Smart contracts are user-defined programs that specify rules governing transactions that are enforced by a network of peers. They are programs on the blockchain consisting of program code, storage file, and an account balance. Anyone can create a smart contract by posting a transaction (with it) on the blockchain, and the program is invoked when it receives a transaction from a user.

The benefits of smart contracts are low legal and transaction costs compared to traditional financial contracts. Programming smart contracts raises new security challenges: if you load money in a smart contract with bugs, you may lose money. It also requires an “economic thinking” perspective that traditional programmers may not have acquired. Once a smart contract is created, it cannot be altered. Furthermore, to enable validation of state transitions during consensus, blockchain data is public. The consensus requirements may also affect the performance in terms of computational power, storage capacity, and transaction throughput, which hampers smart contracts. Proposed solutions to these problems are zero-knowledge proof systems and secure multiparity computation, but they are only applicable to limited use cases with relatively simple computations. Another proposition is to use a trusted execution environment (TEE), which is a fully isolated environment that prevents other software applications, the OS, and the host owner from tampering with or even learning the state of an application running in it. TEE’s needs to enforce critical delivery, like the atomicity property of the blockchain. Consider two messages, m1 the output for the user, and m2 the state update for the blockchain. Both messages need to be delivered and received, or else the blockchain is prone to attacks, such as rewind attacks.

A proposed solution, Ekiden, claims to ensure smart contracts deployed on it will have strong confidentiality, integrity, and availability guarantees. In Ekiden, request executions are decoupled from consensus. Some compute nodes execute the smart contracts, while

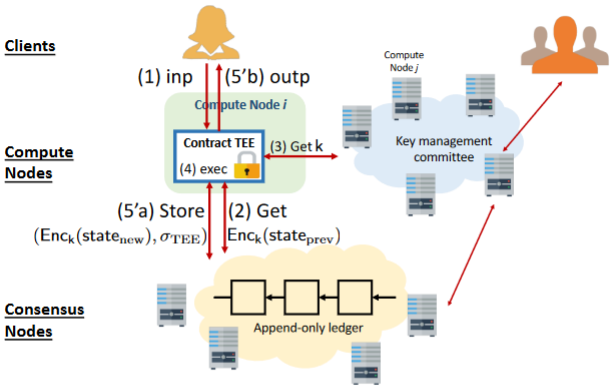


Fig. 1. Overview of Ekiden architecture and workflow. Clients send inputs to confidentiality-preserving smart contracts, which are executed within a TEE at any compute node. The blockchain stores encrypted contract state. See Section IV-B for an overview.

consensus nodes verify the proof of correct execution without repeating the execution itself.