This is a work-in-progress draft by Shunfan Zhou and Hang Yin. Welcome to comment and experiment with the idea.

Root-of-Trust (RoT) is the cornerstone of the TEE chain-of-trust. The end user verifies Remote Attestations signed by the CPU, which ultimately rely on keys derived from a set of hardware-held secrets. The hardware component that manages these root secrets, verifies firmware and applications, and issues Remote Attestations is called the Root-of-Trust.

[

image

2028×1248 147 KB

](https://collective.flashbots.net/uploads/default/original/2X/3/3c548a1b081796f6ba6b99535d3ab7e7654c5727.jpeg)

Hardware Root-of-Trust

However, hardware RoT has several drawbacks due to the limitations of physical hardware:

- Unrecoverable TCB

: Usually, bugs in TEE can be fixed by microcode upgrades with a TCB Recovery process. However, when the secret keys in the RoT are extracted, an attacker can simulate a RoT outside of the hardware and issue arbitrary Remote Attestations using the secrets. This invalidates the TCB Recovery process.

- Vendor/Physical Chip Locked

: The app running in the TEE relies on keys derived from the RoT to seal data and maintain identities (e.g., for signing signatures and establishing TLS connections). When the RoT is lost, the corresponding data and identities are lost as well. This prevents workloads from migrating between TEEs and blocks users from building services with high availability.

Software Root-of-Trust

Given these significant drawbacks of hardware RoT, it’s imperative to explore alternative approaches that offer greater flexibility and resilience. One such approach is abstracting the RoT into software.

To understand how a software RoT can address these challenges, let’s reconsider its role in the context of a TEE and explore how it can enhance security:

1. Measurement

: Verify the authenticity of the child layer in the chain-of-trust. * In Intel SGX, RoT verifies the version of the CPU microcode, BIOS configuration, and the code loaded into the SGX enclave.

1. In Intel SGX, RoT verifies the version of the CPU microcode, BIOS configuration, and the code loaded into the SGX enclave.

2. Key Derivation Service

: Provide a service to derive keys for the child layer. The derived keys can be used to implement data sealing, message signing, and Remote Attestation when combined with the measurements.

These two properties can also be implemented in software, offering better flexibility and resilience. A simple software RoT can be implemented as follows:

- A KMS that maintains a set of secrets.

- Implement Measurement

: Verify the authenticity of a physical TEE by requesting and verifying the native Remote Attestation from the TEE (e.g., Intel SGX DCAP Remote Attestation). Once this measurement is taken, the RoT can trust that the desired software is running on the desired hardware environment with all the proper properties, usually integrity and confidentiality.

- Implement Key Derivation Service

: Once verification passes, the KMS can provide services allowing TEE applications to obtain derived keys from the RoT. The communication channel between the RoT and the TEE application can be secured using end-to-end encryption protocols like RA-TLS. Similar to a hardware RoT, the service provides capabilities to seal data, sign certificates, and implement Remote Attestation for TEE applications.

Implementing a software RoT offers several advantages but also introduces new challenges that must be carefully considered:

- Pros
- Always Recoverable TCB

: TEEs with a compromised hardware RoT can be rejected by the RoT system when measuring their Remote Attestations. In case of a hardware RoT breach, the software RoT remains secure. The software RoT can blacklist the entire product line of the affected hardware to recover the TCB of the system.

- Dynamic Measurement

: Instead of relying on static secrets stored in the hardware RoT, a software RoT can measure the physical TEE periodically and detect vulnerabilities as they occur. When a vulnerability is detected, the RoT can trigger a TEE application migration

, remove the affected TEE from the network, and perform a key rotation to reduce the risk (with forward and backward secrecy).

- Application Migration

: The keys the application relies on (for sealing, signing, etc.) are no longer associated with the hardware RoT. Regardless of which hardware TEE the application runs on, it can always obtain the same set of derived keys from the software RoT, decoupling TEE applications from the hardware implementation.

- Always Recoverable TCB

: TEEs with a compromised hardware RoT can be rejected by the RoT system when measuring their Remote Attestations. In case of a hardware RoT breach, the software RoT remains secure. The software RoT can blacklist the entire product line of the affected hardware to recover the TCB of the system.

- Dynamic Measurement

: Instead of relying on static secrets stored in the hardware RoT, a software RoT can measure the physical TEE periodically and detect vulnerabilities as they occur. When a vulnerability is detected, the RoT can trigger a TEE application migration

, remove the affected TEE from the network, and perform a key rotation to reduce the risk (with forward and backward secrecy).

- Application Migration

: The keys the application relies on (for sealing, signing, etc.) are no longer associated with the hardware RoT. Regardless of which hardware TEE the application runs on, it can always obtain the same set of derived keys from the software RoT, decoupling TEE applications from the hardware implementation.

- Cons
- Centralization Risk

: It’s challenging to run a software RoT securely. The RoT must (1) keep the secrets confidential, (2) ensure the integrity of its code, and (3) maintain the liveness of the service. These three properties are especially challenging without relying on trust in a centralized party.

- Centralization Risk

: It’s challenging to run a software RoT securely. The RoT must (1) keep the secrets confidential, (2) ensure the integrity of its code, and (3) maintain the liveness of the service. These three properties are especially challenging without relying on trust in a centralized party.

Decentralized Root-of-Trust

While a software-based RoT offers significant benefits, it introduces the risk of centralization, which compromises security. To address this, a decentralized RoT can eliminate such risks and offer a more secure, distributed alternative.

We propose a decentralized RoT model. This approach distributes trust across multiple independent nodes, eliminating reliance on a single entity. A decentralized RoT serves as the foundational trust anchor for the entire TEE network, removing dependencies on specific TEE manufacturers or cloud providers.

A decentralized RoT can be implemented as follows:

- A software RoT using Secure Multi-party Computation (MPC) to manage and use the root secrets. MPC distributes

trust across multiple independent nodes, ensuring that no single entity can compromise the RoT. Attackers must control a majority (typically two-thirds) of nodes to access the secrets.

- The RoT nodes run in TEEs to (1) measure the integrity of the code and (2) provide defense-in-depth to reduce the risk of MPC collusion attacks. With reproducible builds, anyone can verify that the RoT nodes have implemented the protocol properly.
- Smart contracts on the blockchain act as the governance layer for the decentralized RoT, enforcing rules around RoT node behavior, including blacklisting certain hardware from joining the TEE network. By verifying Remote Attestations on-chain, smart contracts ensure that only nodes running approved code are part of the system.
- Additionally, staking mechanisms can be introduced to add cryptoeconomic security, where node operators must stake tokens as collateral. Any malicious behavior risks losing this stake, incentivizing honest participation.

[

image

1920×1619 92.2 KB

](https://collective.flashbots.net/uploads/default/original/2X/c/c26de068db9280b600b48dddf456636408bd6b70.jpeg)

The decentralized architecture solves the centralization risk in software RoT because:

1. Confidentiality

: The secrets are protected by threshold MPC, eliminating the single point of failure, with TEE providing defense-in-depth.

1. Integrity

: The code is protected by both TEE Remote Attestation and threshold MPC for redundancy.

1. Liveness

: The MPC setup can tolerate the failure of up to one-third of the nodes. Cryptoeconomics can be introduced to further incentivize operators to behave well.

This decentralized model effectively overcomes the vulnerabilities of centralized RoT by distributing control across a secure, blockchain-governed network of MPC nodes. Additionally, the architecture reduces the complexity for users in verifying the security properties of TEE applications by providing a chain-of-trust. Users can easily verify the chain-of-trust by checking the signatures originating from the smart contract.

Implementation

1. Phala’s SGX Network
2. DAO-governed KMS using TEE and on-chain governance (on a Substrate blockchain).
3. KMS is secured by TEE but not MPC yet.
4. Key rotation is implemented but triggered manually in catastrophes.
5. Apps are decoupled from HW RoT.
6. As of Sep 24, 2024, the network has 11 KMS (called Gatekeepers) and around 39,570 TEE workers.
7. DAO-governed KMS using TEE and on-chain governance (on a Substrate blockchain).
8. KMS is secured by TEE but not MPC yet.
9. Key rotation is implemented but triggered manually in catastrophes.
10. Apps are decoupled from HW RoT.
11. As of Sep 24, 2024, the network has 11 KMS (called Gatekeepers) and around 39,570 TEE workers.
12. (Maybe others who implemented Ekiden?)

Discussion

- Possibility of introducing “Open Source TEE” or very secure RoT?
- Sylvain’s Project qTEE: qtee - HackMD
- Poetic Tech’s Autonomous TEE Manifesto: Autonomous TEE Manifesto · Poetic Technologies
- Sylvain’s Project qTEE: qtee - HackMD
- Poetic Tech’s Autonomous TEE Manifesto: Autonomous TEE Manifesto · Poetic Technologies
- Forward and backward secrecy with key rotation
- Automatic HTTPS certbot for TEE apps via an extension of the chain-of-trust
- Storage migration, persistence, and state continuity

Conclusion

The decentralized RoT model not only addresses the limitations of both hardware and centralized software RoT systems but also ensures a more secure, scalable solution for TEE systems.

Essentially, the architecture implements a decentralized RoT where the user ultimately trusts the security of the blockchain. The trust is then extended through protocol governance, the KMS implementation, the hardware TEE, and finally to the applications. By abstracting RoT and distributing trust across the network, the user no longer relies on any specific hardware implementation, and the hardware TEE becomes hot-swappable.