Treaty

Secure Distributed Transactions

Dimitra Giantsidi, Maurice Bailleu,
Natacha Crooks, Pramod Bhatotia

IEEE/IFIP DSN 2022

Distributed transactions

- A powerful programming abstraction
  - atomic processing of massive datasets
  - serializability
  - fault tolerance

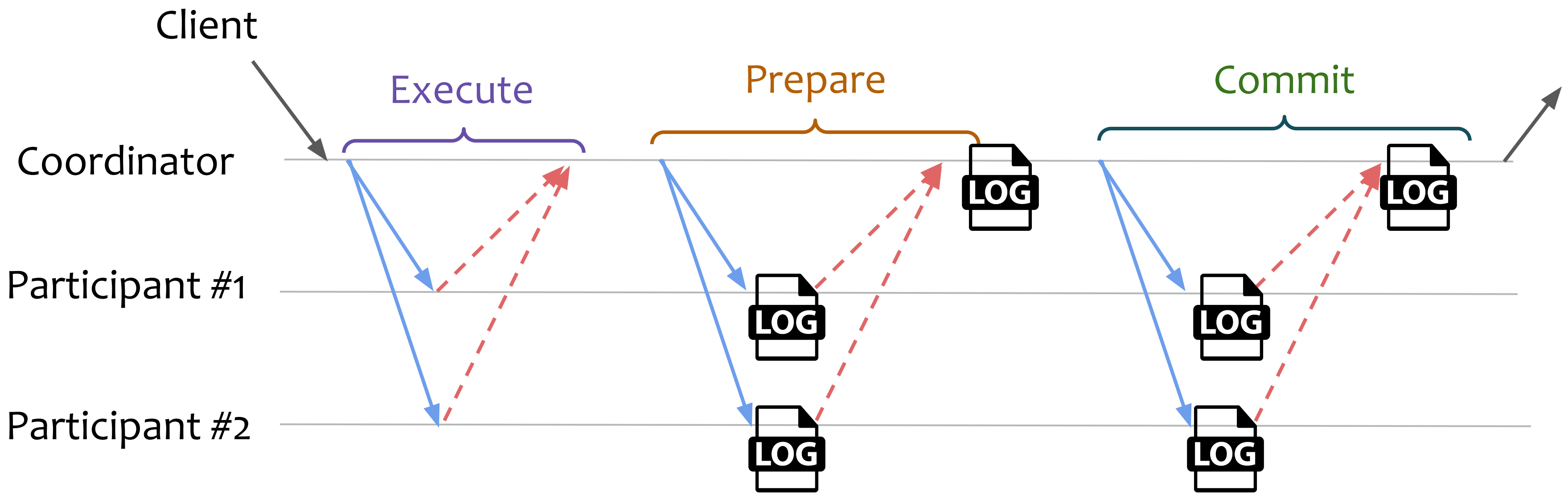
- Properties (ACID)
  - **A**tomicity, **C**onsistency, **I**solation, **D**urability

Google Cloud Spanner

DynamoDB

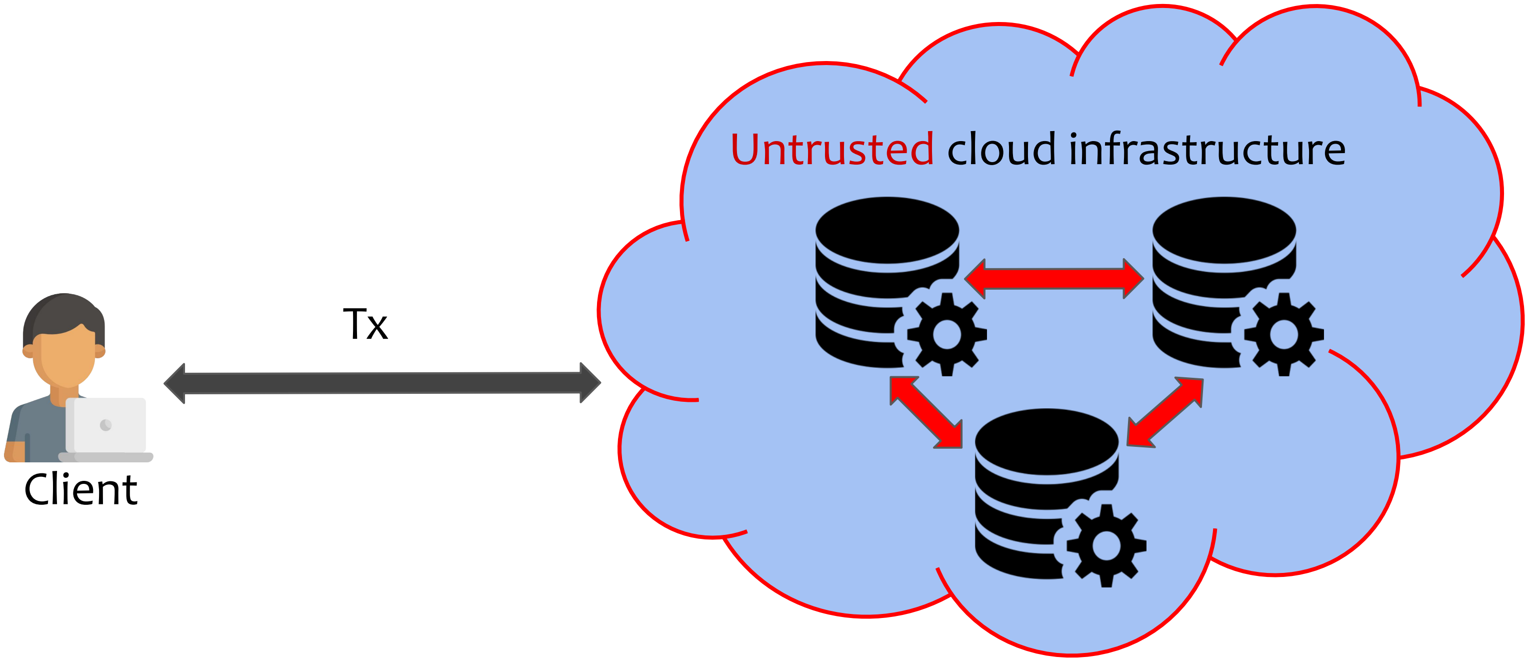
ZippyDB

Two-phase-commit (2PC) protocol



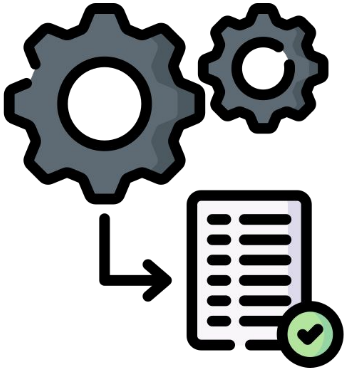
Txs require to exchange messages and log persistently their state

Transactions in the cloud



Attackers can compromise the security properties

Threats for distributed Txs in the cloud

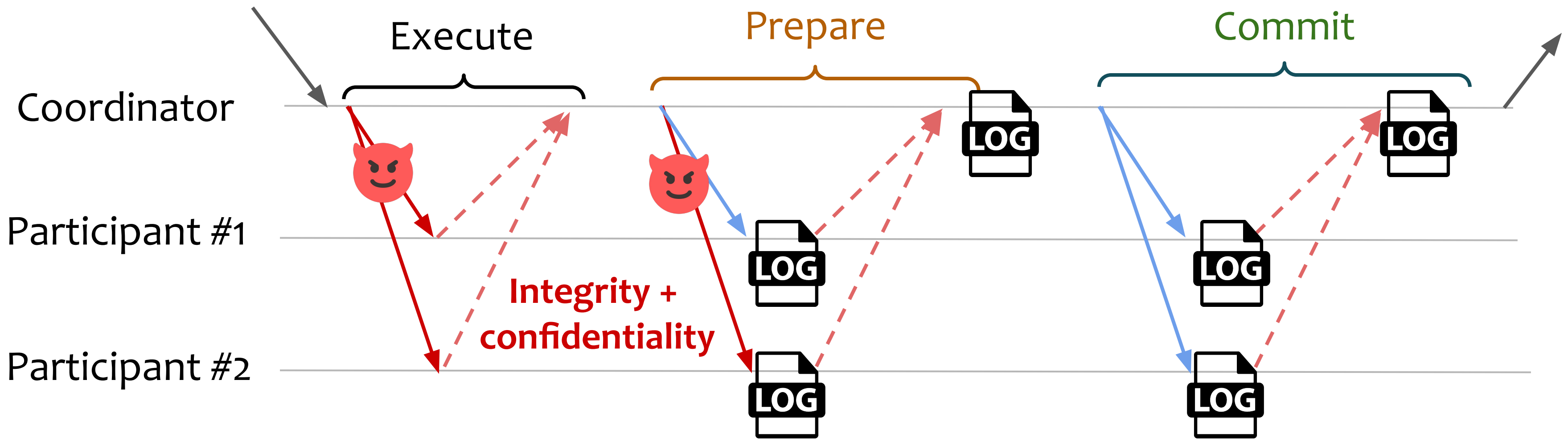


+

#1: Secure execution

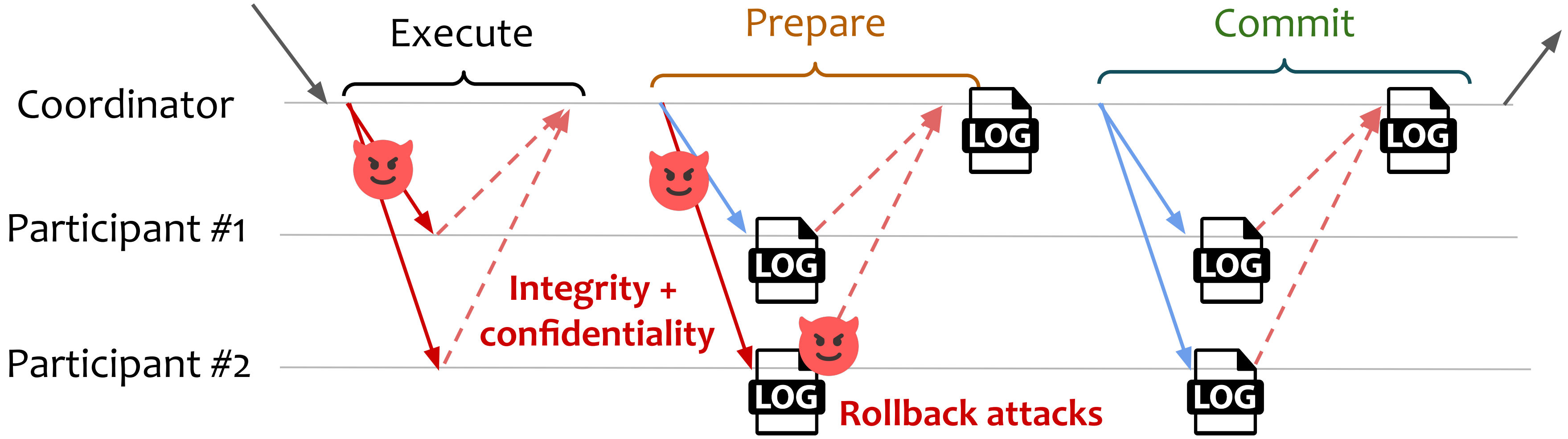
#2: Secure persistency

Threat #1: Secure execution



How to guarantee secure execution for Txs?

Threat #2: Secure persistency



How to ensure secure persistency (crash-consistency + rollback protection) for Txs?

Problem statement

To design **a distributed KV store** with **secure Tx execution and secure persistency**

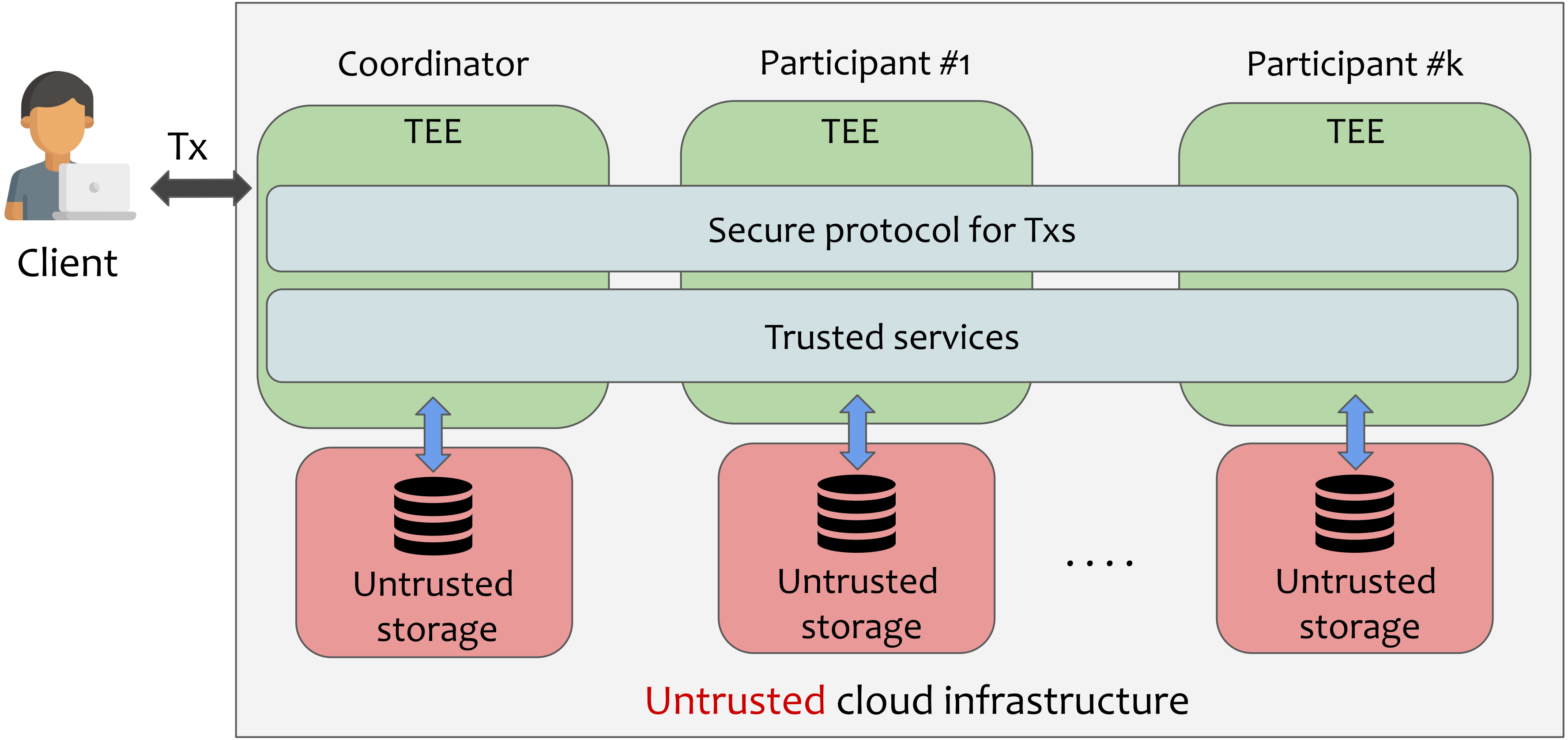
Our proposal

Treaty
A secure distributed transactional KV store

<u>Properties:</u>

- Distributed serializable Txs
- Confidentiality, integrity and secure persistency
- Performance

Treaty overview

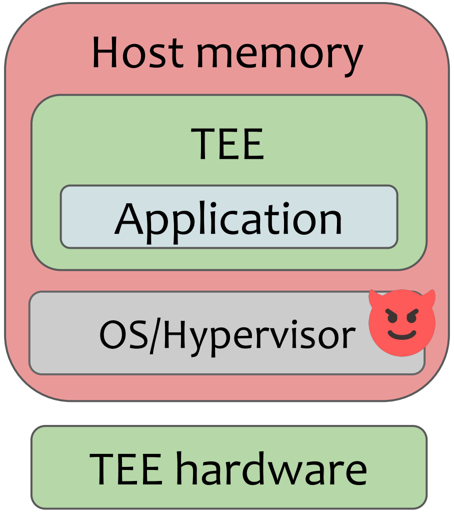


Outline

- ~~Motivation~~
- Background and challenges
- Design
- Implementation
- Evaluation

Trusted Execution Environment

- HW extensions for trusted computing
  - Intel SGX, Arm TrustZone, etc.
- Trusted area (enclave)
  - Integrity + confidentiality

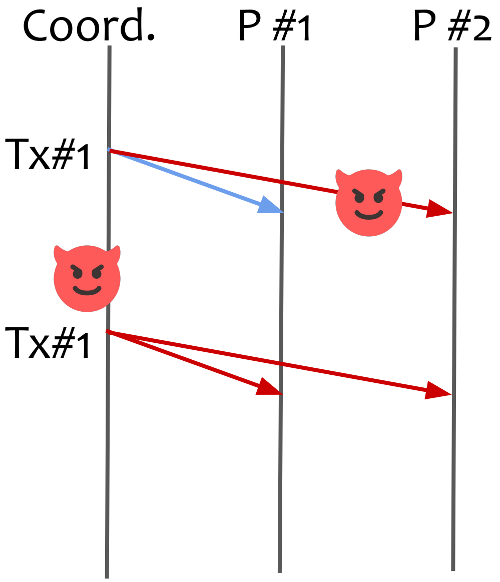


Treaty builds on TEEs to guarantee security for distributed Txs

Challenge #1: Distributed systems

- TEEs do not protect the network operations

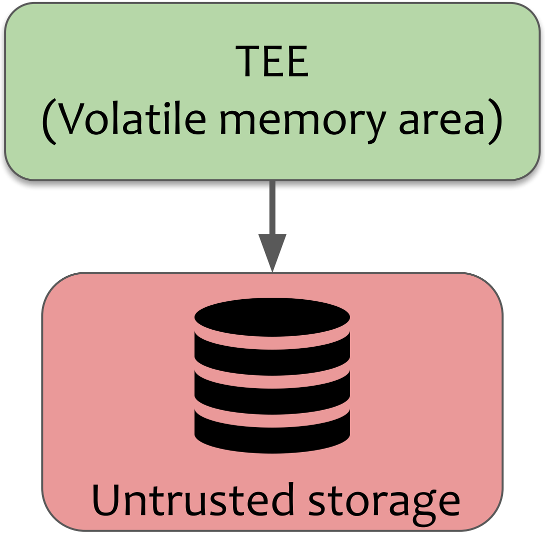
- Adversaries can tamper with Txs messages
  - integrity, confidentiality
  - replay-attacks



TEEs cannot guarantee secure execution for distributed Txs

Challenge #2: Stateful systems

- TEEs do not protect the persistent data and logs
- Adversaries can violate system correctness
  - delete or replace logs
  - compromise persistent data

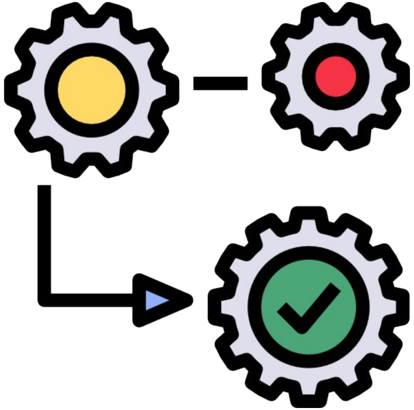


TEEs cannot guarantee secure persistency for committed Txs

Outline

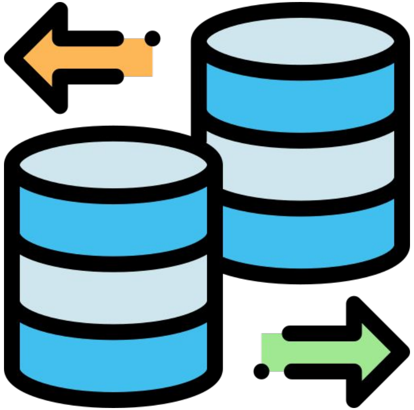
- ~~Motivation~~
- ~~Background and challenges~~
- Design
- Implementation
- Evaluation

Treaty



Secure Tx protocol

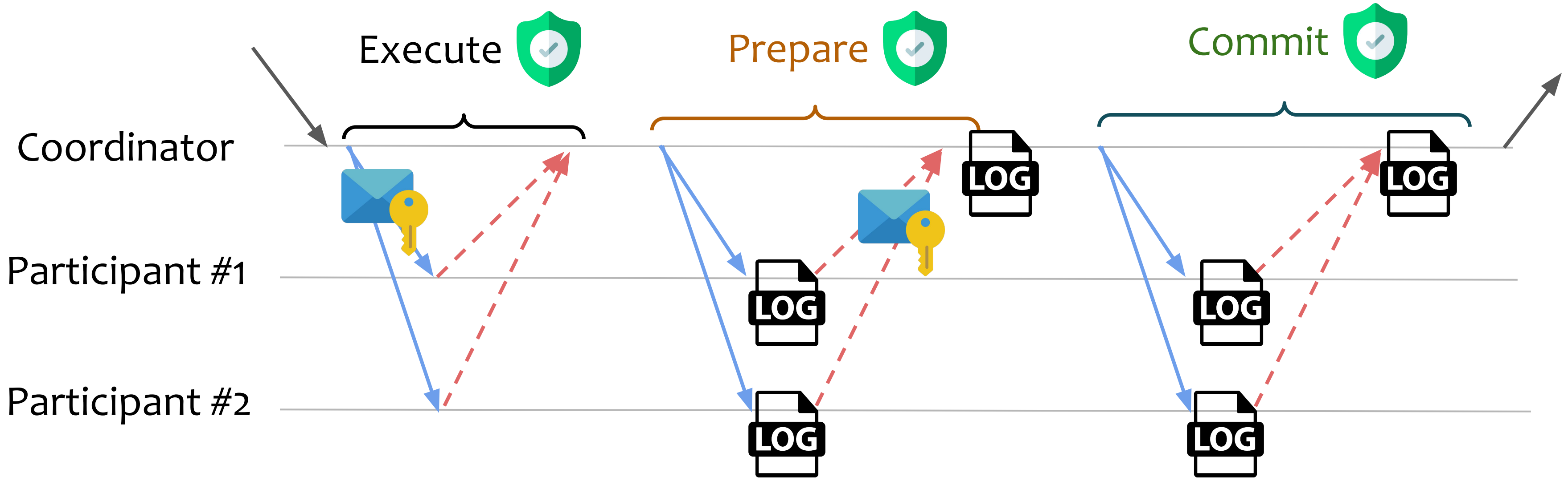
Secure execution



Stabilization protocol

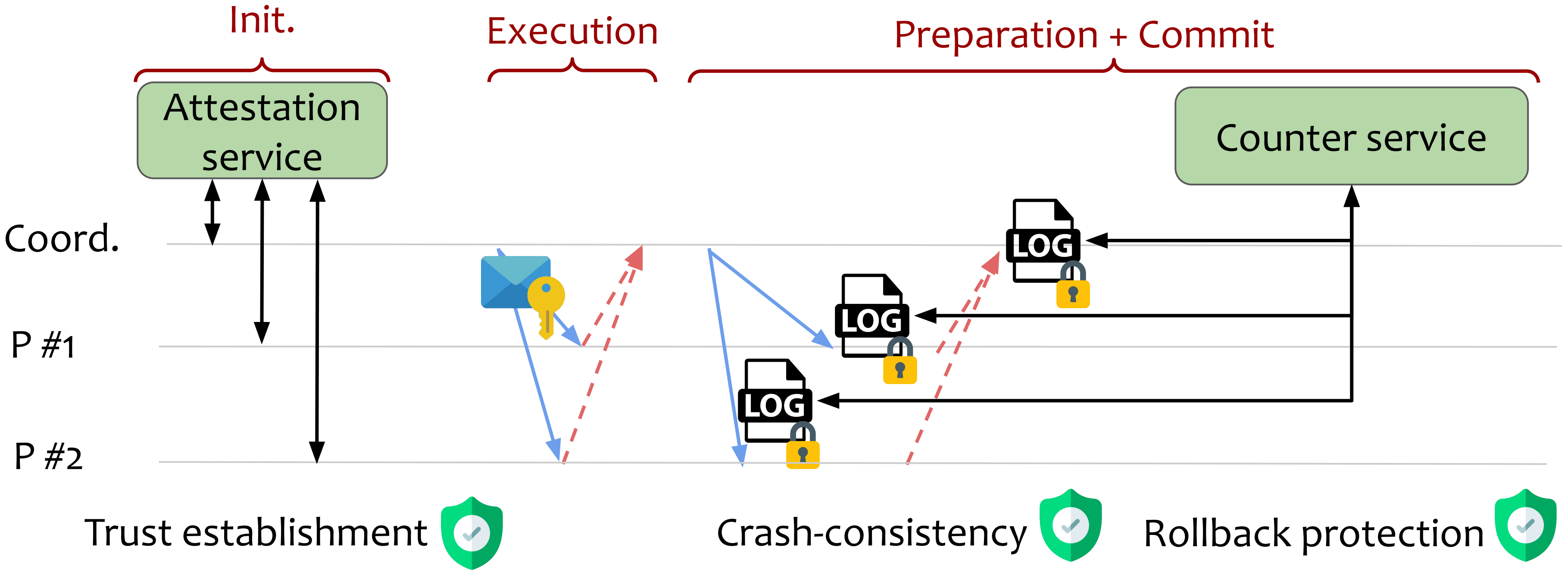
Secure persistency

#1: Secure Tx protocol



Treaty shields (a) the 2PC protocol and (b) the network messages for secure execution

#2: Stabilization protocol

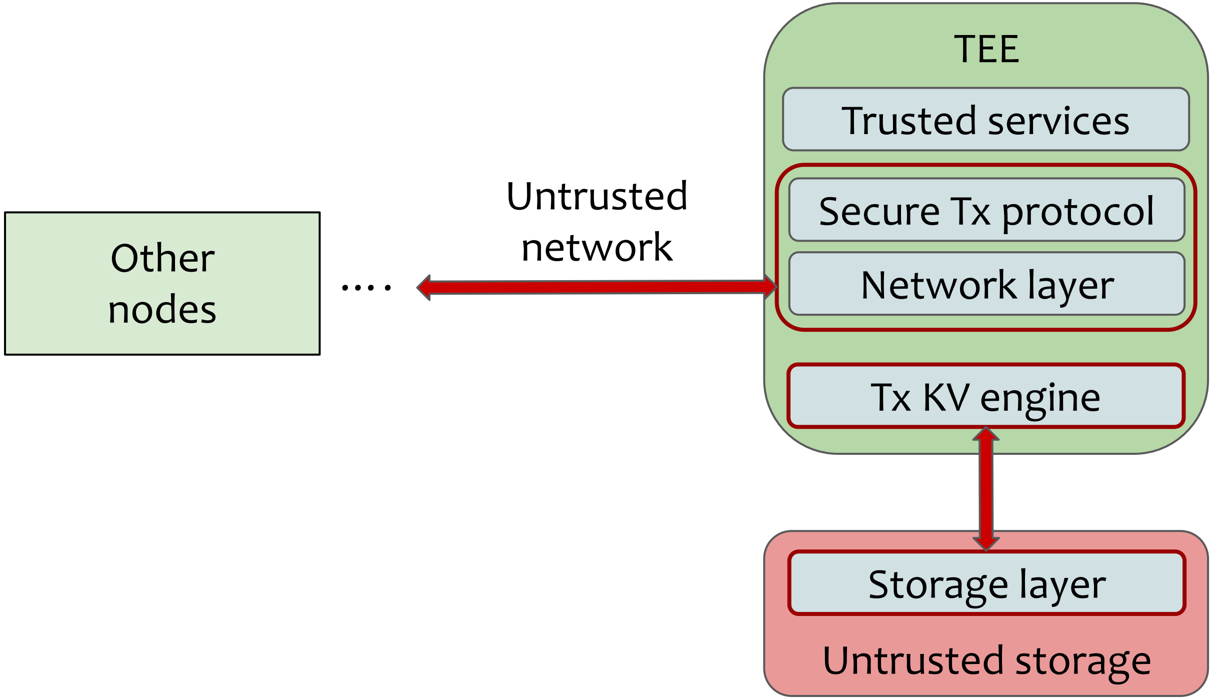


Treaty builds on (a) trusted services and (b) secure log files for secure persistency

Outline

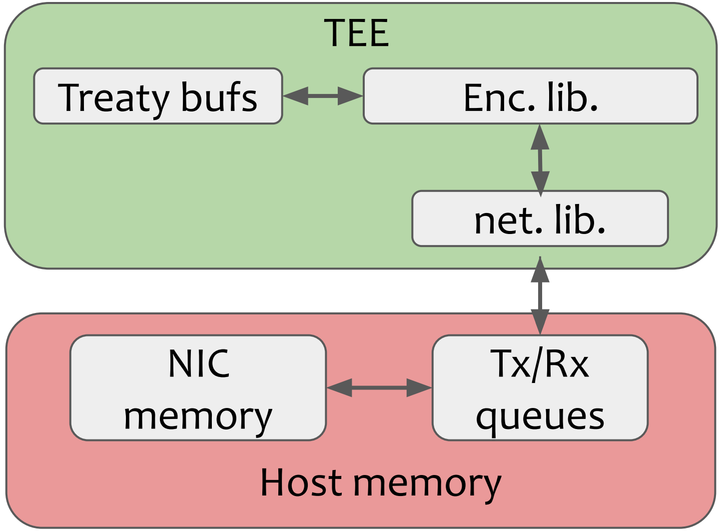
- ~~Motivation~~
- ~~Background and challenges~~
- ~~Design~~
- Implementation
- Evaluation

A Treaty node: System stack



Network layer

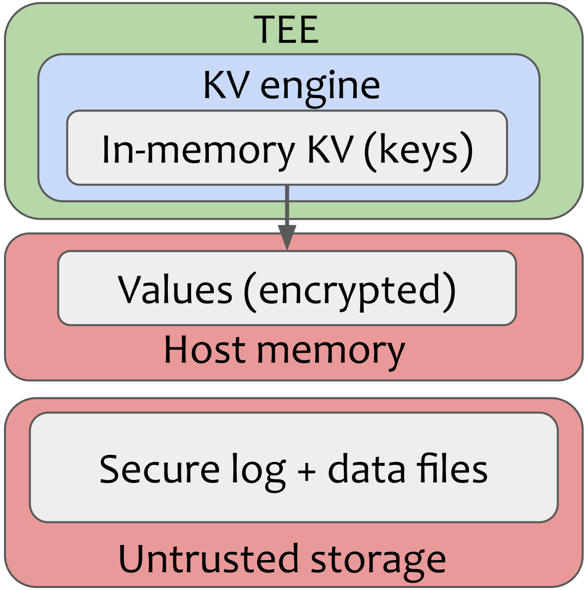
- Low-latency shielded communication
- Direct I/O within the TEE
- Metadata to prevent replay-attacks
- **Implemented** on top of RDMA/DPDK



Our network layer (a) optimises and (b) shields the network operations

Storage layer

- In-memory (hybrid) KV data structure
- Persistent data in authenticated files
- Pessimistic + optimistic single-node Txs
- **Implemented** on top of RocksDB



Our storage layer (a) secures the persistent data and (b) optimises the TEE usage

Outline

- ~~Motivation~~
- ~~Background and challenges~~
- ~~Design~~
- ~~Implementation~~
- Evaluation

Evaluation

Questions:

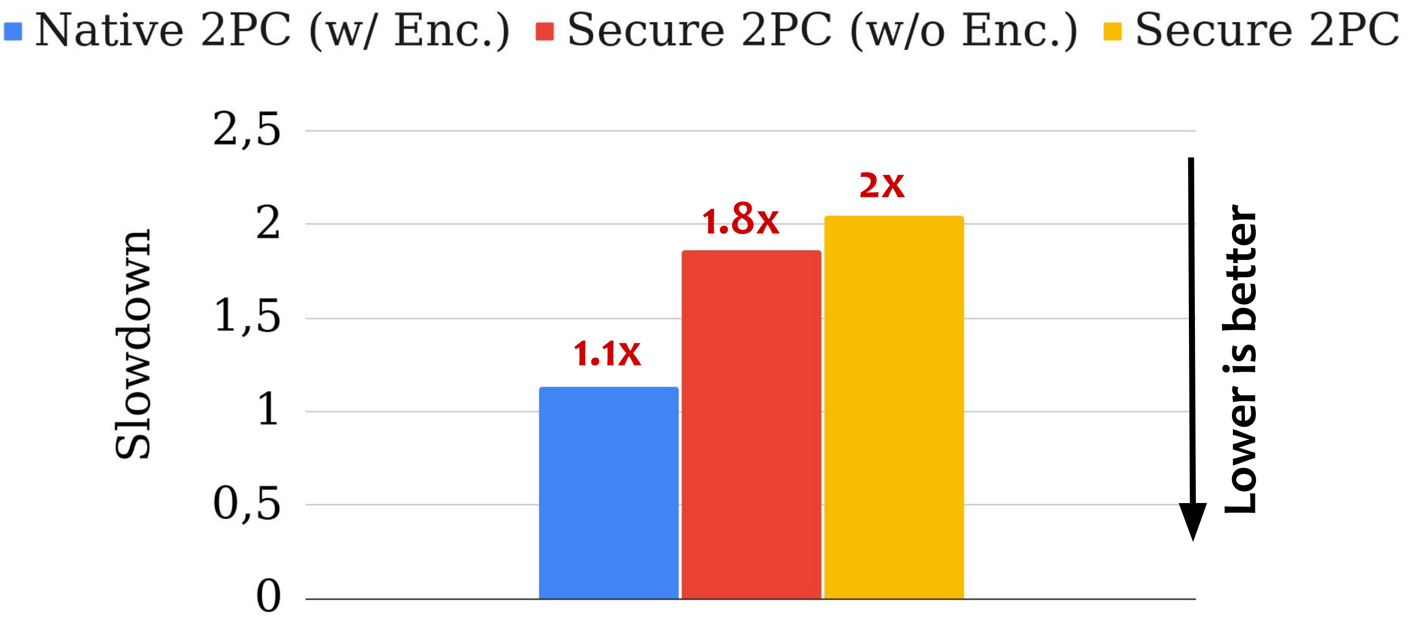
- What are the overheads of Treaty's 2PC (stand-alone)?
- What are the performance overheads for Treaty ?

More results in the paper!

Hardware setup:

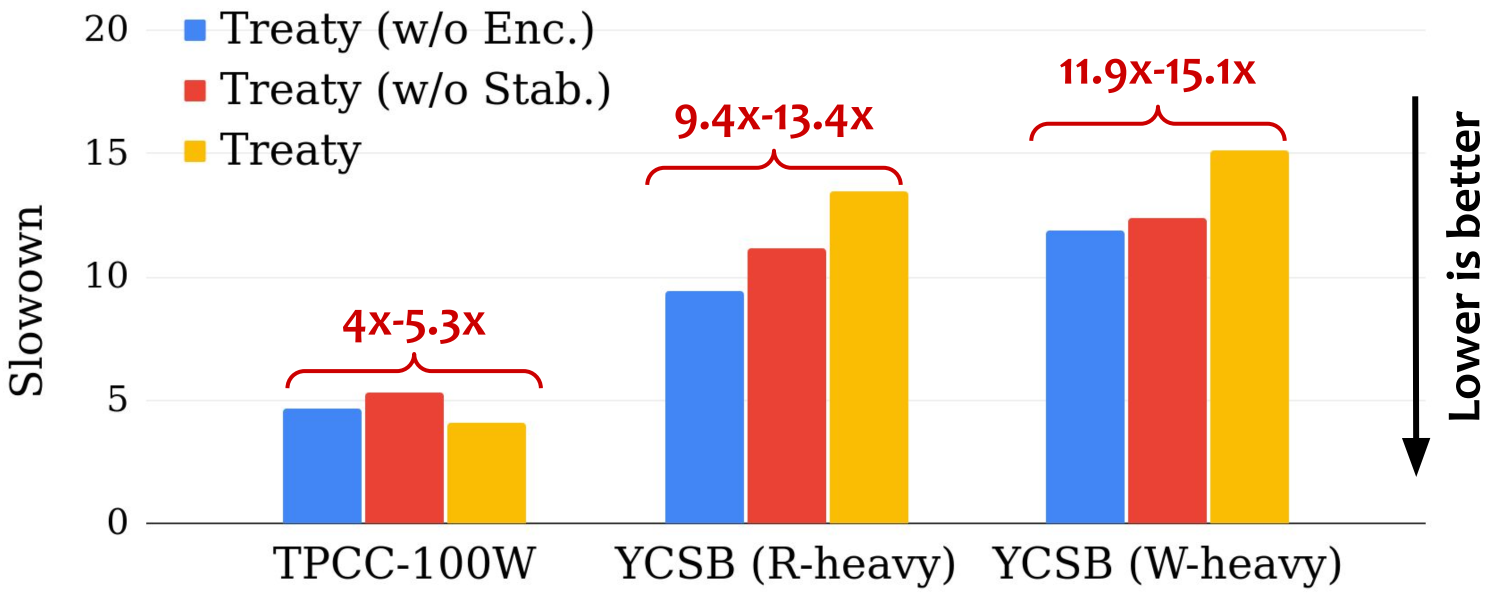
- TEE: Intel SGX
- 3x Intel i9-9900K (@3.60GHz, 8 cores, 16 HT)
- Intel NIC XL710 (40Gb/s, QSFP+)

Q1: 2PC's overheads



Treaty's 2PC overheads mainly derive from the TEE

Q2: Overall overheads



Treaty offers strong security w/ reasonable overheads w.r.t. the state-of-the-art

Summary

- Distributed Txs are an integral part of the third-party cloud infrastructure

- Secure transaction processing is challenging
  - TEEs are not designed for distributed systems with Txs and untrusted storage

- Treaty: A secure distributed Tx KV store with strong security guarantees
  - Secure 2PC protocol
  - Stabilization protocol
  - TEEs + direct I/O

Source code: https://github.com/TUM-DSE/Treaty

Backup slides

Is Treaty a viable solution?

Secure Tx systems	Obladi [OSDI' 18] (single-node)	Fabric [EuroSys' 18] (blockchain)	Treaty
Latency (ms)	~340	370-550	80-320
Secure storage systems	Speicher [FAST' 22] (single-node)	TWEEZER [FAST'22] (single-node)	Treaty
Tps overheads	15x-17x	4x-9x	4x-15x

Treaty incurs similar overheads with the state-of-art secure systems

Threat model

Threat model	Treaty
Compromised system stack (OS/hypervisor)	Yes
Network adversaries, (e.g., delay, drop, replay and manipulate network traffic)	Yes
Host memory memory manipulation	Yes
Unauthorized modifications to persistent storage	Yes
DoS	**No**
Cache-timing attacks (e.g., speculative execution, access pattern leakage, memory safety vulnerabilities)	**No**

EnclaveDB: A secure KVs with Txs

	EnclaveDB [SP'18]	Treaty
TEEs	Emulated h/w	Real h/w
Data model	In-memory KVs	Persistent KVs
Data distribution	No (single-node KVs)	Yes
Overheads	**1.4x**	**4x-15x**

EnclaveDB does not show the real TEEs' overheads

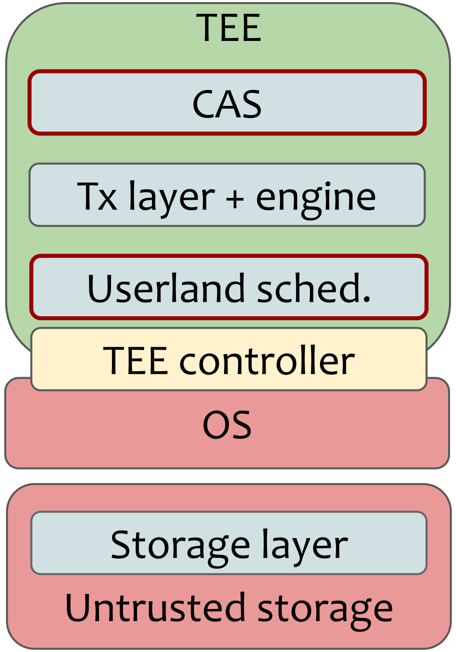
Speicher: A secure LSM-based storage system

	Speicher [FAST'19]	**Treaty**
TEEs	Real h/w	Real h/w
Data model	Persistent KVs	Persistent KVs
Data distribution	No (single-node KVs)	Yes
Txs	No	Yes
Overheads	**~15x**	**4x-15x**

Treaty shares similar overheads with state-of-the-art secure storage systems

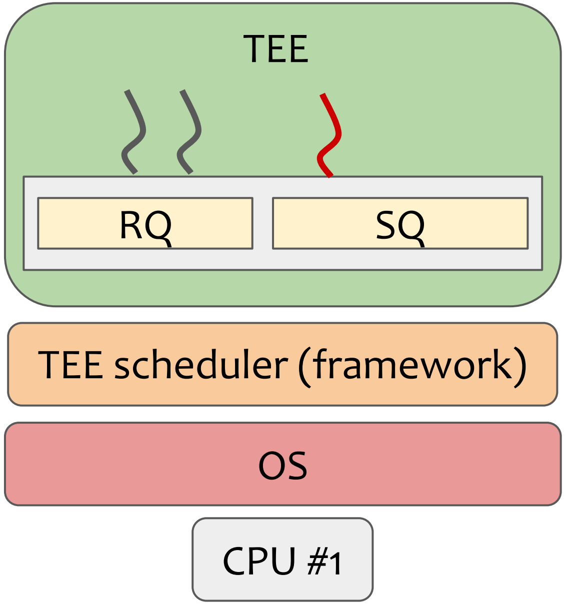
Trusted substrate for Txs

- Configuration and attestation service (CAS)
  - low-latency attestation
- Userland scheduler
  - low-latency operations
- Memory management
  - TEE memory usage

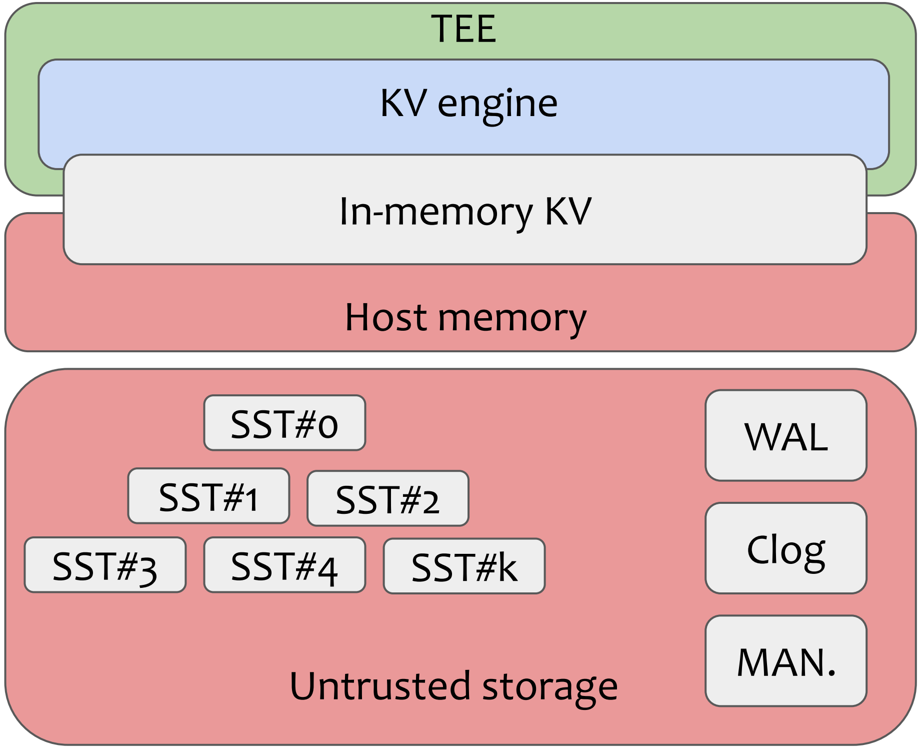


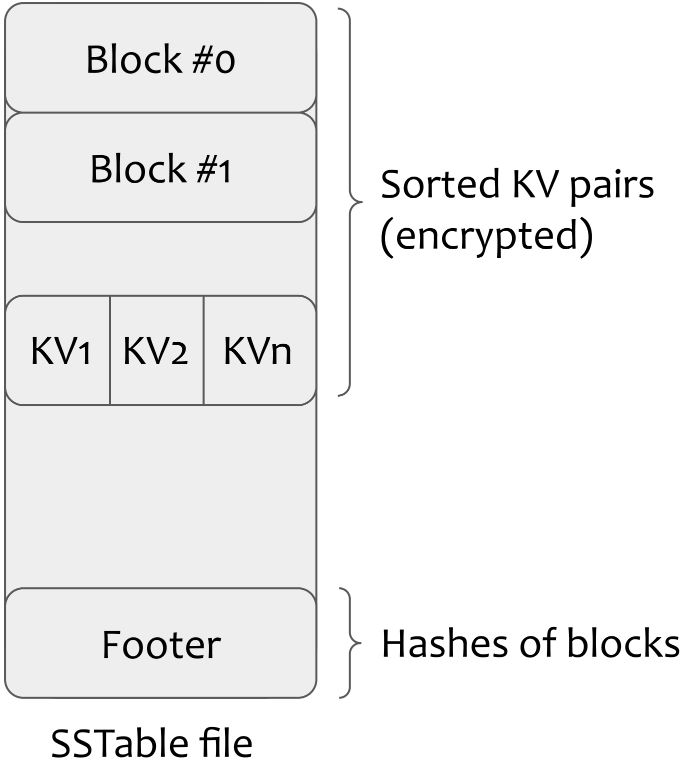
Userland scheduler

- Low-latency operations for multiple clients
- A userspace thread (fiber) for each client
- Lightweight context switches
    - Round-robin scheduling
    - No context-switches or interrupts



Authenticated LSM data structure





Log file and message format

Trusted id	Hash	Encrypted entry

Secure Log file format

IV	metadata	TX data	MAC

Secure message format