# zkMove: a zero-knowledge proof-based smart contract runtime environment

| Author | Version | Date |
| --- | --- | --- |
| Guangyu Zhu | Draft v0.1 | Jul.7.2021 |
| Guangyu Zhu | Draft v0.2 | Dec.10.2021 |

## 1.Background

With the boom of Defi and the emergence of non-financial smart contracts, the scalability of public chains, represented by Ethereum, is being increasingly challenged. Although technologies such as POS and sharding can improve the throughput rate to a certain extent, the root cause of congestion still exists in the long run. This is because any transaction that wants to be on the chain requires most nodes across the network to verify its legitimacy, and the method of verification is to execute the transaction repeatedly. As the number of applications grows exponentially and the logic of smart contracts becomes more complex, the computational resources required to verify their legitimacy will increase exponentially, which is reflected in network congestion and high transaction fee.

## 2.Basic ideas

> Improving scalability through zero-knowledge proof technology

To fundamentally improve the scalability of the blockchain, we propose zkMove - a zero-knowledge proof-based smart contract runtime environment that combines Move [1], the most secure smart contract programming language, with PLONK [2], a maturing zero-knowledge proof technology, to "Move" computation from on-chain to off-chain, significantly improving the scalability of blockchain while ensuring security.

> Enhancing programmability through zero-knowledge proof virtual machine

Currently on Ethereum, projects using zero-knowledge proofs include Loopring and ZKSwap, which only support single application scenarios and do not have programmability. zkSync [3] and StarkNet [4] are designed to provide a programmable scaling solution, and their multi-application scenarios are still under development. zkMove hopes to combine years of experience in programming language virtual machines with zero-knowledge proof cryptography to create a Turing-complete zero-knowledge proof virtual machine that allows smart contracts to be deployed directly through the virtual machine without developing separate circuits.

> Providing security beyond the blockchain through the Move language

First, the security of the blockchain is inherited through zero-knowledge proof technology. With zkMove as the foundation, it is easy to build various Layer2 solutions on the main chain, so that users do not have to monitor the network all the time, no one or organization can steal user assets or destroy user state, and users can withdraw assets unconditionally at any time. Secondly, it goes beyond mainchain security with the Move language. zkMove uses Move, a new generation smart contract programming language for digital assets, combined with tools such as formal verification, which can further enhance the security of smart contracts.

> Become a cross-blockchain smart contract runtime environment

Unlike existing Layer2 solutions, zkMove does not position itself as Layer2 of some public blockchain, but as a cross-blockchain environment for running smart contracts. A smart contract running on zkMove can directly interact with another smart contract running on zkMove, regardless of the underlying blockchain, even if the underlying layer is not blockchain dependent.

## 3.Working principle

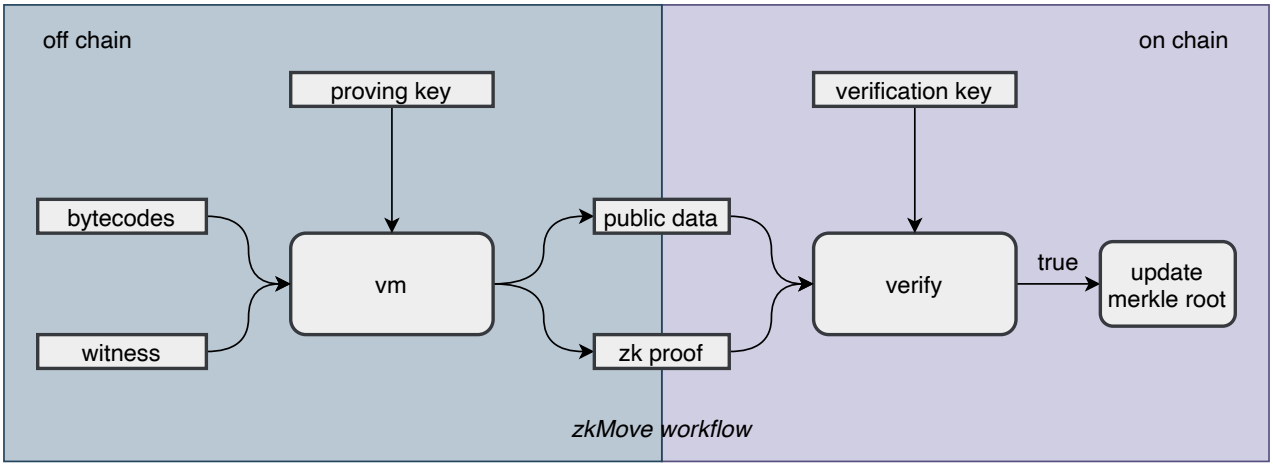
We will take zkMove's most typical usage scenario, zk-rollup, as an example to illustrate how it works. From the perspective of distributed computing, a blockchain is a replicated state machine, where S represents the current account state, which changes to S' when the transaction txn is executed.

$$STF(S,\ txn) \rightarrow S'$$

In order to "Move" the computation from on-chain to off-chain, it is necessary to move the account state S to off-chain and maintain it with merkle tree, and the signature verification and execution of user transactions are performed off-chain. The user synchronizes his account state to the chain only when he needs it, otherwise only the Merkle tree root R of the state is uploaded to the chain. The correctness of the state is guaranteed by the merkle_proof of the accounts involved in the transaction. This process can be represented by the following replicated state machine.

$$STF(R,\ accounts,\ merkle\_proof,\ txn)\ \rightarrow\ R'$$

In order to "Move" the computation from on-chain to off-chain, user transactions need to be executed off-chain in the order they are submitted, and zero-knowledge proof zk proof and encoded operation records are generated, and then the run results and zk proof are submitted to the chain. A smart contract on the chain verifies the zk proof, and if it passes, the user's transaction is indeed executed correctly, and then the latest Merkle root R' is recorded. The encoded operation record is uploaded to the chain as the parameter of the smart contract verifying the zk proof.




The above diagram depicts a typical workflow of zkMove. At its core is a bytecode virtual machine, and the bytecode conforms to the Move specification. Move is a new generation of smart contract programming language for digital assets, and its security and formal verification features largely satisfy the requirements of zkMove. The witness is the input to the transaction, which typically contains the accounts involved in the transaction, the Merkle proof, and the root of the state tree before the transaction is executed. The public data is the output of the transaction, which typically contains the root of the new state tree after the transaction is executed. zkMove uses the PLONK zero-knowledge proof algorithm, which requires only one trusted initial setup, and the proving key and verification key are generated at the time of the smart contract is published.

## 4.Project progress

As of the release of draft v0.2 of this document, we are nearing completion of the first phase of the POC for the zkMove zero-knowledge proof virtual machine, where non-Turing-complete Move smart contracts can execute correctly and their zk proofs can be generated and verified correctly. In the next phase, we will implement Turing-complete while improving the existing features, and more detailed information will be released later.。

## 5.References

[1] Sam Blackshear, Evan Cheng, David L. Dill, Victor Gao, Ben Maurer, Todd Nowacki, Alistair Pott, Shaz Qadeer, Rain, Dario Russi, Stephane Sezer, Tim Zakian, Runtian Zhou *Move: A Language With Programmable Resources*

[2] Ariel Gabizon, Zachary J. Williamson and Oana Ciobotaru *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*

[3] Alex Gluchowski *Introduction to zkSync*

[4] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh and Michael Riabzev *Scalable, transparent, and post-quantum secure computational integrity*