CODE EATER

zkEVM Notes

Prerequisite Videos

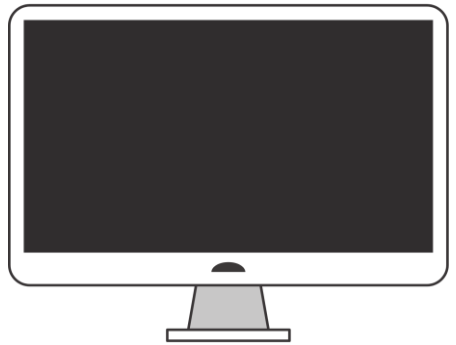
- Click Here – Zk Proof
- Click Here – Layers Of Blockchain

zkEVM

Zero Knowledge Proof + EVM

How Program is Executed?

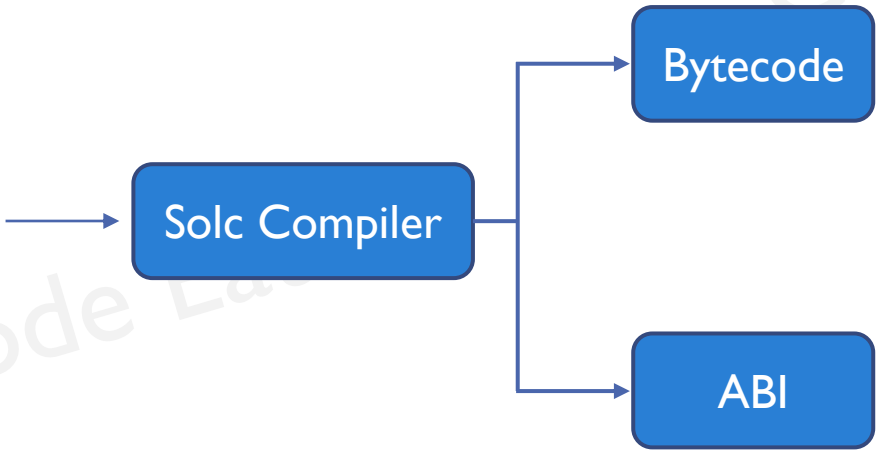
Program



Computer
(Machine)

What is EVM?

```
1  // SPDX-License-Identifier: GPL-3.0
2
3  pragma solidity >=0.8.2 <0.9.0;
4
5  contract Storage {
6
7      uint256 number;
8      function store(uint256 num) public {
9          number = num;
10     }
11
12     function retrieve() public view returns (uint256){
13         return number;
14     }
15 }
```

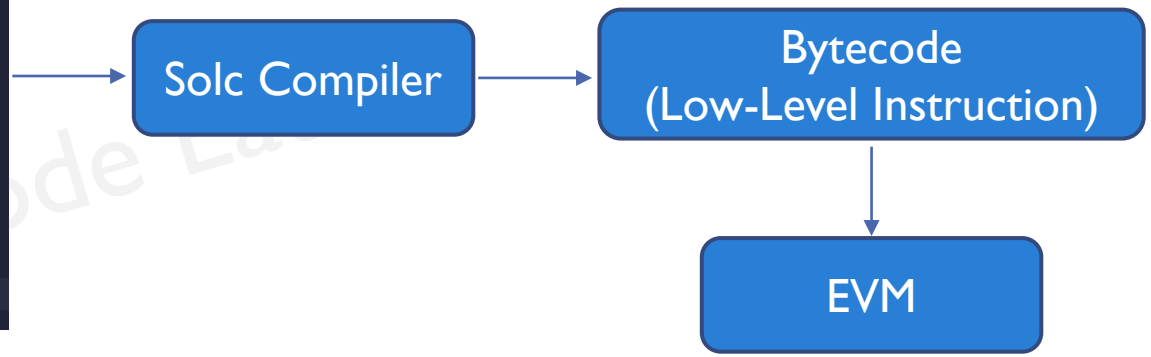


Smart Contract(Program)

What is EVM?

```
1  // SPDX-License-Identifier: GPL-3.0
2
3  pragma solidity >=0.8.2 <0.9.0;
4
5  contract Storage {
6
7      uint256 number;
8      function store(uint256 num) public {
9          number = num;
10     }
11
12     function retrieve() public view returns (uint256){
13         return number;
14     }
15 }
```

Smart Contract(Program)



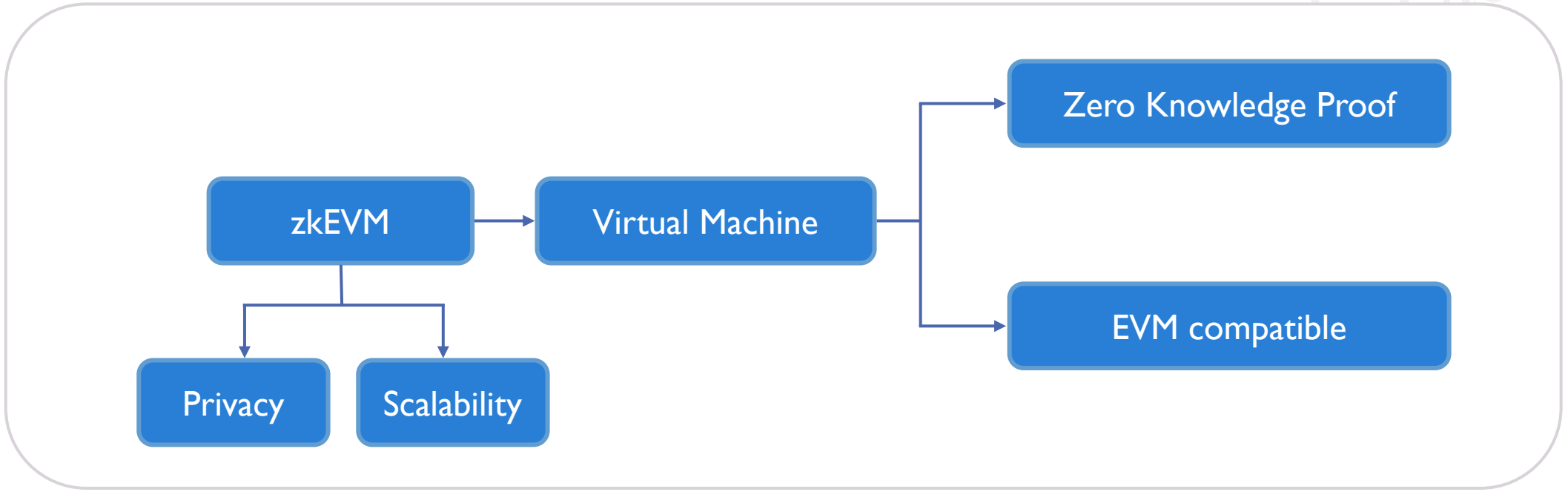
EVM

- The Ethereum Virtual Machine (EVM) is a **decentralized virtual machine that executes smart contracts** on the Ethereum blockchain. It is the runtime environment for Ethereum's smart contracts and is responsible for processing transactions and executing the code of smart contracts.
- The **EVM is a software-based emulation of a physical computer**, similar to other virtual machines. It has its own instruction set and operates independently of the underlying hardware. The EVM is designed to be secure, deterministic, and gas-efficient.
- Smart contracts are written in high-level programming languages such as Solidity and compiled into EVM bytecode. This bytecode can then be deployed to the Ethereum network and executed by the EVM.
- When a transaction is submitted to the Ethereum network, the EVM verifies that the transaction is valid and executes the corresponding smart contract code. The EVM maintains a record of all transactions and contract states on the Ethereum blockchain.

Zero Knowledge Proof

- A zero-knowledge protocol is a method by which one party (the prover) can prove to another party (the verifier) that something is true, without revealing any information apart from the fact that this specific statement is true.

zkEVM in one slide



Layer 2 solution

Virtual Machine

- In computing, a virtual machine (VM) is a software-based emulation of a physical computer that can run an operating system and execute applications as if it were a real machine.
- A virtual machine creates a virtual environment that behaves like a real computer system, including a virtual CPU, memory, storage, and other components. This enables multiple virtual machines to run simultaneously on a single physical machine, allowing for better resource utilization and more efficient use of hardware.

EVM compatibility

- The EVM (Ethereum Virtual Machine) is the runtime environment in which smart contracts deployed on the Ethereum network are executed.
- **A virtual machine is "EVM-compatible" if it can run programs created to run in the EVM environment.**
- Such VMs can execute smart contracts written in Solidity or other high-level languages used in Ethereum development. **ZkEVMs are EVM-compatible because they can execute Ethereum smart contracts without extensive modifications of the underlying logic.**

Use Cases of zkEVM

- DeFi Applications
- NFT Marketplace
- Gaming Industry


NFT


DeFi
DECENTRALIZED FINANCE

zkEVM

- A zkEVM is an **EVM-compatible virtual machine** that supports **zero-knowledge-proof computation**. Unlike regular virtual machines, a zkEVM proves the correctness of program execution, including the validity of inputs and outputs used in the operation.
- ZK EVM (Zero-Knowledge Ethereum Virtual Machine) is a **layer 2 scaling solution** that aims to improve the scalability and privacy of the Ethereum network by leveraging **zero-knowledge proofs.**
- **In the ZK EVM, smart contracts are executed off-chain and only the final result is stored on the Ethereum blockchain**, reducing the amount of data that needs to be stored and processed. This makes the system more scalable and reduces transaction fees.
- **It does this by using zero-knowledge proofs**, which allow users to prove the validity of a computation without revealing any of the input or output data.

Example

- Polygon ZkEVM

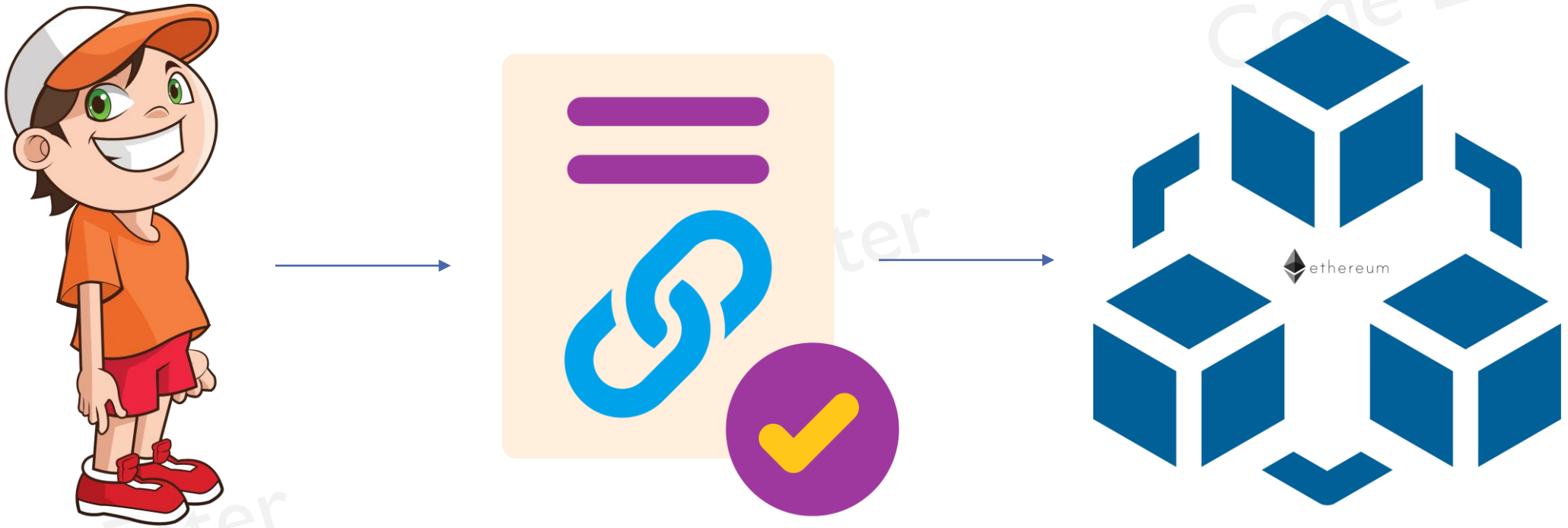
What is Polygon zkEVM?

- Polygon zkEVM is the first zero-knowledge scaling solution that is **fully equivalent to an EVM**. All existing smart contracts, developer tooling, and wallets work seamlessly. Polygon zkEVM harnesses the power of zero-knowledge proofs in order to reduce transaction costs and massively increase throughput, all while inheriting the security of Ethereum.

- Building dApps on zkEVM is completely similar to Ethereum. Simply switch to the zkEVM RPC and start building on a network with much higher throughput and lower fees. Polygon zkEVM provides a complete EVM-like experience for Developers and Users alike. So you do not need special toolings or new wallets for building or interacting with zkEVM.

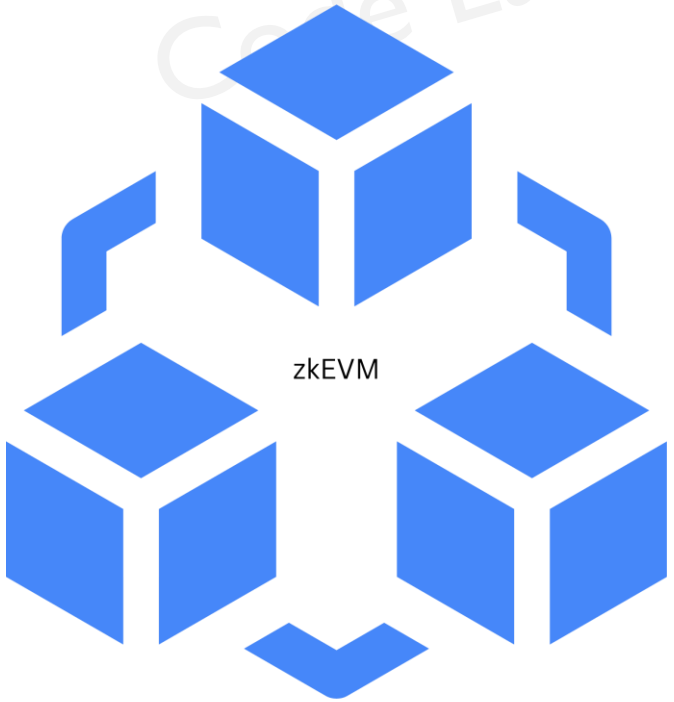
- Click Here - PolygonZkEVM

zkEVM working Example

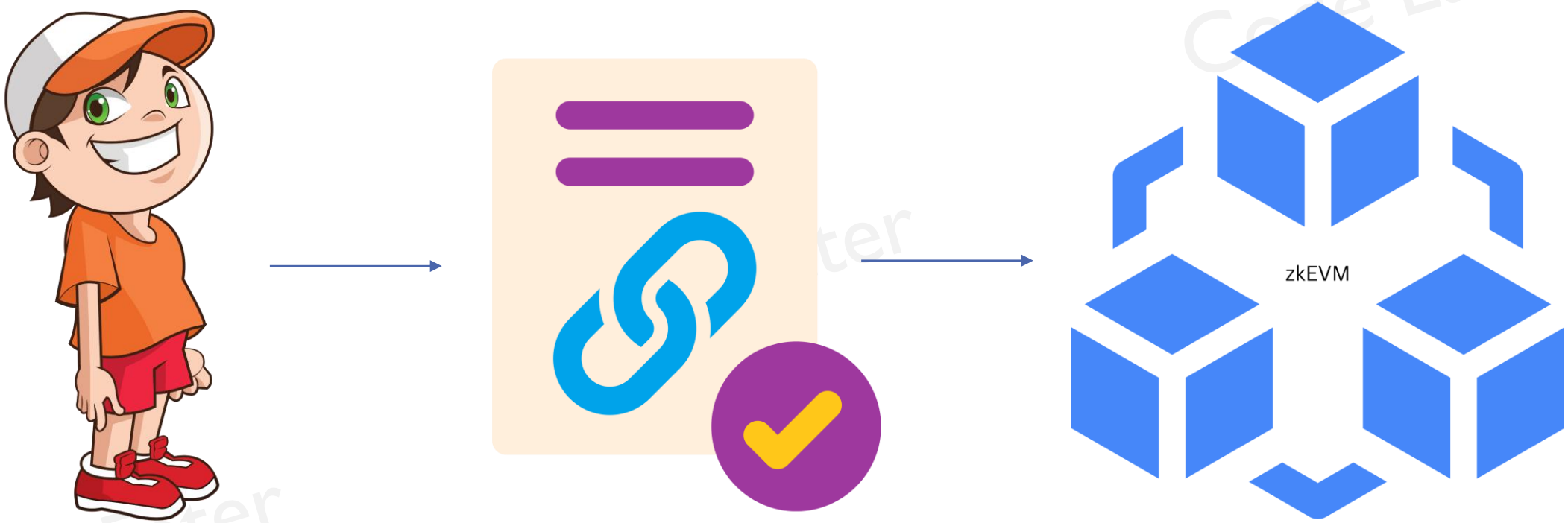


zkEVM working Example

zkEVM working Example



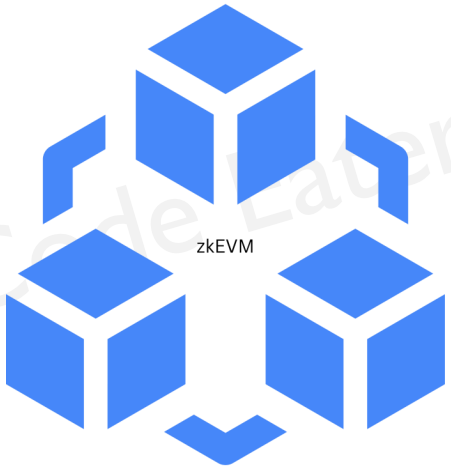
zkEVM working Example



zkEVM working Example

Zk Proofs +
Cryptographic Techniques

Validation



Zk proof +
Transaction

zkEVM working Example

- Here is an example of how ZK-EVM can use zero-knowledge proofs to enable off-chain processing of smart contract computations:
- Let's say Pintu wants to execute a smart contract on the Ethereum network using ZK-EVM. Normally, this would require Pintu to submit the smart contract code and all input data to the Ethereum network for processing, which can be slow and expensive, especially if the smart contract involves complex computations.
- With ZK-EVM, Pintu can instead submit the smart contract code and input data to a ZK-EVM node, which will use zero-knowledge proofs to verify the correctness of the computation off-chain. Specifically, the ZK-EVM node will use a combination of cryptographic techniques to prove that the smart contract code was executed correctly and that the output is valid, without revealing any of the intermediate computation steps.
- Once the zero-knowledge proof has been generated and verified, the ZK-EVM node will submit the transaction to the Ethereum network and the proof for final confirmation. Because the computation has already been verified off-chain, the Ethereum network only needs to validate the proof and record the transaction, rather than re-executing the entire smart contract computation.
- This off-chain processing of smart contract computations allows for much faster transaction processing times and significantly increases the scalability of the Ethereum network.

zkEVM architecture

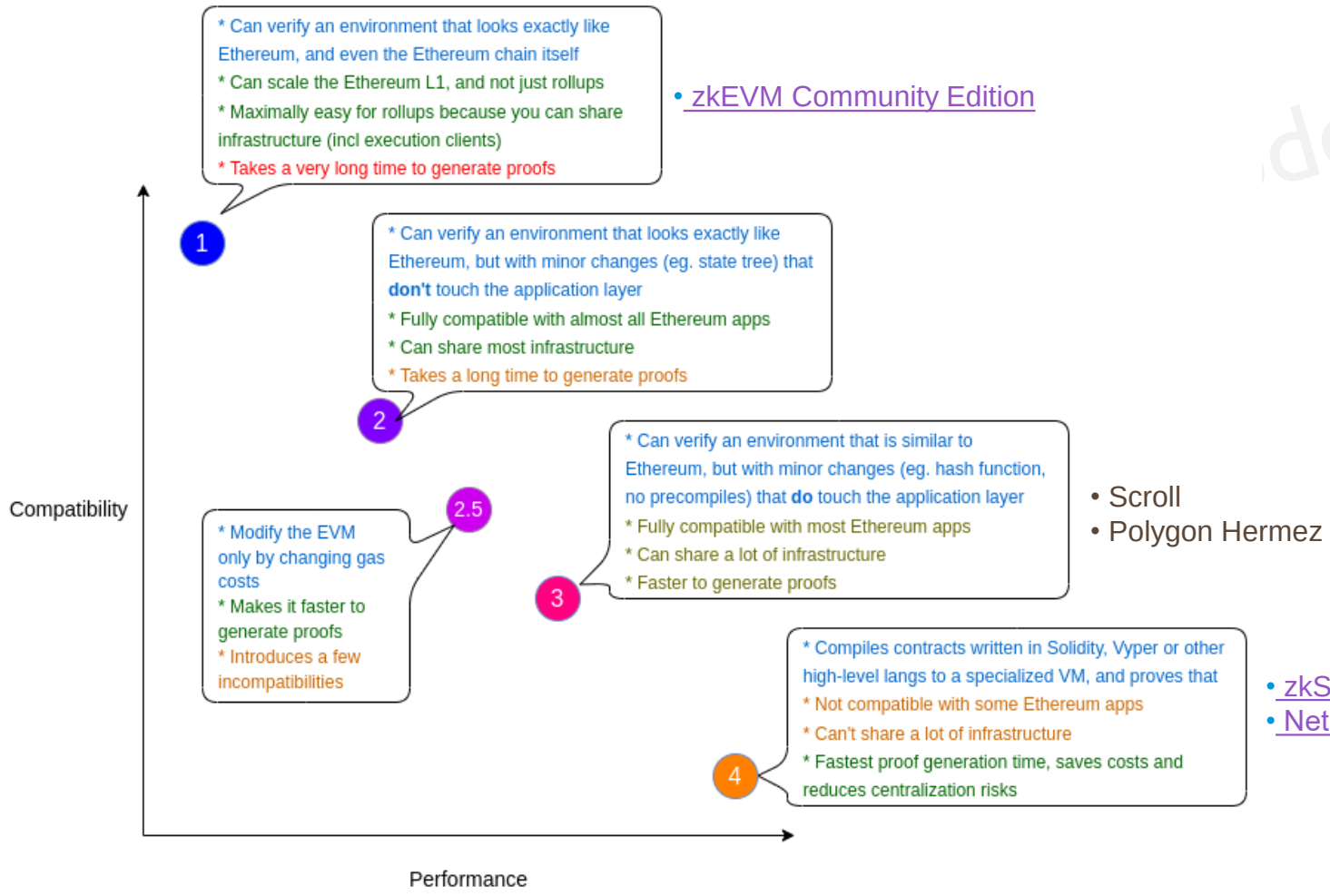
- The Execution environment
- The Proving circuit
- The verifier contract

zkEVM architecture

The zkEVM is divided into three parts: an execution environment, proving circuit, and verifier contract. Each component contributes to the zkEVM's program execution, proof generation, and proof verification.

- **1. The execution environment -** As the name suggests, the execution environment is where programs (smart contracts) are run in the zkEVM. The zkEVM's execution environment functions much like the EVM: it takes the initial state and current transaction to output a new (final) state.
- **2. The proving circuit -** The proving circuit produces zero-knowledge proofs verifying the validity of transactions computed in the execution environment.
- **3. The verifier contract -** ZK-rollups submit validity proofs to a smart contract deployed on the L1 chain (Ethereum) for verification. The input (pre-states and transaction information) and output (final states) are also submitted to the verifier contract. Then the verifier runs computation on the provided proof and confirms that the submitted outputs were correctly computed from the inputs.

Types of zkEVM



- zkEVM Community Edition
- Scroll
- Polygon Hermez
- zkSync 2.0
- Nethermind's Warp

Summary of zkEVM types by Vitalik Buterin (source)

Types of zkEVMs

- Current zkEVM projects fall into two main categories: zkVMs supporting native EVM opcodes and zkVMs using customized EVM opcodes.

Polygon zkEVM

- Polygon Hermez is a Polygon ZK-rollup with a zero-knowledge virtual machine designed to support EVM compatibility. To do this, **EVM bytecode is compiled into "micro opcodes"** and executed in the uVM—a virtual machine that uses SNARK and STARK proofs to verify the correctness of program execution.
- The decision to combine the two proof types is strategic. STARK (Scalable Transparent Argument of Knowledge) proofs are faster to generate, but SNARK (Succinct Non-Interactive Argument of Knowledge) proofs are smaller and cheaper to verify on Ethereum.
- The Polygon Hermez zkEVM uses a STARK proving circuit to generate proofs of validity for state transitions. A STARK proof verifies the correctness of STARK proofs (think of it as generating "proof of a proof") and is submitted to Ethereum for verification.

Types of zkEVMs

- **zkSync zkEVM**
- zkSync is an EVM-compatible ZK-rollup developed by Matter Labs and powered by its own zkEVM. ZkSync achieves compatibility with Ethereum using the following strategy:
- 1. **Compiling contract code written in Solidity to Yul, an intermediate language that can be compiled into bytecode for different virtual machines.**
- 2. Re-compiling the Yul bytecode (using the LLVM framework) to a custom, circuit-compatible bytecode set specially designed for zkSync's zkEVM.
- Like Polygon Hermez, the zkSync zkEVM achieves EVM compatibility at the language level, not the bytecode level. For example, traditional multiplication and addition opcodes (ADDMOD, SMOD, MULMOD) are not supported by zkSync's zkEVM.

1) The smart contract is deployed on ZkEVM or EVM?

- In the case of ZK-EVM, the smart contract is deployed on the ZK-EVM sidechain, rather than the main Ethereum network.
- ZK-EVM is essentially a separate virtual machine that can run smart contracts in a privacy-preserving manner. When Pintu submits a smart contract to ZK-EVM for execution, it is deployed on the ZK-EVM sidechain and executed off-chain using zero-knowledge proofs. Once the computation is verified and confirmed, the results are sent back to the Ethereum network as a transaction for final validation and recording on the Ethereum blockchain.
- It's important to note that while ZK-EVM and the Ethereum network are separate, they are still connected, and transactions between them are facilitated by specialized smart contracts called relayers. Relayers are responsible for receiving transaction data from ZK-EVM, verifying the zero-knowledge proofs, and then forwarding the transaction data to the Ethereum network for final confirmation.
- So, in summary, smart contracts are deployed on ZK-EVM, a separate sidechain from the main Ethereum network, and are executed off-chain using zero-knowledge proofs. Once the computation is confirmed, the results are sent back to the Ethereum network for final validation and recording.

2) What Ethereum records and validate if everything is executed on ZkEVM and stored on ZKEVM?

- While ZK-EVM is a separate sidechain from the main Ethereum network, it is still designed to be interoperable with Ethereum, which means that transactions and data can be transferred between the two networks.
- When a transaction is executed on ZK-EVM and the results are confirmed, a corresponding transaction is created on the Ethereum network that records the final state of the ZK-EVM sidechain. This transaction serves as a proof of the computation that was executed on ZK-EVM, and it is validated by the Ethereum network to ensure that it is correct.
- So, even though the actual execution and storage of the smart contract happens on ZK-EVM, the final state of the computation is still recorded on the Ethereum network for transparency and security purposes. This allows Ethereum users to verify that the computation was executed correctly and that the results are valid, even if they don't have access to the details of the off-chain computation.
- Additionally, the interoperability between ZK-EVM and Ethereum allows for greater flexibility in terms of how smart contracts are deployed and executed. For example, a smart contract could be deployed on ZK-EVM for privacy reasons and then later moved to the Ethereum network for greater transparency and accessibility.

3) Is zk rollups being used to submit a batch of transactions at once?

- Yes, zk rollups can be used to submit a batch of transactions at once. This is because zk rollups allow for off-chain processing of multiple transactions, which can then be bundled together and submitted to the main chain as a single batch transaction.
- In a zk rollup system, a group of transactions is processed off-chain using zero-knowledge proofs to generate a single proof that represents the validity of all of the transactions in the group. This proof is then submitted to the main chain as a single transaction, which saves significant gas costs and reduces the load on the main chain.
- By batching multiple transactions together in this way, zk rollups can greatly improve the scalability of the Ethereum network, allowing for faster and more efficient processing of transactions.

4) What is this verifier contract deployed on L1 (Ethereum) chain?

- The smart contract deployed on the L1 (layer 1) chain in a zk-rollup system is known as the verifier contract. It is responsible for verifying the validity of the transactions that were processed off-chain in the zk-rollup.
- When a group of transactions is processed off-chain in a zk-rollup, a zero-knowledge proof is generated to demonstrate the validity of all the transactions in the group. This proof is then submitted to the verifier contract on the L1 chain along with the inputs and outputs of the transactions.
- The verifier contract is designed to verify the validity of the zero-knowledge proof using the inputs and outputs of the transactions, and to ensure that the transactions are consistent with the current state of the L1 chain. If the proof is valid and the transactions are consistent with the L1 chain state, then the transactions are considered valid and the verifier contract updates the L1 chain state accordingly.
- By using a verifier contract on the L1 chain, zk-rollups enable trustless and efficient off-chain processing of transactions, while still maintaining the security and decentralization of the Ethereum network.

Referred blogs

- Vitalik Buterin
- Coindcx
- Chainlink
- Alchemy
- PolygonZkEVM