BitVM

Off-chain Bitcoin Contracts

Motivation

- Blockchains don't scale
- Lightning doesn't quite scale
- Can we scale Bitcoin to billions of users?

Envisioning an Ideal World

- 100 - 1000x higher throughput
- Free market of L2s (zk-rollups, sidechains, zkCoins, ...)
- Rapid innovation / cheap experimentation
- All L2s interconnected via LN
- Possible, but we need bridges!

Overview

- Stateful Bitcoin Scripts
- BitVM architecture
- BitVM bridges

Stateful Bitcoin Scripts

Bitcoin Script

- Satoshi disabled all the fun opcodes
- Reduced to minimum
- But enough interesting opcodes are left
- Scripts can be up to 4mb
- Bitcoin Script code golf

Stateful Bitcoin Scripts

- Idea: If we could sign a value...
- Enforcing the same value for x in script1 and script2
- Punish equivocation
- How to sign a value though?
- We don't have CSFS...

Lamport Signatures

- Conceptually very simple
- Require only hash functions
- Possible in Bitcoin Script
- Main drawback: large
- But one can sign a u8, u32, u160

Lamport Signature for a 1-bit Message

```
OP_HASH160
OP_DUP

<0xf592e757267b7f307324f1e78b34472f8b6f46f3>    // This is hash1
OP_EQUAL
OP_DUP

OP_ROT
<0x100b9f19ebd537fdc371fa1367d7ccc802dc2524>    // This is hash0
OP_EQUAL

OP_BOOLOR
OP_VERIFY

// Now the value of the bit commitment is on the stack. Either "0" or "1".
```

BitVM Architecture

BitVM the paradigm

- Tries to keep things off-chain (like LN)
- 2-party setting: prover and verifier
- Optimistic computation
- Disprove a faulty result (much easier than execution)
- Tree++

Tree++

- Language to express Bitcoin Contracts in graphs of transactions
- Templating language for Script
  - Evaluate constant expressions
  - Unroll loops
  - Compose functions
- Statefulness via Lamport signatures (u8, u32, u160, ...)
- Composite opcodes (xor, shift, mul, blake3, ...)
- Connector outputs
- Potentially large scripts, large Taptrees, and large TX graphs

BitVM the Bitcoin VM

- Don't want to hand-craft and hand-optimize a low-level circuit for every application
- Build some generic VM
- Succinctly disprove any faulty result
- Ideally: RISC-V architecture

BitVM Specs

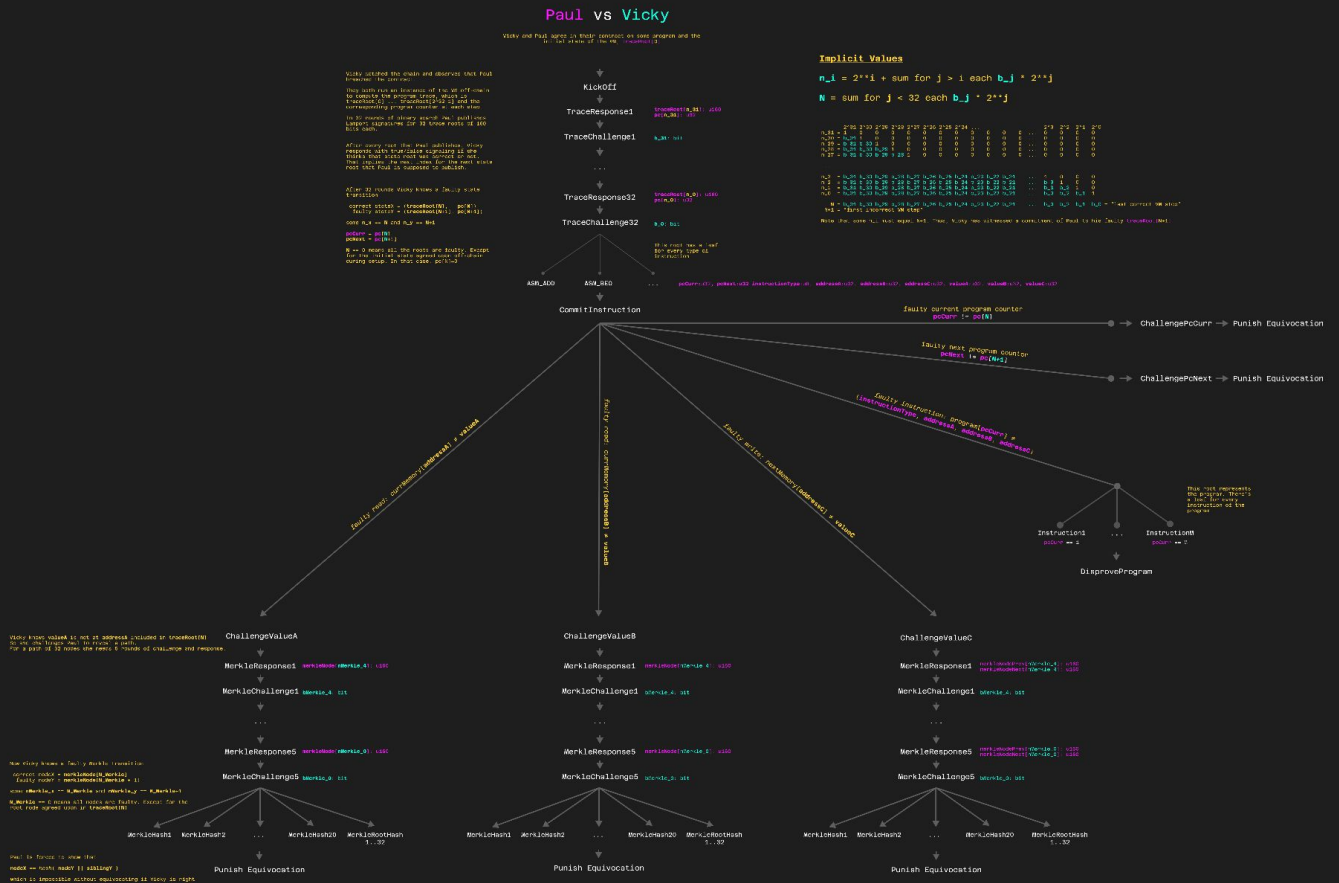
- Basic instruction set rv32i
- Compile target for clang, gcc, LLVM
- Use existing C / C++ / Rust / ... libraries
- STARK & SNARK verifiers, etc...

BitVM Detailed Specs

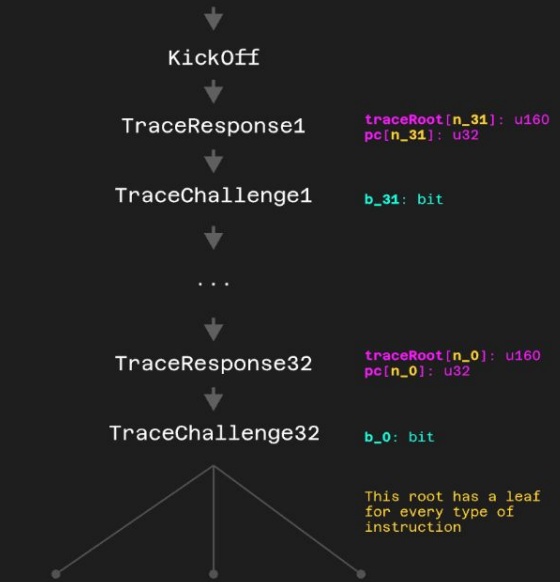
- Max steps: 2^32
- Memory: 2^32 * 4 bytes ~ 17GB
- Worst case: ~40 rounds of challenge & response
- "Court case" runs for up to half a year
- In total: ~150kb of Scripts
- Worst case is heavily disincentivized
- Likely never happens in real world applications

BitVM Transaction Graph

github.com/bitvm/bitvm/docs



Vicky and Paul agree in their contract on some program and the initial state of the VM, traceRoot[0]



Implicit Values

n_i = 2**i + sum for **j** > i each **b_j** * 2**j

N = sum for **j** < 32 each **b_j** * 2**j

	2^31	2^30	2^29	2^28	2^27	2^26	2^25	2^24	...				2^3	2^2	2^1	2^0
n_31 =	1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
n_30 =	b_31	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0
n_29 =	b_31	b_30	1	0	0	0	0	0	0	0	0	...	0	0	0	0
n_28 =	b_31	b_30	b_29	1	0	0	0	0	0	0	0	...	0	0	0	0
n_27 =	b_31	b_30	b_29	b_28	1	0	0	0	0	0	0	...	0	0	0	0
...																
n_3 =	b_31	b_30	b_29	b_28	b_27	b_26	b_25	b_24	b_23	b_22	b_21	...	1	0	0	0
n_2 =	b_31	b_30	b_29	b_28	b_27	b_26	b_25	b_24	b_23	b_22	b_21	...	b_3	1	0	0
n_1 =	b_31	b_30	b_29	b_28	b_27	b_26	b_25	b_24	b_23	b_22	b_21	...	b_3	b_2	1	0
n_0 =	b_31	b_30	b_29	b_28	b_27	b_26	b_25	b_24	b_23	b_22	b_21	...	b_3	b_2	b_1	1
N =	b_31	b_30	b_29	b_28	b_27	b_26	b_25	b_24	b_23	b_22	b_21	...	b_3	b_2	b_1	b_0 = "last correct VM step"
N+1 =	"first incorrect VM step"															

Note that some n_i must equal N+1. Thus, Vicky has witnessed a commitment of Paul to his faulty traceRoot[N+1]

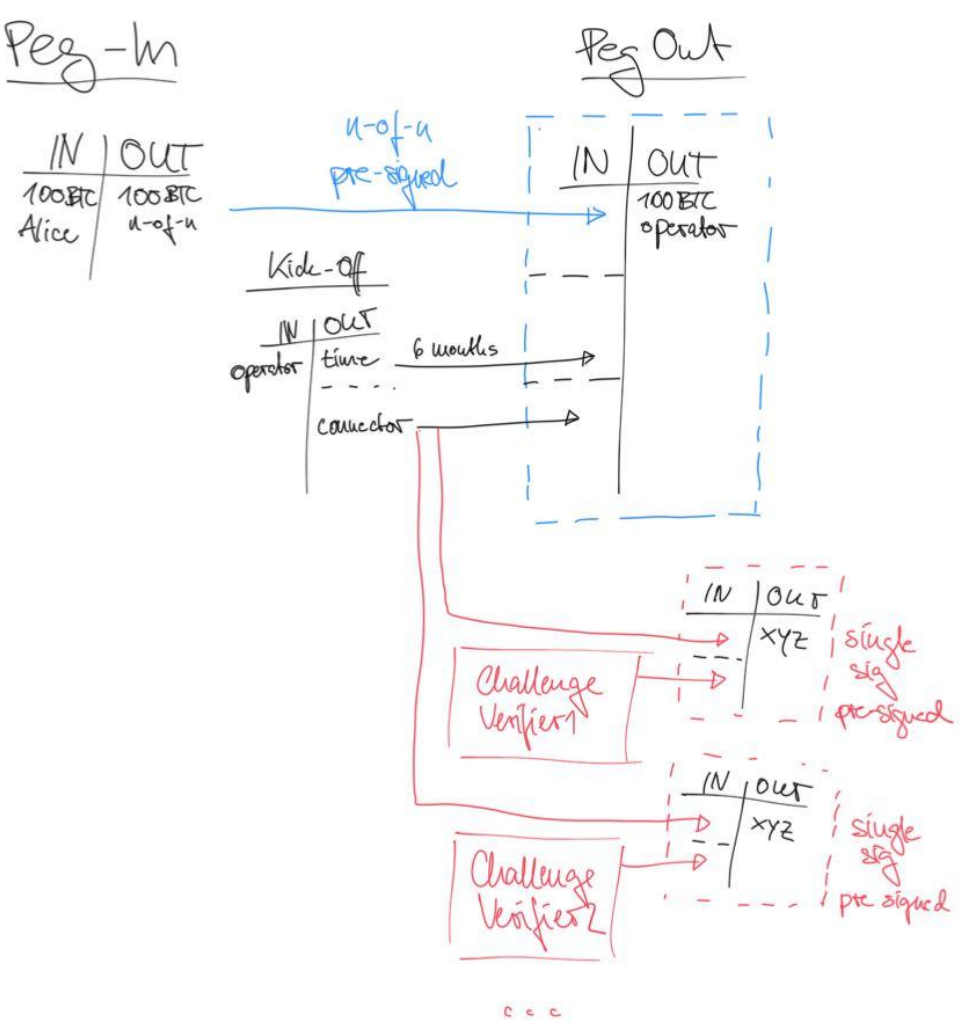
BitVM Bridges

BitVM Bridges

- Bridge BTC to any other system
- Idea: a bit clunky is fine
- Bridge is used rarely. Only large amounts
- End users use cross-chain swaps

BitVM Bridge Guarantees

- A Federation, but a single honest member suffices
- Guarantees that the bridge is safe and live
  - safe: nobody can steal the deposit
  - live: you can't stop a valid peg-out
- Large federations: +100 members
- *You* can be a member. Then you don't have to trust anyone


Peg-In
IN
OUT
100 BTC
Alice
100 BTC
n-of-n
Peg Out
n-of-n
pre-signed
IN
OUT
100 BTC
operator
Kick-Off
IN
OUT
operator
time
6 months
connector
IN
OUT
XYZ
single sig pre-signed
Challenge Verifier1
IN
OUT
XYZ
single sig pre-signed
Challenge Verifier2

Limitations

- Complexity
- Balancing incentives: Loser has to pay winner's fees + bounty
- If incentives are balanced the chain is not needed
- Potentially capital intensive
- But no 1:1 collateral required
- For every peg-in all N parties have to pre-sign N peg-out TXs
- Federation can censor peg-ins

Summary & Outlook

- BitVM enables more complex Bitcoin contracts
- Use case: trust-minimized bridges for rollups, sidechains, L2s, ...
- Limitation: practical but clunky
- Requires no softfork
- Toy version ready this month
- Reckless mainnet this year

Questions?