted by low-budget hobbyists? And if so, how can one best defend against them?

2. Our Contributions

In this poster we demonstrate for the first time a memory bus interposition attack on server grade DDR5 memory. Moreover, our attack can be done in under $1000 by computer hobbyists using equipment readily available on the secondhand market. Next, we use our interposition setup to observe memory bus transactions on Intel Xeon Scalable 5th Generation and AMD Zen 5 systems, thereby re-enabling ciphertext-based attacks on these machines. More specifically, we break SGX and TDX security guarantees by extracting attestation keys from machines in a fully trusted `UpToDate` attestation status. With extracted attestation keys in hand we demonstrate the effectiveness of our attacks by breaking the security guarantees of realistic deployments, including running GPU workloads outside TEE protections while passing attestation for NVIDIA Confidential Computing. Finally, we attack the newest version of AMD SEV-SNP present in Zen 5 EPYC processors, which includes ciphertext hiding features for preventing prior software-based ciphertext attacks [7]. To the best of our knowledge, this is the first ciphertext attack on DDR5-based systems, including SGX, TDX and SEV-SNP. Our results impact nearly all server-based TEE implementations with commercially-available hardware at the time of writing.

Constructing a Cheap Memory Interposition Setup. While prior work cites costs of memory interposition setups to be around $170,000 [8], we begin by showing that such numbers are vastly overestimated. Here, we show how hobbyist attackers can construct DRAM interposers costing under $1000, using supplies from secondhand electronic marketplaces, with just tweezers and simple soldering irons. We then demonstrate the use of our interposer to observe DDR5 DRAM traffic, using outdated (and thus cheap) logic analyzers not originally intended for such analysis.

Controlling Enclave Execution. With our ability to observe DDR5 memory transactions, our next step is to obtain a sufficient level of control over SGX and TDX execution. First, while Intel requires at least 8 DIMMs (e.g., 16 channels for DDR5 memory) for TDX and SGX activation, our setup is only capable of observing a single channel. Thus, we first reverse engineer the mapping between physical address and DIMM locations on 5th generation Intel Scalable Xeon systems. Next, we modify the OS kernel to place virtual addresses of interest on the DIMM and channel connected to our logic analyzer, allowing us to observe memory transactions to these addresses. Finally, we overcome the system's caching and force DRAM traffic via flushing, while using a control channel attack to precisely control TEE execution, synchronizing it to our logic analyzer triggering logic.

Attacking TDX and SGX Attestation. Next we notice that both TDX and SGX attestation rely on a single source of trust, namely an Intel-signed Provisioning Certification Enclave (PCE). Combining this observation with Intel's use of deterministic AES-XTS encryption, we are able to use our memory interposition setup to break the confidentiality of the PCE enclave, extracting its Provisioning Certification Key (PCK). With a PCK from a machine in fully trusted `UpToDate` status in hand, we then sign our own attestation keys which do not originate from an Intel-signed Quoting Enclave (QE). This in turn allows us to sign arbitrary SGX or TDX reports without any TEE protections, thereby

completely breaking SGX and TDX security guarantees. To the best of our knowledge, this is the first end-to-end attestation key extraction attack on a TDX system in a fully trusted `UpToDate` status.

Breaking TEE Confidentiality and Integrity. Having utilized our attack to subvert the attestation process, we proceed to examine the security of real-world applications relying on SGX and TDX for confidentiality and integrity guarantees. We notice a clear gap in threat model. While server TEEs are designed to run in data centers (and thus rely on the operator for physical security) [1, 3], many applications use TEEs with the explicit goal of avoiding operator trust, shifting it to the underlying hardware.

We first investigate BUILDERNET, a part of the Ethereum blockchain ecosystem that uses TDX to provide integrity, confidentiality, and trustworthiness for block builders and users, processing millions of dollars in value each month. We break these guarantees, demonstrating how a malicious operator can both extract configuration secrets and gain the ability to frontrun (and profit) without being detected.

NVIDIA Confidential Computing. Using our subverted TDX attestation process, we investigate Phala Network's DSTACK, an SDK for deploying docker containers into TDX with minimal code changes. We are able to break TDX's guarantees for all applications relying on DSTACK. Furthermore, we show that we are able to use an NVIDIA Confidential Computing attestation from a different computer as if it were our own. We then break the security guarantees of two applications relying on DSTACK, a tool for hosting Jupyter notebooks and an LLM frontend, passing NVIDIA Confidential Computing attestation while running workloads without any TEE protections.

Targeted Data Extraction Beyond Attestation. Having explored the implications of PCK extraction, we now proceed to demonstrate that even securing the system's PCE and QE is insufficient for mitigating bus interposition attacks. Here, we examine SECRET, one of the first privacy-preserving smart contract systems and wide-spread deployment of SGX, featuring a $57M USD market cap. To ensure the confidentiality of contract execution, SECRET relies on a single master key which is shared among all validators and stored in an enclave. While it would be possible to utilize our already-extracted PCK to falsely attest an attacker, we instead elect to use our memory interposition setup to directly extract a node's SECRET-specific ECDH private key from the SECRET enclave. This allows us to obtain the network's master key directly, thereby completely breaching SECRET's confidentiality guarantees, without attacking the system's quoting enclave or needing the machine's provisioning key.

Attacking AMD-SEV with Ciphertext Hiding. Going beyond Intel's SGX and TDX, we also investigate AMD's SEV-SNP. Here, in an effort to mitigate prior attacks [9], SEV-SNP running on Zen 5 EPYC CPU support Ciphertext Hiding [7], which hides a confidential VM's encrypted state from malicious hypervisors. However, as this does not protect against attackers with physical bus access, we re-enable ciphertext attacks on these machines by demonstrating the extraction of signing keys from OpenSSL's constant-time ECDSA implementation running within a confidential VM.

3. Mitigations

Avoiding Deterministic Encryption. We note that one of the fundamental issues with server TEEs is the use of deterministic memory encryption. This is a step back from the prior SGX implementation on client hardware, aimed at increasing encrypted memory support from 512 MB to 1 TB. Thus, reconciling these goals and obtaining scalable memory encryption without sacrificing these guarantees warrants further research. For current CPUs Intel has indicated that it is impossible to issue a microcode to update the memory encryption engine. Thus, we expect bus interposition to remain a viable attack vector in the foreseeable future.

Location Verification and CPU Whitelisting. Following our disclosure TEE deployments seem to have migrated to location or cloud verification primitives, which allow users to ascertain that TEE hardware is physically located in secure cloud environments as opposed to adversarial hands. An even more restrictive approach would be to allow users to whitelist specific CPU instances which are known to be in physically secure locations, preventing other CPUs from obtaining secrets. Both of these approaches are currently being implemented by several Web3 TEE deployments [10].

References

[1] S. Johnson, R. Makaram, A. Santoni, and V. Scarlata, "Supporting Intel SGX on multi-package platforms," 2021. [Online]. Available: https://arxiv.org/pdf/2507.08190

[2] Intel, "Intel® Hardware Shield – Intel® Total Memory Encryption," 2022. [Online]. Available: https://www.intel.com/content/dam/www/central-libraries/us/en/documents/white-paper-intel-tme.pdf

[3] AMD, "AMD SEV-SNP: Strengthening VM isolation with integrity protection and more," 2020. [Online]. Available: https://docs.amd.com/v/u/en-US/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more

[4] S. S. Zhou, "Technical analysis of why phala will not be affected by the intel sgx chip vulnerabilities," 2022. [Online]. Available: https://phala.network/posts/technical-analysis-of-why-phala-will-not-be-affected-by-the-intel-sgx-chip-vulnerabilities-e045b0189dc2

[5] A. Shidham, "Trusted execution environments (tees): A primer," 2025. [Online]. Available: https://a16zcrypto.com/posts/article/trusted-execution-environments-tees-primer/

[6] S. Network, "Secret network graypaper," 2025. [Online]. Available: https://scrt.network/graypaper

[7] AMD, "SEV ciphertext side channel attacks," 2025. [Online]. Available: https://www.amd.com/en/resources/product-security/bulletin/amd-sb-3021.html

[8] D. Lee, D. Jung, I. T. Fang, C.-C. Tsai, and R. A. Popa, "An Off-Chip attack on hardware enclaves via the memory bus," in *USENIX Security*, 2020.

[9] AMD, "Technical guidance for mitigating effects of ciphertext visibility under AMD SEV," 2022. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/resources/bulletin/technical-guidance-for-mitigating-effects-of-ciphertext-visibility-under-amd-sev.pdf

[10] https://proofofcloud.org/, 2025.

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

Jalen Chuang, Alex Seto, Nicolas Berrios,
Stephan van Schaik, Daniel Genkin, Christina Garman

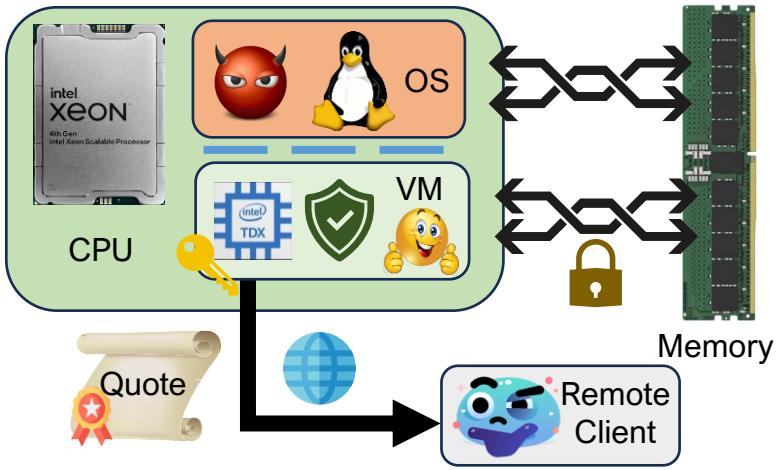
Georgia Tech

PURDUE UNIVERSITY

https://tee.fail

About TEEs

Server TEEs provides low-cost hardware-backed isolation even in the presence of a malicious host



Memory Bus Interposition

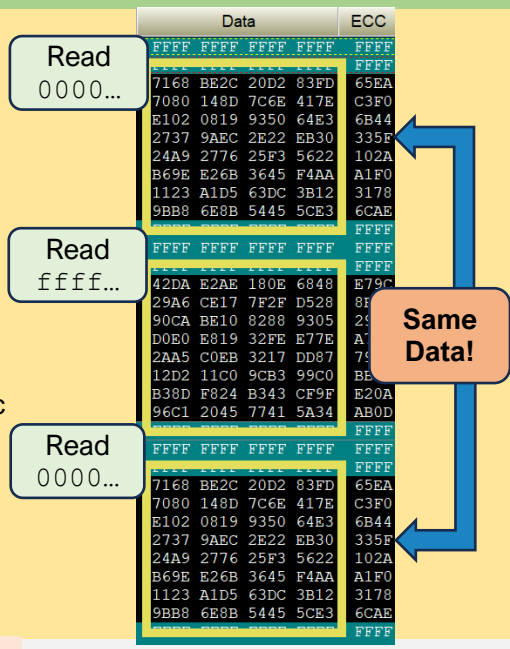
We designed and built an interposer in under $1000 to see memory reads/writes...

We found that TDX encryption is deterministic AES-XTS!

Original

Deterministic

Random



Attack

We use our interposition 🔍 capabilities on weak encryption to leak an Intel attestation key from a single quote signing operation

```
def mul_point_by_scalar(P: Point, k: Scalar):
  # Compute lookup table
  T: list[Scalar] = []
  for i in range(2**w):
    T[i] = P**i

  # Process k
  Q: Point = 1
  for i in range(0, k.bit_length(), w):
    Q *= (2**w)
    R = T[k[i:i+w]]
    Q *= R
  return Q
```

Public

Private

We construct a dictionary of ciphertexts making up T

Fill

T[]

We use our interposer to see R and match it to an entry of T, learning k[i:i+w]

Query

With our leaked attestation key, we can sign arbitrary forged attestation data on any CPU.

Applications using TEEs for trust lose security guarantees

Affected deployments include:

Secret network

dstack

BuilderNet

NVIDIA Confidential Compute

AMD SEV-SNP also affected

Intel and AMD consider these attacks out of scope. Users of TEEs must ensure they trust their cloud. Talk to us if you'd like to discuss in-depth mitigations!