Blockstream RESEARCH

Blockstream

Why There's No ZK in Bitcoin

The Missing Pieces

Jonas Nick
@n1ckler, jonas@n-ck.net

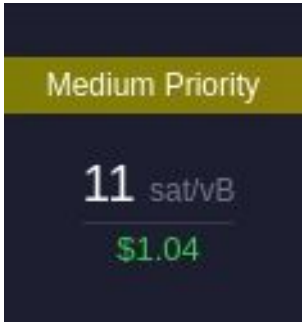
2024-05-23

ZKProof 6

Bitcoin is a Social System

> I've been working on a new electronic cash system that's fully

> peer-to-peer, with no trusted third party.



mempool.space

> Participants can be anonymous.

Founders And CEO Of Cryptocurrency Mixing Service Arrested And Charged With Money Laundering And Unlicensed Money Transmitting Offenses

Wednesday, April 24, 2024

Share

Keonne Rodriguez and William Lonergan Hill Are
Unlicensed Money Transmitting Business That E
Laundered Over $100 Million in Criminal Proceed

MAY 2, 2024 | 2 MINUTES READ

ZKSNACKS IS DISCONTINUING ITS COINJOIN COORDINATION SERVICE 1ST OF JUNE

After years of relentless dedication to improve Bitcoin's privacy, zkSNACKs, the company pioneering the development of Wasabi Wallet, is shutting down its coinjoin coordination service, effective from June 1st, 2024.

1. Today’s ecosystem does not fully realize Bitcoin’s ideals
2. Technology is evolving

➤ How should the Bitcoin protocol evolve?

What can ZK and SNARKs in Bitcoin do?

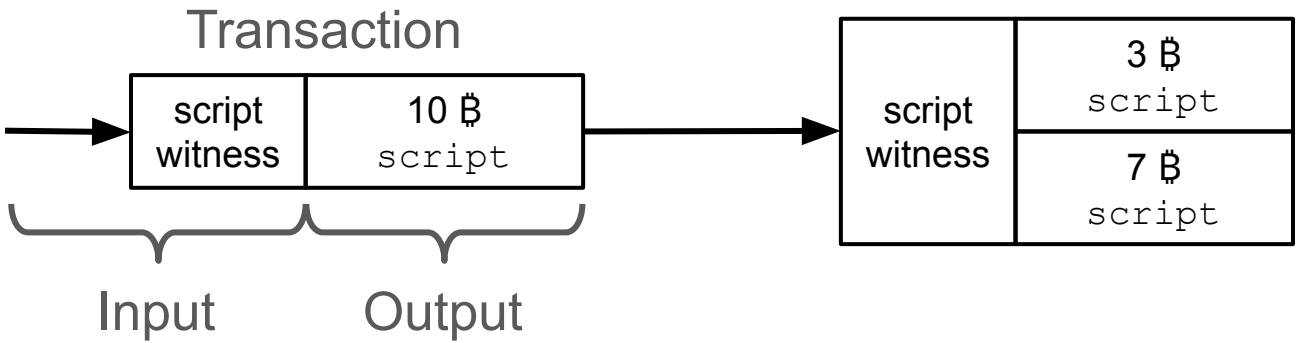
- Improve **privacy**
  - Homomorphic commitment to values (elements, 2013)
  - Obfuscate transactions via linkable ring signatures (monero, 2013)
  - Private transactions (halo2, 2020)
- Improve transaction **throughput**
  - Bridge to alternative systems (coinwitness, 2014)

Why is it not implemented yet?

Because Bitcoin is difficult to change, and that’s good.

What can we do without changing Bitcoin?

Bitcoin Transaction Basics



- Transaction valid only if it provides a witness that makes the script succeed.
- Bitcoin script is a sequence of **opcodes** that manipulate a stack.
- Example script: `` `<some_pubkey> OP_CHECKSIG` `` fails if the topmost stack element is not a valid signature for the pubkey.

Bitcoin Script

- What it **can** do:
  - Rearrange the **stack**, check for equality, branch on stack values
  - Limited arithmetic on 32-bit numbers: **add and subtract**
  - **Hash** and check **ECDSA/Schnorr** signatures
- What it **cannot** do:
  - No **loops**, gotos, recursion
  - No **bitwise** operations
  - No arithmetic opcodes to do **multiplication** or division
  - No **concatenation** of elements on the stack
  - No way to **introspect** transactions, or to carry over state from output to output
- Some capabilities were disabled by Satoshi

Can you verify a SNARK in Bitcoin script?

- **Yes**, can verify any computation in Bitcoin script…
  - despite not being “Turing-complete”.
- …but **no**, verifier would result in **much** larger script than the 4MB we can put in a block
  - 254 bit integer multiplication is 77kB
  - Checking merkle proofs very, very expensive without concatenation
  - No abstraction in script, need to unroll
- Why not simply add **more powerful opcodes** to the Bitcoin protocol?

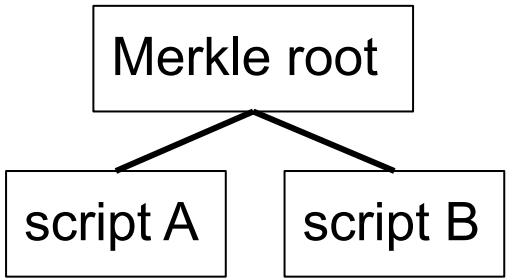
Changing Bitcoin is hard

- All participants need to follow **exactly** the same protocol rules
  - No central decision maker
  - Any change has tradeoffs and has opposition
  - No way to determine rough consensus
- Any update **risks chain split**
- Most recent updates: SegWit (2017), Taproot (2021)

Taproot 🥕

- Why **successful**?
  - Low-hanging fruits: obvious improvements, cleanups & bugfixes
  - No change in existing security assumptions
- Relevant changes
  - Removed script limits
  - Added upgrade mechanisms
  - Reduces script that needs to be revealed

For `script A OR script B`, put only Merkle root in output.



At spending time reveal either script A or script B and Merkle proof.

OP_SNARK

- Add dedicated opcode that verifies SNARK?
- Contrast to Taproot:
  - **Huge** design space
    - Will never be able to remove the opcode from the protocol
  - Very complex
    - Taproot only about ~1600 loc (without tests)

Make Script More Powerful?

- Add **simple** opcodes that allows to write SNARK verifiers in script
- Downside: **script is difficult** to write and reason about

Alternative: add language that is designed for blockchains

- Simplicity is deliberately not Turing-complete, was built for easy static analysis and formal verification.
- But would be relatively large change to Bitcoin

Re-enable OP_CAT? 😸

- Existed in original Bitcoin, but **disabled** by Satoshi
- `[ 0xBE, 0xEF ] -> [ 0xBEEF ]`
- Surprisingly powerful
  - Transaction introspection and state
  - Allows Merkle proof verification
  - … and more
- Does this help to verify SNARKs in script? **Yes, somewhat**
- Status unclear
- But potential path for more simple opcodes (“Great Script Restoration”)

Hardware Requirements

- Users must be able to verify Bitcoin's rules; don't make that more expensive!
- Hard limits
  - 4 MB block per 10 minutes
  - At most 80,000 signature verifications per block
- On average
  - 7k to 10k signature verifications per block
  - 90th percentile block verification time on my node: 3.2 seconds

Hardware Requirements

- Hardware “wallets” for long-term storage and signing
- Example: ESP32 SoC with 240 MHz dual core and 520KiB memory

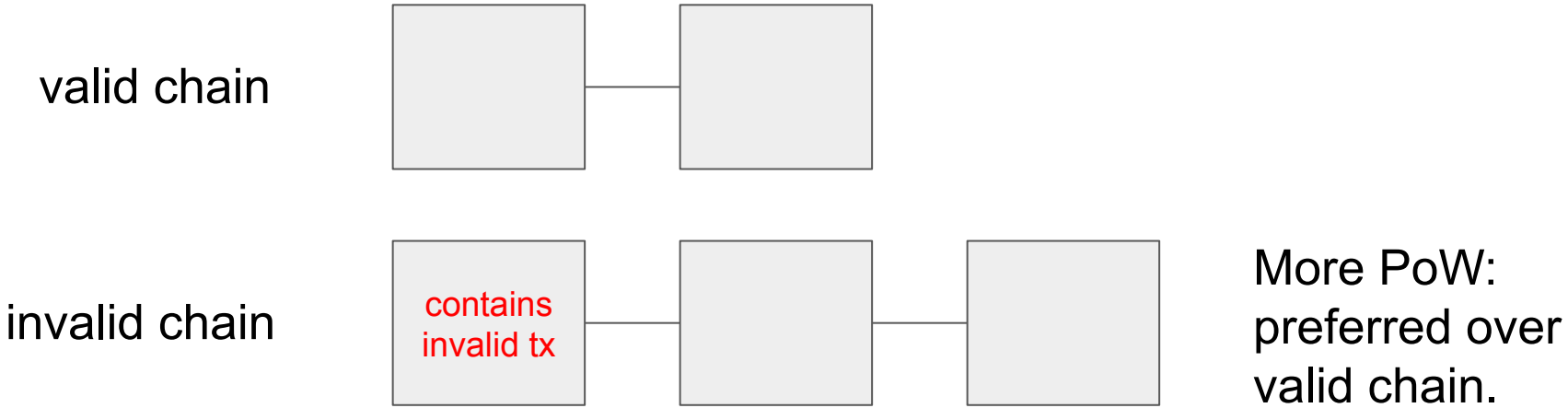
What is the maximum acceptable time a user can wait until the signing device (HWW, mobile phone, ...) produced the proof?

There are plenty of applications of SNARKs that do not require a protocol change

Chain State Proofs

- Proof that a given byte array is the block hash of a valid blockchain
- Applications
  - Reduce resources required to verify the blockchain
  - Bridging with BitVM
- ZeroSync team generated proof for weaker “header chain” statement
  - Only the headers of blocks are proven to be correct, not the transactions
  - ~840,000 block headers of 80 bytes, double SHA256 each header
  - Using STARKs, this cost about $4000 to prove (2023) and takes ~3 seconds to verify in my browser.

Header chain proof requires assumption that blockchain with the most proof-of-work is valid.



Expensive to create, but maybe worth to the attacker.

Full Chain State Proof

- Requires proving all of Bitcoin's protocol rules:
  - ECDSA & Schnorrsigs over secp256k1
    - 30M per month, 2.5B in total
  - SHA256
    - > 7GB per month, > 650 GB in total
    - more likely 2-3 times that
  - **Bug-for-bug** compatibility with the C++ "specification" (see libbitcoinkernel)

BitVM

- In theory: makes verifying any computation possible today
- Overcomes maximum Script size limitation using
  - **Taproot Merkle tree** of scripts
  - **key-value store** that can be accessed across individual scripts
- KV store requires **fraud proofs**
  - If prover posts invalid claim or uses inconsistent store, someone can make a transaction to take the prover’s funds.

Summary

- Bitcoin faces challenges, but changing Bitcoin remains hard (and that is good)
- SNARKs have potential to address Bitcoin's problems
- Most plausible ways to get SNARKs in Bitcoin, short to midterm (imho):
  1. BitVM (needs R&D)
  2. Re-enable opcodes (needs rough consensus)
- Either way, there are numerous exciting opportunities to make this technology practical and develop innovative and useful applications!

Slides at nickler.ninja/slides

Thanks to Andrew Poelstra, @LiamEagen, @Robin_Linus, @0xB10C

Jonas Nick
@n1ckler, jonas@n-ck.net