Haym @SalomonCrypto

Sep 12 • 14 tweets • SalomonCrypto/status/1569119846952235008

(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.

(2/13) In America, citizens don't have ID numbers... and yet we live in a society that necessitates them. If you've ever applied for a loan, tried to rent a house or opened a bank account you already know: we've just decided to use our social security numbers instead.

(3/13) Originally intended to be specific to a single government benefits program, the social security number (SSN) have become an American's unique identifier.

Almost like private keys for meat-space.

And yet, we're required to give it to basically anyone who asks.

(4/13) Imagine a world where you could prove, with 100% certainty, that you were who you said you were (corresponding to your SSN) without revealing ANY information about your SSN.

The world would look entirely the same, expect with >90% less identity theft.

(5/13) A Zero-Knowledge Proof (ZK-proof) is a way of proving a statement is valid without revealing it.

A ZK-proof relies on verifiable algorithms that take input data and return "true" or "false" without sharing the statement's contents or how you discovered the truth.

(6/13) A ZK-proof must be:

- complete (all valid inputs return true)
- sound (all invalid inputs return false)
- zero-knowledge (the party requesting verification learns nothing about the statement that they didn't already know)

(7/13) ZK-proofs are a category of mathematical tools with applications far beyond cryptography and cryptocurrency, but the technology has important implications on-chain, particularly for privacy and scalability.

(8/13) Privacy

(Most) blockchain computers are built on top of public, decentralized ledgers. By their very design, every txn is visible for all to see... forever

ZK-proofs allow users to securely interact on-chain while obfuscate txn details and guaranteeing financial privacy

(9/13) Scalability

ZK-proofs introduce a new paradigm: verifiable computation. Verifiable computing allows us to send computation to another entity while maintaining verifiable results.

(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.

(11/13) ZK-technology is still very young. Generating proofs is extremely challenging and computationally expensive. Verifying proofs is also costly (although significantly computationally cheaper) and must be done on-chain, where costs are high.

(12/13) Current implementations have a few important weakness:

- Some ZK-proofs require a reference string that must be generated from trusted parties
- Some ZK-proofs are vulnerable to quantum computers

Fortunately, development is moving quick and accelerating.

(13/13) In summary, 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.

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.

• • •