Haym @SalomonCrypto

Sep 17 • 28 tweets • SalomonCrypto/status/1570984357774655489

(1/27) @ethereum Roadmap: ZK-EVMs

The path from today's Ethereum to the World Computer of the future is through ZK-EVMs. But not all are made the same, and the field is getting crowded...

Fortunately, @VitalikButerin weighed in!

Your guide to ZK-EVMs and VB's ZK-EVM taxonomy.

(2/27) ZK-Proofs are a tool that mathematically prove the validity of a statement without sharing any information about it.

All projects built around ZK-proofs share the same core goal: build scalable technology to make cryptographic proofs of @ethereum-like transactions.

> **Haym**
> @SalomonCrypto · **Follow**
>
> (1/13) Cryptography Basics: Zero-Knowledge Proofs
>
> To keep **@ethereum** decentralized & fair, we keep it slow enough for even the most humble nodes to keep up. But what if they didn't have to keep up with the whole network? What if you could just trust the summaries...
>
> Trustlessly.
>
> 12:24 AM · Sep 12, 2022
>
> Read the full conversation on Twitter
>
> 433 Reply Copy link
>
> **Read 12 replies**

(3/27) ZK-proofs allows one party (prover) to prove to another party (verifier) that a statement is true while also ensuring that the prover does not give the verifier any info that the verifier didn't already have.

All with cryptographic, mathematical certainty.

(4/27) One of the more exciting applications of ZK-proofs are ZK-rollups, enabling layers on top of the World Computer that inherit the benefits @ethereum but are much more scalable.

This technology, with additional mainnet upgrades, will scale Ethereum to 100k+ txns/sec.

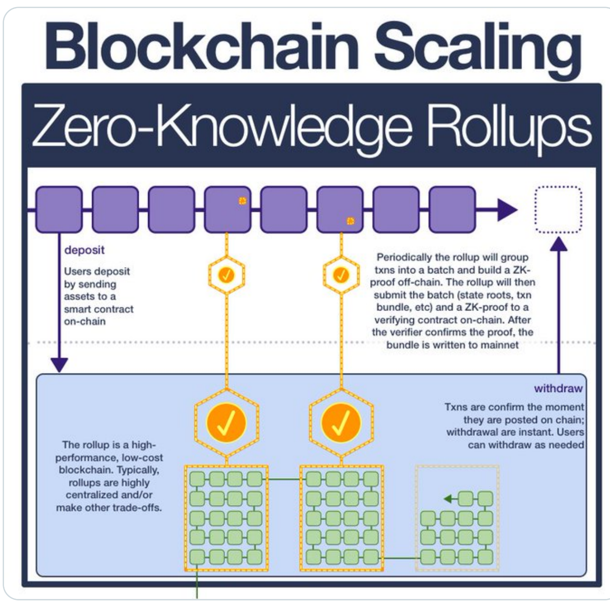
Haym
@SalomonCrypto · **Follow**

(1/23) Blockchain Scaling: Zero-Knowledge Rollups

State Channels → Plasma → Optimistic Rollups → ZK-Rollups

This is how **@ethereum** scales to 100k txn/sec. This is how Ethereum becomes the World Computer.

Your guide to the future of blockchain scaling technology.



2:39 PM · Sep 12, 2022

Read the full conversation on Twitter

404 See the latest COVID-19 information on Twitter

Read 15 replies

(5/27) A ZK-rollup will bundle up all the transactions that occurred on the rollup chain and created a ZK-proof. Although this proof is difficult to generate, it is very easy to verify.

Once generated, the new state, txns and the ZK-proof are posted to a smart contract.

Haym
@SalomonCrypto · **Follow**

Replying to @SalomonCrypto

(12/15) The ZK-proof represents mathematically certainty that whatever is posted on-chain was both valid and actually happened on the rollup. If the proof verifies, the transaction is final both on the rollup and on **@ethereum**.

All the benefits of rollups with instant settlement.

> **Haym** @SalomonCrypto
> Replying to @SalomonCrypto
> (10/13) Verifiable computation is critical to improving processing speeds on blockchains without reducing security.
>
> Instead of processing every txn on-chain, @ethereum can offload execution. After processing, that chain can return the results to mainnet with a ZK-proof.

11:04 PM · Sep 12, 2022

5 Reply Copy link

Read 1 reply

(6/27) ZK-rollups are not the only application of ZK-proofs, but the core functionality is shared across all ZK-proof projects: build cryptographic proofs of the execution of @ethereum(-like) txns.

The first protocols were application specific (eg send payments).

(7/27) But [@ethereum](https://twitter.com/ethereum) is the World Computer; it is capable of so much more than sending payments. Its computational capabilities are defined by the Ethereum Virtual Machine (EVM).

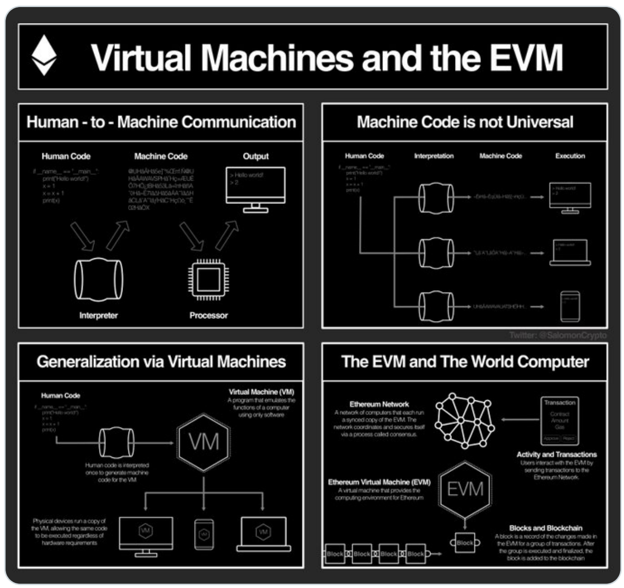
Projects that can process all possible EVM txns are call ZK-EVMs.

Haym
@SalomonCrypto · **Follow**

(1/17) Heard of The World Computer, but not sure how **@ethereum** possibly qualifies?

You need to learn about virtual machines, the EVM and how Ethereum provides a common, trusted and secure computing environment.

After this thread, you'll never think of Ethereum the same again!



7:47 PM · Jul 27, 2022

Read the full conversation on Twitter

664 Reply Copy link

Read 27 replies

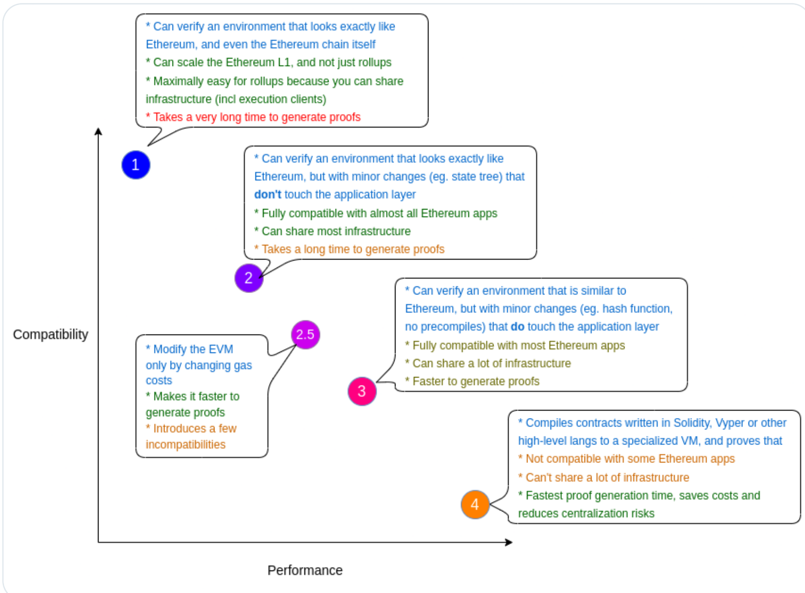
(8/27) Already there are multiple ZK-EVM projects gathering a lot of attention:

- [@0xPolygon](https://twitter.com/0xPolygon) zkEVM
- [@zksync](https://twitter.com/zksync) 2.0
- [@Scroll_ZKP](https://twitter.com/Scroll_ZKP)
- [@nethermindeth](https://twitter.com/nethermindeth)'s complier from Solidity to Cairo ([@StarkWareLtd](https://twitter.com/StarkWareLtd))

It's becoming more important than ever to understand the ZK-EVM landscape.

(9/27) There are 4(ish) categories of ZK-EVM:

1) Fully @ethereum-equivalent
2) Fully EVM-equivalent
2.5) EVM-equivalent, except for gas costs
3) almost EVM-equivalent
4) high-level-language equivalent



(10/27) Type 1: Fully @ethereum-equivalent

Fully and uncompromisingly Ethereum equivalent, regardless of ZK-proof implications, both within the context of the EVM and within the context of Ethereum as a whole.

(11/27) The goal of type 1 ZK-EVMs is to be able prove @ethereum (execution-layer) blocks in full, as they exist on the blockchain.

Ultimately, type 1 ZK-EVMs are required to create trustless scaling of the World Computer. But, at least today, they have drawbacks.

(12/27) @ethereum was not originally designed around ZK-friendliness, so there are many parts of the Ethereum protocol that take a large amount of computation to ZK-prove.

At present, it takes at type 1 ZK-EVM many hours to produce and prove a block.

(13/27) One of the particularly challenging data structures to create a ZK-proof for is a Merkle Tree, a data structure @ethereum used to store information needed for Ethereum nodes to run the EVM.

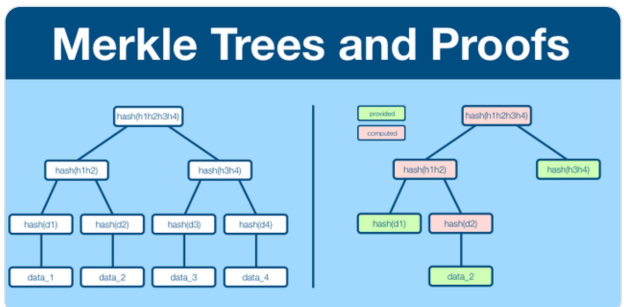
From within the EVM, applications don't see (most) Merkle Trees.

Haym
@SalomonCrypto · **Follow**

(1/13) Computer Science 201: Merkle Trees and Merkle Proofs

If you want to understand **@Bitcoin**, **@ethereum** and blockchain technology, you need to learn:

- How a Merkle trees expresses a large dataset
- How a Merkle proof works
- Why a Markle tree is so efficient



10:17 PM · Sep 7, 2022

Read the full conversation on Twitter

415 Reply Copy link

Read 15 replies

(14/27) Type 2 - fully EVM-equivalent

Whereas type 1 ZK-EVMs aim to be equivalent within the entire context of @ethereum, type 2 ZK-EVMs aim to be equivalent only within the context of the EVM.

(15/27) The goal of type 2 ZK-EVMs is to be fully compatible with existing World Computer applications, but they make modifications in block structure, state (Merkle) tree and other data structures.

(16/27) All modifications are done in areas outside the awareness of applications, and so any application that works on @ethereum will work in a type-2 ZK-EVMs as well.

Most developer tooling will also work out of the box.

(17/27) Unfortunately, while type-2 ZK-EVMs make major improvements over type-1s, they are still slow.

The ZK-unfriendly design choices were not just in how the network operates the EVM, its within the EVM itself. Type-2s run up against the same unoptimized EVM that Type-1s had.

(18/27) Type 2.5 - EVM-equivalent, except for gas costs

Type 2.5 ZK-EVMs alter the gas costs of @ethereum txns in order to disincentive the use of operations that are very difficult to ZK-prove.

(19/27) Type 2.5s represent a significant optimization without major changes to the EVM. This can have implications on developer tooling and break some applications on the margin, but they are minimal and manageable.

Nevertheless, type 2.5s are a clear step away from the EVM.

(20/27) Type 3 - almost EVM-equivalent

Type 2 ZK-EVMs make changes to the data structures outside the EVM, type 3 ZK-EVMs take it a step further and begin making changes within the ZK-EVM itself.

(21/27) The main change is (usually) removing special functions called precompiles, but type 3s can also make changes to how code works.

While type 3 ZK-EVMs gain even further performance improvements, they begin to make big enough changes that they might break certain projects.

(22/27) Type 4 - high-level-language equivalent

Type 4 ZK-EVMs take regular EVM code and transforms (compiles) it to a different language that is much easier to ZK-prove.

(23/27) There are huge benefits to this approach; the ZK tech can specialize and optimize, delivering incredible performance and a great platform for devs.

However, the point of a ZK-EVM is to process @ethereum txns without a lot of work, and type 4s require a lot of work.

(24/27) First and foremost, type 4 ZK-EVMs requires the development and maintenance of a whole new piece of software: the compiler.

Furthermore, type 4s begin to require real changes to smart contract code and break a lot of developer tooling.

Disadvantage: more incompatibility

A "normal" application written in Vyper or Solidity can be compiled down and it would "just work", but there are some important ways in which very many applications are not "normal":

- **Contracts may not have the same addresses** in a Type 4 system as they do in the EVM, because CREATE2 contract addresses depend on the exact bytecode. This breaks applications that rely on not-yet-deployed "counterfactual contracts", ERC-4337 wallets, EIP-2470 singletons and many other applications.
- **Handwritten EVM bytecode** is more difficult to use. Many applications use handwritten EVM bytecode in some parts for efficiency. Type 4 systems may not support it, though there are ways to implement limited EVM bytecode support to satisfy these use cases without going through the effort of becoming a full-on Type 3 ZK-EVM.
- **Lots of debugging infrastructure cannot be carried over**, because such infrastructure runs over the EVM bytecode. That said, this disadvantage is mitigated by the *greater* access to debugging infrastructure from "traditional" high-level or intermediate languages (eg. LLVM).

(25/27) Types are not better or worse than each other; they simply have different trade-offs. Furthermore, a ZK-EVM can move up or down in classification over time - it just depends what features they add and how @ethereum mainnet evolves over time.

(26/27) Projects:

Type 1)
Privacy and Scaling Explorations team

privacy-scaling-explorations/**zkevm-...**

41 Contributors | 74 Issues | 334 Stars | 153 Forks

GitHub - privacy-scaling-explorations/zkevm-circuits
Contribute to privacy-scaling-explorations/zkevm-circuits development by creating an account on GitHub.
https://github.com/privacy-scaling-explorations/zkevm-circuits

Type 2)
none... yet

Type 3)
[@0xPolygon](), [@Scroll_ZKP]() (both working towards type 2)

Type 4) @zksync, @StarkWareLtd + @nethermindeth

(27/27) And there you go! The taxonomy of ZK-EVM

Want to hear it from the source? Check out Vitalik's post:

The different types of ZK-EVMs
https://vitalik.ca/general/2022/08/04/zkevm.html

Like what you read? Help me spread the word by retweeting the thread (linked below).

Follow me for more explainers and as much alpha as I can possibly serve.

Haym
@SalomonCrypto · **Follow**

(1/27) **@ethereum** Roadmap: ZK-EVMs

The path from today's Ethereum to the World Computer of the future is through ZK-EVMs. But not all are made the same, and the field is getting crowded...

Fortunately, **@VitalikButerin** weighed in!

Your guide to ZK-EVMs and VB's ZK-EVM taxonomy.

3:53 AM · Sep 17, 2022

Read the full conversation on Twitter

5 Reply Copy link

Read 1 reply

• • •