Deniz Aytemiz

Ekiden: A Platform for Confidentiality-Preserving, Trustworthy, and Performant Smart Contracts

As blockchain technology gets more pervasive so does the technology of decentralized applications built on Ethereum commonly. Ethereum blockchain technology introduces the concept of smart contracts that are programs running in blocks. Data is public and visible in blockchain hence smart contracts inherently lack privacy or confidentiality. In addition to poor performance in terms of latency and throughput. In this paper, these drawbacks are discussed and a novel architecture that has better performance and confidentiality which is called 'Ekiden' is introduced.

Ekiden leverages Trusted Execution Environments (TEE) and blockchain technology together. A TEE is a fully protected environment in which an application running is not reachable from other software applications or even the operating system. The main idea behind the architecture of Ekiden is to create a hybrid from TEE and blockchain such that they would complement each other. For instance blockchain guarantees availability whereas TEE does not and TEE has minimal overhead while blockchain is computationally expensive. However one should be careful in harmonizing these technologies since naively combining them can also increase the attack surface of the application.

Unlike common smart contracts Ekiden is the first confidentiality preserving smart contract. Ekiden separates computation and consensus operations from each other. The computation of smart contracts are realized on compute nodes which are off-chain with private-data. So it is not visible or reachable from the public unlike blockchain. Blockchain in this architecture has the consensus nodes which validates the correct execution of smart contracts. Since the smart contracts are private and off-chain they are agnostic to consensus layer.

There are many technical challenges faced, discussed and optimized in the paper. First of all, the TEE environment is not perfect and is actually vulnerable from side channel attacks. Though protection against side channel attacks are mainly left for application level designers, the main idea of the authors is to limit the accessibility of the architecture in case of such an exploit. The key management system limits the access to the rest of the architecture providing this limitation of exploitation against adversaries. In addition, even a non-malicious host can lose enclaves in the power cycle. Therefore in the design of Ekiden, blockchain is used for resolving conflicts arising from concurrency. TEEs are stateless and any persistent state is held in the blockchain. Hence the system is resistant to lost enclave failures. Secondly, the blockchain's poor computation performance is the bottleneck of the architecture and hence it slows down the performance and adds complexity. Hence authors discuss how they implemented the system with smallest interactions to blockchain without sacrificing security of the whole architecture. Another issue is timer failures. In general TEEs do not have trusted time sources, therefore instead of synchronized clocks in Ekiden blockchain simply rejects any update on a stale input state. In addition, since TEEs provide confidentiality even if an attacker is able to alter a timer within the system the enclave will still be able to verify the contents in the blockchain. One other technical limitation is the persistence of keys. General method is to replicate keys within the system to support resilience however as a trade-off it increases the attack surface. In Ekiden the main approach is to again limit the area of exploitation such that an exploit would affect only a small portion of the architecture yet not the whole system. Lastly there is the issue of simultaneous delivery of executions namely ‘atomic delivery of execution’. In blockchain systems failure of execution of both or none creates a fundamental problem. In Ekiden in order to countermeasure this, message m1 is encrypted and sent to the client over a secure channel. After m1 is obtained the system is acknowledged by the client and then m2 is sent to the blockchain. After obtaining the validation for m2, the key for decrypting m1 is sent to the client, hence the system ensures both messages are delivered or none of them did.

Ekiden’s main architecture parts can be summarized as follows; clients, compute nodes and consensus nodes. Clients are end users and can create or execute smart contracts. The computing nodes are invoked in both cases. Compute nodes processes the request coming from the client and generates validation for state updates of the system. A part of compute nodes are responsible for obtaining and distributing keys used for contracts. The consensus nodes are in blockchain and they perform as distributed write only ledger running a consensus protocol as in typical blockchain environment.

The architecture has a lot of advantages with respect to standard smart contract systems. First of all the confidentiality achieved with separating consensus nodes from computation nodes and keeping the data agnostic to the system. In addition to, achieving 600x more throughput and 400x less latency at 1000x less cost than the Ethereum mainnet. It is shown that Ekiden can suit a range of different applications and it enables application deployment where current systems cannot due to confidentiality or privacy and performance concerns. It has its shortcomings as not being totally secured and requiring application level work for protection. The architecture's shortcomings seem like they are the inherent problems in both blockchain and TEE technologies.

From this paper I have learned about TEE technology along with its vulnerabilities and gained an understanding on how combining blockchain technology and TEE can result in vulnerabilities or how they can complement and strengthen each other. I also learned about smart contracts confidentiality vulnerabilities and what are the challenges when creating a system that would cover those vulnerabilities.