Summary Smart Contracts

Tim Sigl

In contrast to regular programming, programming smart contracts requires “economic thinking” to ensure fairness even in situations where users cheat and try to maximize their own profit. A smart contract is simply a computer program that runs on the blockchain and is executed by all consensus nodes. A smart contract can be created by posting a transaction with the compiled program code as the payload to the blockchain. The program code of the smart contract cannot be changed after being submitted to the blockchain. A contract can be executed by sending a transaction to the address of the contract. Contracts can be executed either by users or other contracts. The function to execute and input data to the smart contract is determined by the payload of the transaction invoking the smart contract. In Ethereum users have to pay Gas a fee for computational resource consumption for every instruction in the smart contract.
A common problem users run into when implementing smart contracts is that they reveal secret information. For example in games it is important that the opponent players don't know your next move before they have done theirs. Also programmers should be aware of edge cases where tokens can leak out of the contract or be stuck in the contract forever. There are a couple known Bugs like the call-stack bug where the maximum depth of the call stack is 1024 and all instructions increasing the depth won’t get executed.
In comparison to Bitcoins scripting language are Ethereum’s smart contracts turing complete. All state changes of the contract are recorded via transaction on the ledger. Ethereum has two kinds of accounts. One is managed directly by users via public-private key pairs and the other are contract accounts managed by smart contracts. The most popular programming language for smart contracts is Solidity. Smart contracts can have the same name since they are identified by their addresses. Versions of the Solidity compiler are updated regularly and normally users adapt quickly to new versions. 80% of the total balance of all smart contracts only belongs to 10 smart contracts. Smart contracts have on average only 321 lines of code. Most of the analyzed contracts don’t collect Ether inside.
Ekiden’s idea is to use trusted execution environments (TEEs) to make smart contracts more private, less computational intensive to increase TPS and decrease the strain on the underlying blockchain. Ekiden should theoretically be deployable on most blockchains. The usage of TEEs opens up new attack vectors against which new protection mechanisms are built. Ekiden decouples the execution of smart contracts and the consensus on a new state. Therefore they employ three entities: Clients (end users of smart contracts), compute nodes (which run the smart contract in a TEE) and consensus nodes (which maintain the ledger state). For the TEE to verify if an item has been stored on the blockchain they employ a nover time-based proof-of-publication protocol. Also they use a key management system to generate key pairs for the encryption of the TEEs. Atomic delivery has to be guaranteed, meaning that messages will be delivered to the caller and the blockchain as a state update.
There are a few misconceptions when it comes to the capabilities of smart contracts. First is the usage of external services. Since all computation in smart contracts has to be deterministic the input from an external source cannot be different for execution of the same contract on different nodes. As a workaround, trusted third parties so called ‘oracles’ have been established that publish external data as transactions on the ledger for all nodes to use. The issue with this approach is the use of a often centralized data provider. Another

misconception is the possibility to enforce on-chain payments. Because the smart contract is the holder of the funds, the owner cannot directly access the funds making the interaction with customs more difficult. The last misconception a lot of people fall for is the missing feature to hide confidential data. Since all state updates are directly saved on the public ledger there is no option to hide data.