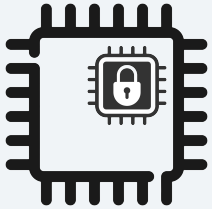
TRAIL OF BITS

One, Two, TEE: Trust in Numbers Meets Hardware Security

Paul Bottinelli @ Trail of Bits

Trusted Execution Environments (TEEs)

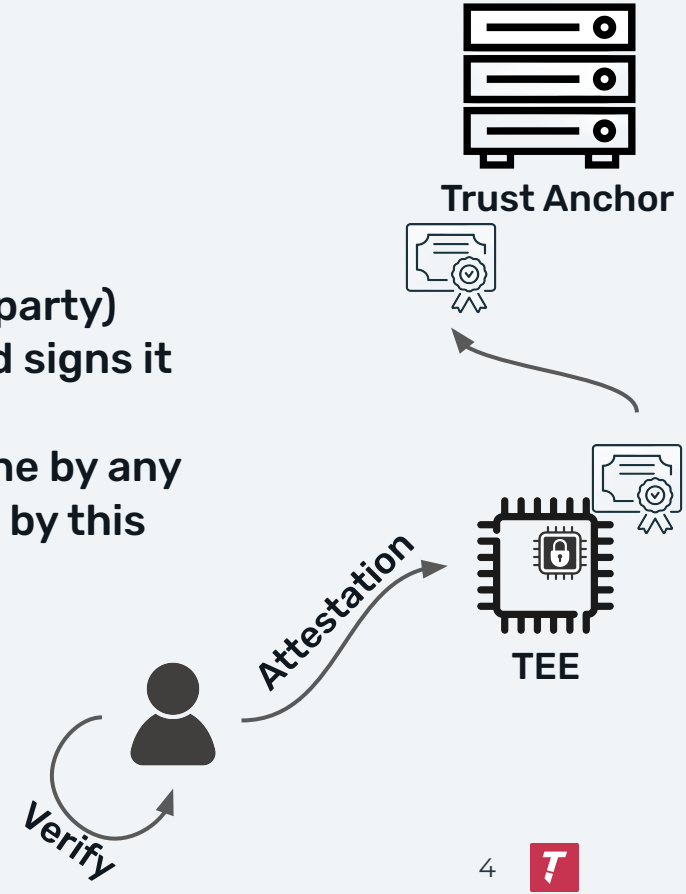


- **Secure areas within a processor that provide hardware-based protection for code and data, even from privileged software like the operating system or hypervisor***
- **Confidentiality: Data and code encrypted in memory and only accessible within the TEE**
- **Integrity: Data and code protected from tampering**
- **Attestation: Remote parties can cryptographically verify what code is running**

*In theory

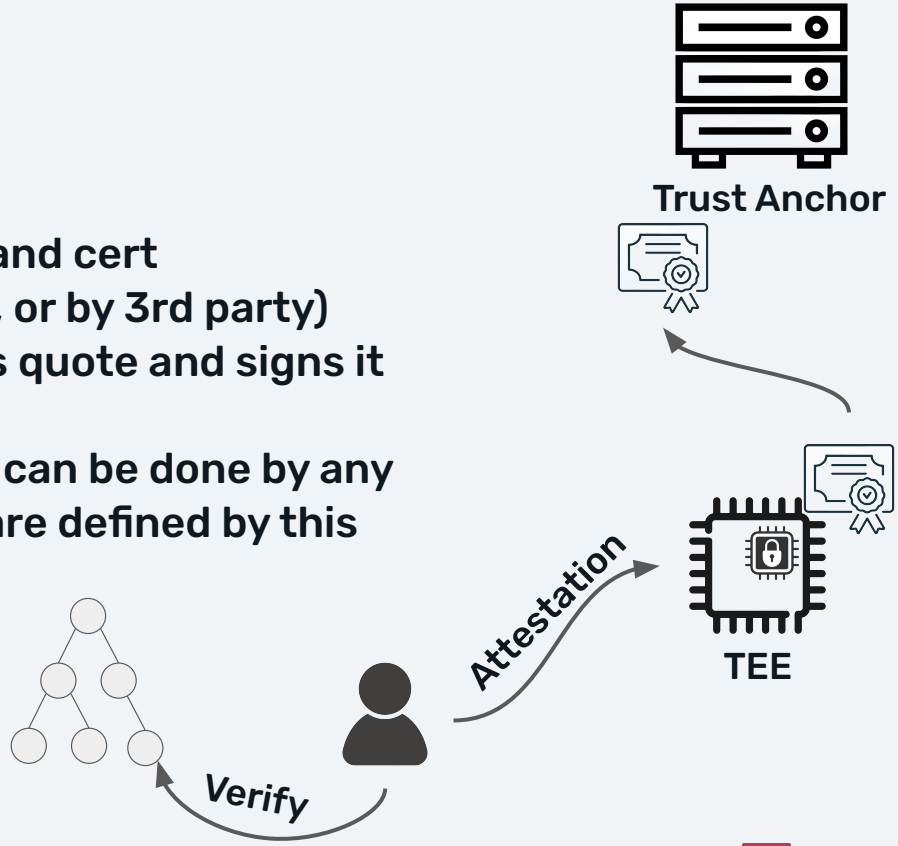
TEE: Attestations

1. **TEE provisioned with private key and cert**
2. **Attestation is requested (on boot, or by 3rd party)**
3. **TEE takes measurements, creates quote and signs it**
4. **External party *verifies* the quote**
   Intel TDX Guide: "This verification can be done by any party and the checks performed are defined by this party."
   - Verify quote signature
   - Verify certificate chain



TEE: Attestations

1. **TEE provisioned with private key and cert**
2. **Attestation is requested (on boot, or by 3rd party)**
3. **TEE takes measurements, creates quote and signs it**
4. **External party *verifies* the quote**
   Intel TDX Guide: "This verification can be done by any party and the checks performed are defined by this party."
   - Verify quote signature
   - Verify certificate chain
   - Measurement verification
   - ...

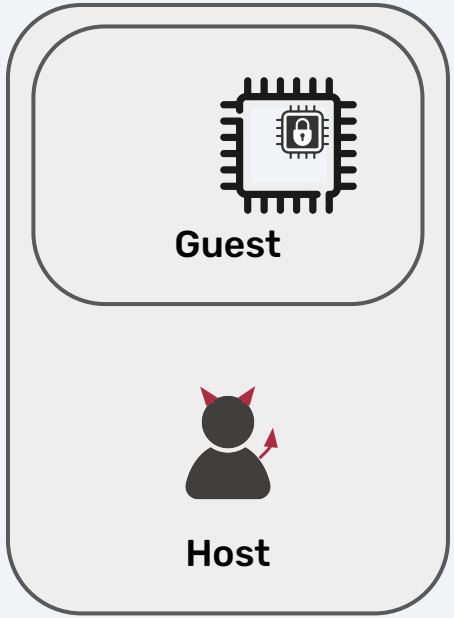


TEE: Shifting the Threat Model

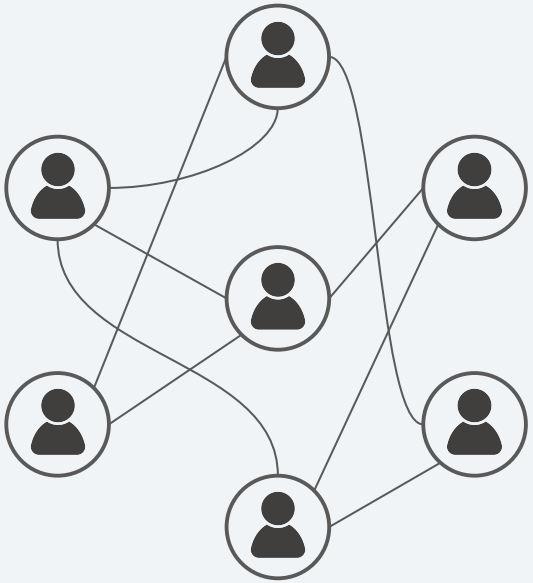
- **Traditionally, virtualization protects the host from the guest**
- **TEE threat model: the host is untrusted**
- **But the host controls**
  - I/O operations
  - Memory management
  - CPU scheduling
  - ...
- **Trust ultimately rooted in the manufacturer**

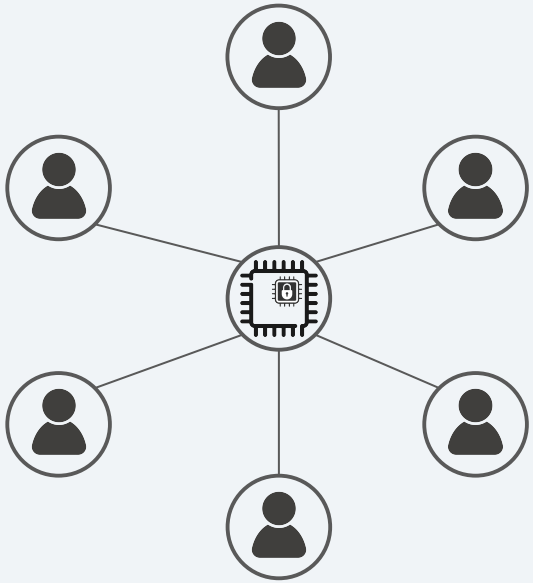
TEE: Shifting the Threat Model

- **Traditionally, virtualization protects the host from the guest**
- **TEE threat model: the host is untrusted**
- **But the host controls**
  - I/O operations
  - Memory management
  - CPU scheduling
  - ...
- **Trust ultimately rooted in the manufacturer**

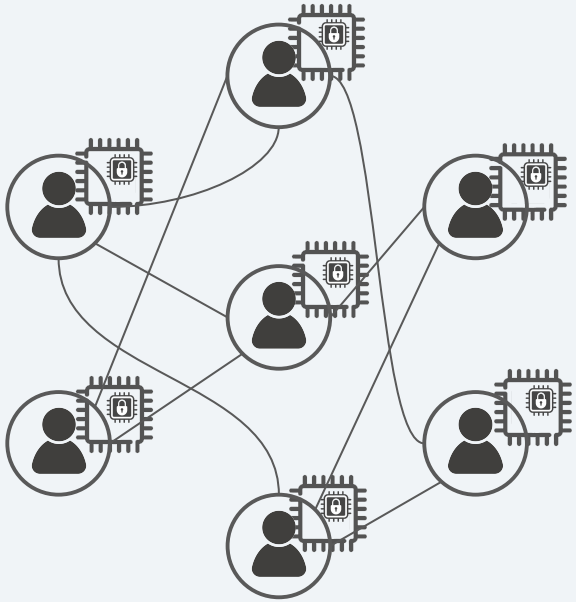


MPC vs TEE: A Clash of Trust





MPC inside TEEs



- **Combining them is desirable**
  - Beware of what claims you make!
- **If your TEE is not secure or not attesting to the correct things, you only get the illusion of additional security**
- **Many common MPC vulnerabilities are not mitigated by running inside a TEE**

Common MPC Vulnerabilities

Ambiguous encoding in hash functions (link)

Misbehaving participant

Missing parameters in Fiat-Shamir transform (link)

Missing input validation (link)

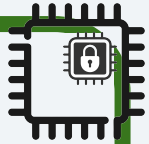
Side-channel attacks

Protocol issues (link)

Common MPC Vulnerabilities

Ambiguous encoding in hash functions (link)

Misbehaving participant



Missing parameters in Fiat-Shamir transform (link)

Missing input validation (link)

Side-channel attacks

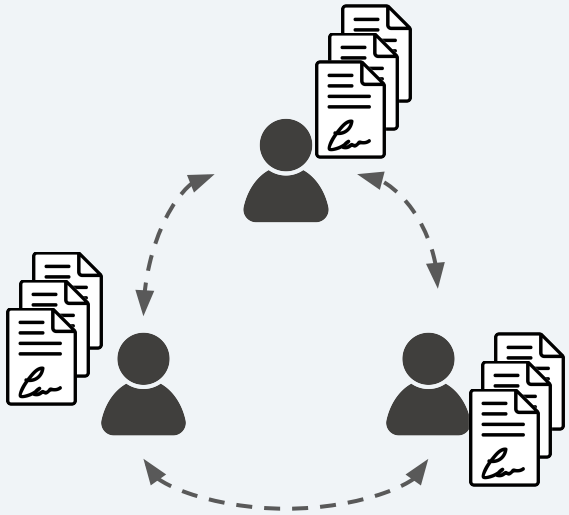
Protocol issues (link)

Spotlight: Threshold Signatures

- **Common threshold signature optimization: presignatures**
  - Independent of the message
  - Computationally expensive
- **Nonce reuse leads to key recovery!**

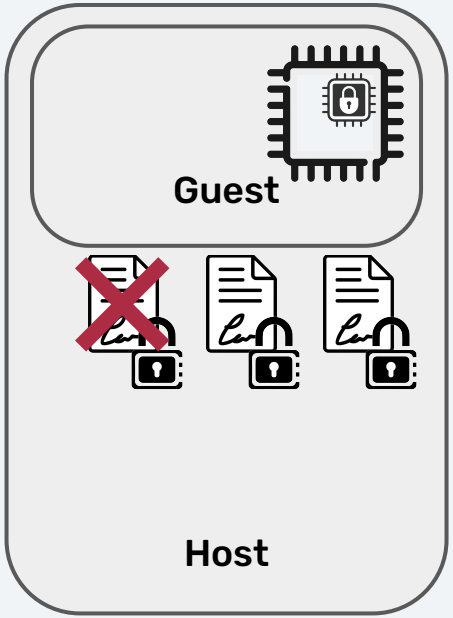
$$x = (s_1 - s_2) / (c_1 - c_2)$$

- **How about running nodes on a TEE**



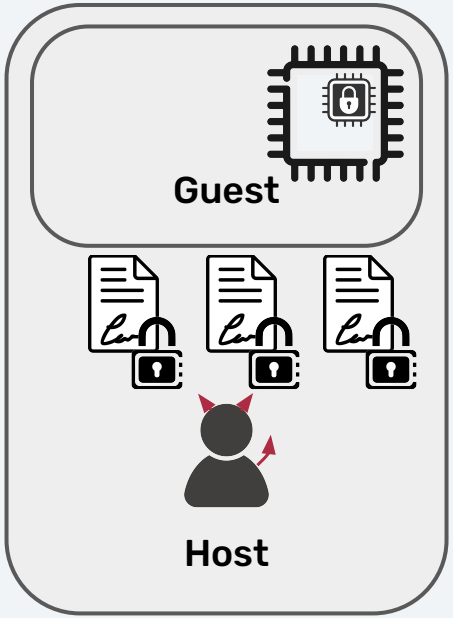
Spotlight: Threshold Signatures in a TEE

- **Some implementations store presignatures to disk**
- **Important to delete after use**



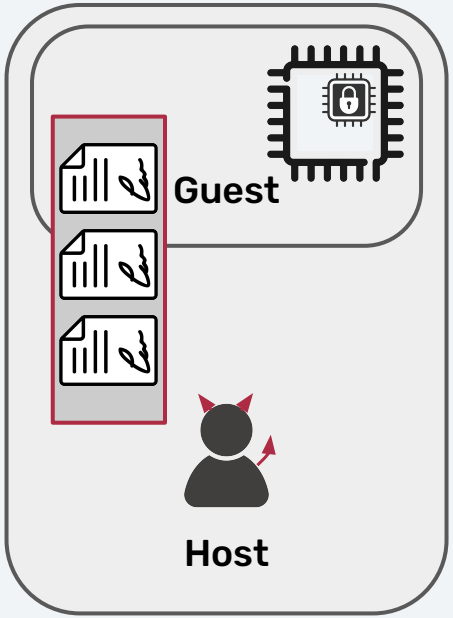
Spotlight: Threshold Signatures in a TEE

- **Some implementations store presignatures to disk**
- **Important to delete after use**
- **Malicious host can simply rollback the state**
  - ➢ **key recovery!**



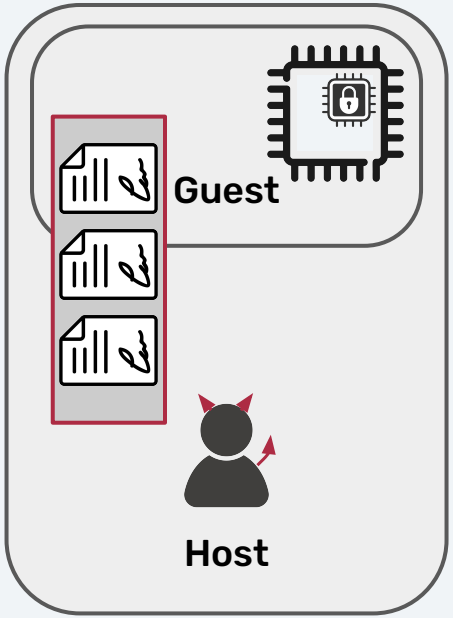
Spotlight: Threshold Signatures in a TEE

- Some implementations store presignatures to disk
- Important to delete after use
- Malicious host can simply rollback the state
- What if we only kept the presignatures in RAM?



Spotlight: Threshold Signatures in a TEE

- Some implementations store presignatures to disk
- Important to delete after use
- Malicious host can simply rollback the state
- What if we only kept the presignatures in RAM?
- Some TEEs do not provide rollback protection for the RAM itself
  - ➢ key recovery!



Other TEE Pitfalls

13. CVMs can be compromised via environment variable injection

Severity: **High** | Difficulty: **Medium**

Meta WhatsApp Private Processing Review (link)

- **Incomplete measurements (e.g. `LD_PRELOAD` attack)**
- **Missing verifications**
- **Side-channel attacks**
- **Trust model shift**
- **Not all TEEs are equally secure**
- **Insecure defaults**

HECKLER: Breaking Confidential VMs with Malicious Interrupts

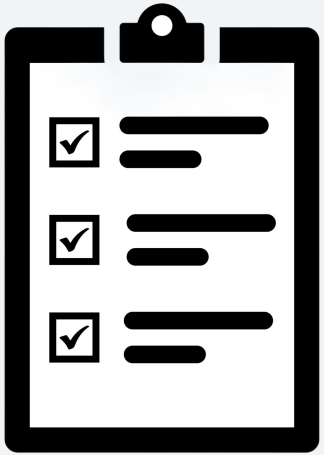
Benedict Schlüter Supraja Sridhara Mark Kuhne Andrin Bertschi Shweta Shinde
ETH Zurich

SEVered: Subverting AMD's Virtual Machine Encryption

Mathias Morbitzer, Manuel Huber, Julian Horsch and Sascha Wessel
Fraunhofer AISEC
Garching near Munich, Germany
{firstname.lastname}@aisec.fraunhofer.de

Deployment Best Practices

- **Require strong attestation and binding to MPC parties' identities**
- **Terminate P2P communications inside TEE**
- **Comprehensive verification**
- **Write constant-time code**
- **Follow best practices for/from specific vendors (e.g., Best practices for AWS Nitro Enclaves)**



Takeaways

- **Combining MPC and TEEs isn’t automatic security and can introduce subtle attacks**
- **TEEs solve lots of problems**
- **Follow best practices for TEE deployment and attestation verification**
- **Carefully review code and protocols**

blog.trailofbits.com/