Confidentiality-preserving Smart Contracts

Ahmed Lekssays
University of Insubria, Varese, Italy

September 23rd, 2021

Outline

Introduction

Introduction

Motivation

- Smart contracts inherit some **undesirable** blockchain properties;
- Existing smart contract systems thus lack **confidentiality** or **privacy**;
- Blockchain consensus requirements also hamper smart contracts with **poor performance**.

Problem Statement

Design a platform for confidential and performant smart contracts' execution.

Background

Smart Contracts and Blockchains

- Smart Contracts are programs executed by a network of participants who reach agreement on the programs' state;
- **Full replication** on all nodes provides a high level of **fault tolerance** and **availability**;
- On-chain computation of fully replicated smart contracts is inherently **expensive**;
- Contract state and user input must be **public** in order for miners to verify correct computation.

⇒ **Lack of privacy**.

Trusted Hardware with Attestation

- A trusted execution environment (TEE) protects the **confidentiality** and **integrity** of computations;
- A TEE can issue proofs, known as **attestations**, of computation **correctness**;
- **Intel SGX** provides a CPU-based implementation of TEEs—known as **enclaves** in SGX—for general-purpose computation;
- It is **infeasible** for any entity other than an SGX platform to generate any attestation;
- SGX alone **cannot guarantee availability**: a malicious host can terminate enclaves or drop messages arbitrarily.

⇒ **Lack of availability.**

Challenges

Tolerating TEE Failures

- **Availability failures**
  - A malicious host can **terminate enclaves,** and even an honest host could **lose enclaves in a power cycle**.
- **Side channels**
  - Recent work has **uncovered data leakage** via side channel attacks;
  - Existing defenses are generally **application** and **attack-specific**;
  - It is still desirable to **limit the impact** of compromised TEEs.
- **Timer failures**
  - TEEs in general **lack trusted time sources**;
  - Although a trusted relative timer is available, the communication between enclaves and the timer can be **delayed by the OS**.

Proof of Publication for PoW blockchains

- A TEE must be able to **efficiently** verify that an item has been stored
- in the blockchain.
- Such a proof can consist of **signatures** from a quorum of consensus nodes (Permissioned Blockchain).
- TEEs must be able to **validate new blocks** (Permissionless Blockchain).
  - A trusted timer is needed to defend against an adversary isolating an enclave and presenting an invalid subchain.
- An attacker delaying a timer's responses **cannot** prevent an enclave from successfully verifying blockchain contents **given trust in, e.g. TLS-enabled NTP servers**.

Atomic Delivery of Execution Results

- Atomicity of executions namely either both executions *exc1*, *exc2* finish or none of them;
- TEE **cannot** tell whether an input state is **fresh**, an attacker can provide **stale** states to resume a TEE's execution from an **old state**;
  - An attacker may repeatedly rewind until receiving the **desired output**;
  - Another example is that rewinding could defeat budget based privacy protection, such as **differential privacy**.

Ekiden

Cheng, Raymond, et al. "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts." 2019 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2019.

Overview

- **Clients** are end users of smart contracts;
  - A client can create contracts or execute existing ones with secret input.
- **Compute nodes** process requests from clients by running the contract in a contract TEE and generating **attestations** proving the correctness of state updates.

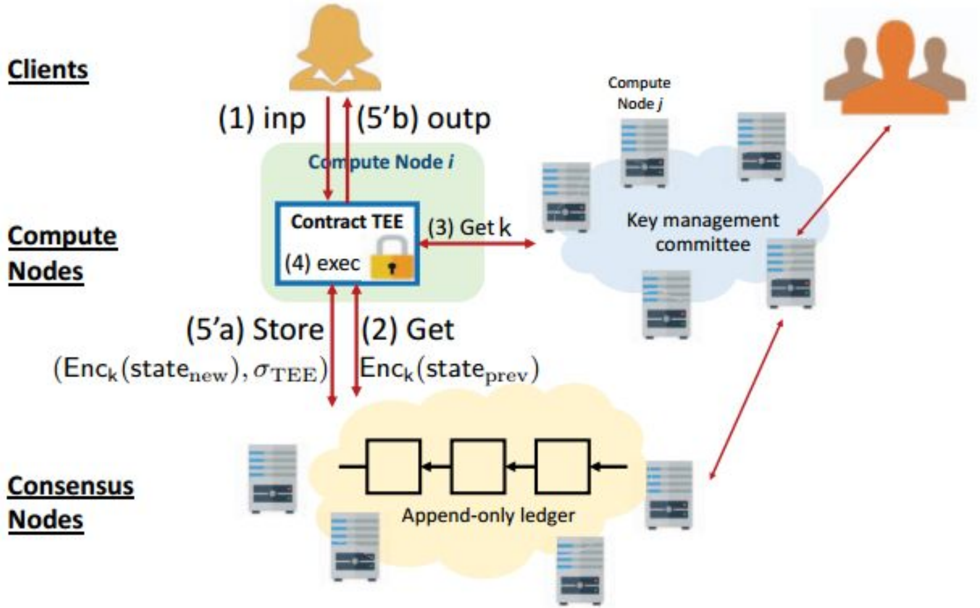

Figure 1: Ekiden Overview

Overview (Cont'd)

- **Consensus nodes** maintain a distributed **append-only** ledger by running a consensus protocol;
- Contract state and attestations are **persisted** on the blockchain;
- Consensus nodes are responsible for checking the **validity of state updates using TEE attestations.**

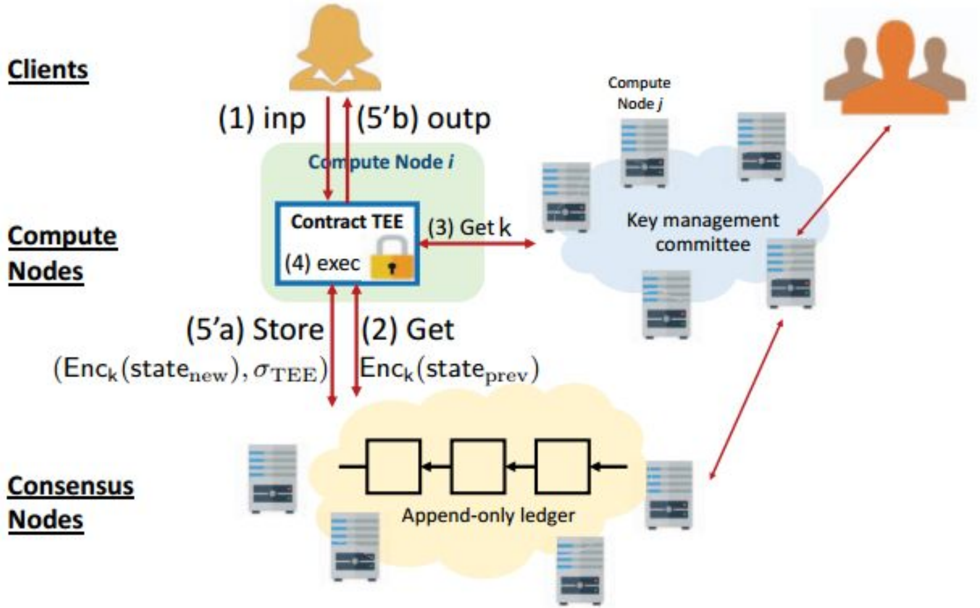

Figure 1: Ekiden Overview

Overview (Cont'd)

- Ekiden **decouples** request execution from consensus;
- A request is only executed by **K compute nodes** (possibly K=1);
- **Proof of correct execution** takes the form of a signature.
- Consensus nodes do not need neither **trusted hardware** nor **to contact the IAS** to verify it.

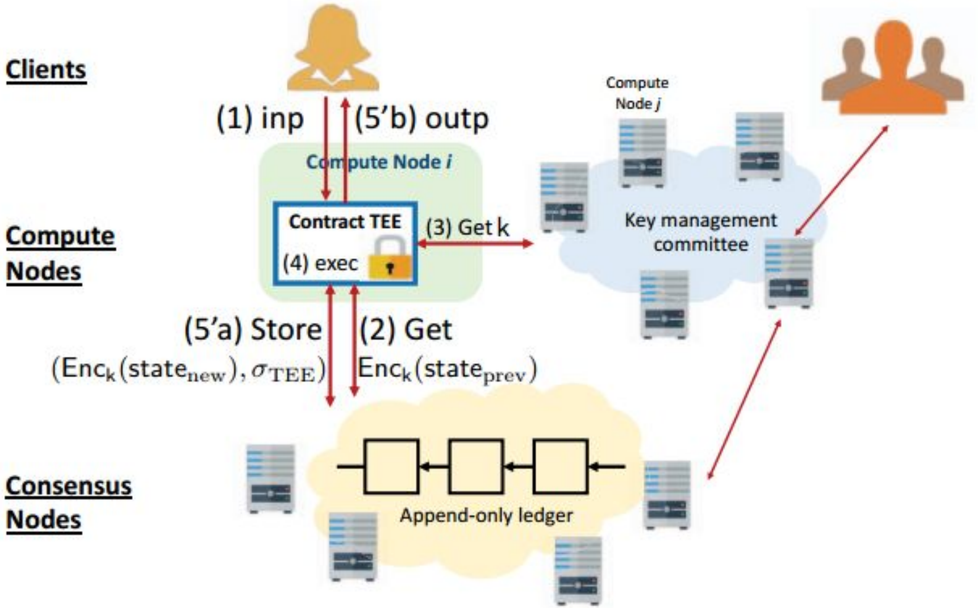

Figure 1: Ekiden Overview

Security Goals

- **Correct execution**:
    - Contract state transitions reflect correct execution of contract code on given state and inputs.
- **Consistency**:
    - At any time, the blockchain stores a single sequence of **state transitions consistent with the view of each compute node**.

Security Goals (Cont'd)

- **Secrecy**:
  - Ekiden guarantees that contract state and inputs from honest clients are **kept secret from all other parties** (without any TEE breach);
  - Ekiden is resilient to some **key-manager TEEs being breached**.
- **Graceful confidentiality degradation**:
  - Should a confidentiality breach occur in a computation node, Ekiden provides **forward secrecy** and **reasonable isolation** from the affected TEEs.

⇒ Ekiden **does not prevent** contract-level leakage (e.g. through covert channels, bugs or side channels).

Evaluation

- **Use cases**:
  - **Machine Learning Contracts** (predicting the likelihood of heart disease based on medical records)
  - **Smart Building Thermal Modeling** (an implementation of non-linear least squares, which is used to predict temperatures based on time series thermal data from smart buildings).
  - **Tokens** (an implementation in Rust of an ERC20 Token);
  - **Poker** (a contract where users take turns submitting their actions to the contract, and the smart contract contains all of the game logic for shuffling and (selectively) revealing cards);
  - **CryptoKitties** (an Ethereum game that allows users to breed virtual cats, which are stored on chain as ERC721 tokens).

Evaluation

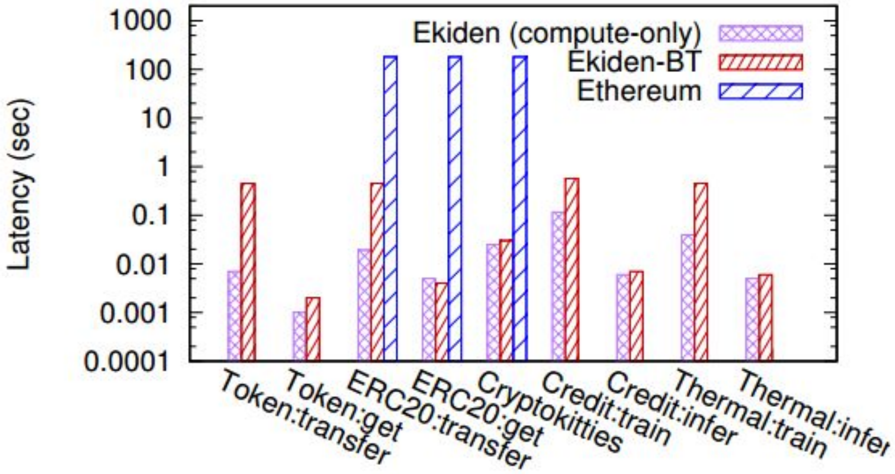

Figure 2: End-to-end latency of client requests for various contracts.

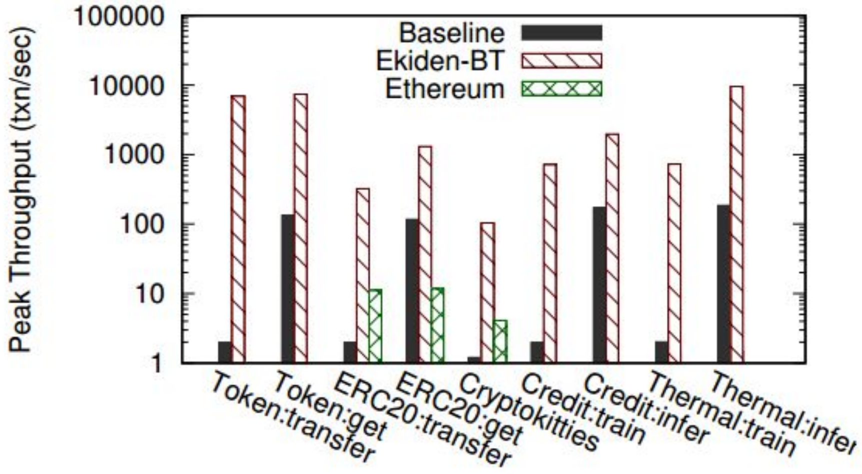

Figure 3: Throughput comparison across contracts and systems.

Other Approaches

ZKP-based Approaches: Hawk

- Hawk has strong privacy goals that include :
  - Hiding the amounts and transacting parties of monetary transfers;
  - Hiding contract state from non-participants;
  - Supporting private inputs that are hidden even from other participants in the contract.
- It suffers from some limitations:
  - SNARKs require a per-circuit trusted setup, which means that for every distinct program that a contract implements, a new trusted setup is required;
  - Each contract requires kilobytes of data to be put on-chain;
  - Privacy in Hawk relies on trusting a third-party manager who gets to see all the private data.

Kosba, Ahmed, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. 2016. "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts." In 2016 IEEE Symposium on Security and Privacy (SP), 839–58. ieeexplore.ieee.org.

Secure MPC-based Approaches: Enigma

- Secure multi-party computation is a cryptographic technique that allows parties to compute functions on private inputs without learning anything but their output.
- This enables attaching monetary conditions to the outcome of computations and incentivizing fairness (by penalizing aborting parties).
- MPC based systems require the active (and interactive) participation of all computing nodes.
- The cryptographic tools impose a significant efficiency burden.

Zyskind, G., Nathan, O., & Pentland, A. (2015). Enigma: Decentralized computation platform with guaranteed privacy. arXiv preprint arXiv:1506.03471.

Off-chain Approaches: Arbitrum

- Smart contracts are considered as VMs.
- Execution verification is only launched in case of a dispute (challenge-based verification).
- The challenger and the entity that run the VM deposit a stake.
- The verifiers need to check only specific instructions.
- Whoever fails loses deposit, half for the winner and the other half for the verifier.
- It is consensus agnostic.

Kalodner, Harry, et al. "Arbitrum: Scalable, private smart contracts." 27th {USENIX} Security Symposium ({USENIX} Security 18). 2018.

Existing Technologies

HF Private Chaincode

- Hyperledger Fabric Private Chaincode (FPC) enables the execution of chaincodes using **Intel SGX** for Hyperledger Fabric.
- It allows to write chaincode applications where the data is **encrypted** on the ledger and can only be accessed in clear by authorized parties [2].

HYPERLEDGER FABRIC

Hyperledger PDOs

- **Private Data Objects (PDO)** enables sharing of data and coordinating action amongst mutually distrusting parties;
- Interaction is mediated through a “smart contract” that defines data access and update policies;
- The smart contracts policies are enforced through execution in a **Trusted Execution Environment (TEE)**;
- PDOs use **Hyperledger Sawtooth** distributed ledger [3].

HYPERLEDGER SAWTOOTH

Microsoft CCF

- **Confidential Consortium Framework (CCF)** is an open-source framework for building a new category of secure, highly available, and performant applications that focus on multi-party compute and data;
- **CCF** leverages trust in a consortium of governing members and in a network of replicated **hardware-protected execution environments** to achieve high throughput, low latency, strong integrity and strong confidentiality [4].

Conclusion

Conclusion

- Smart contracts lack **privacy** and TEEs lack **availability**;
- TEE and Blockchain are **complementary**;
- Usage of TEE in blockchain **improves performance** and **preserves privacy**;
- There are current PoC developed by various companies like **Hyperledger** (PDOs, HF Private Chaincodes), **Microsoft** (CCF), and **Oasis Labs** (Ekiden).
- Other approaches that involve different techniques like ZKP, MCP, and off-chain evaluation were proposed.

Thank you!

Any question?

References

[1] Cheng, Raymond, et al. "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts." 2019 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2019.

[2] Kosba, Ahmed, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. 2016. "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts." In 2016 IEEE Symposium on Security and Privacy (SP), 839–58. ieeexplore.ieee.org.

[3] Zyskind, G., Nathan, O., & Pentland, A. (2015). Enigma: Decentralized computation platform with guaranteed privacy. arXiv preprint arXiv:1506.03471.

[4] Kalodner, Harry, et al. "Arbitrum: Scalable, private smart contracts." 27th {USENIX} Security Symposium ({USENIX} Security 18). 2018.

[5] Hyperledger-Labs. "Hyperledger-Labs/Fabric-Private-Chaincode." GitHub, github.com/hyperledger-labs/fabric-private-chaincode.

[6] Hyperledger-Labs. "Hyperledger-Labs/private-data-objects." GitHub, github.com/hyperledger-labs/private-data-objects.

[7] Microsoft. "Microsoft/CCF" GitHub, github.com/microsoft/CCF.