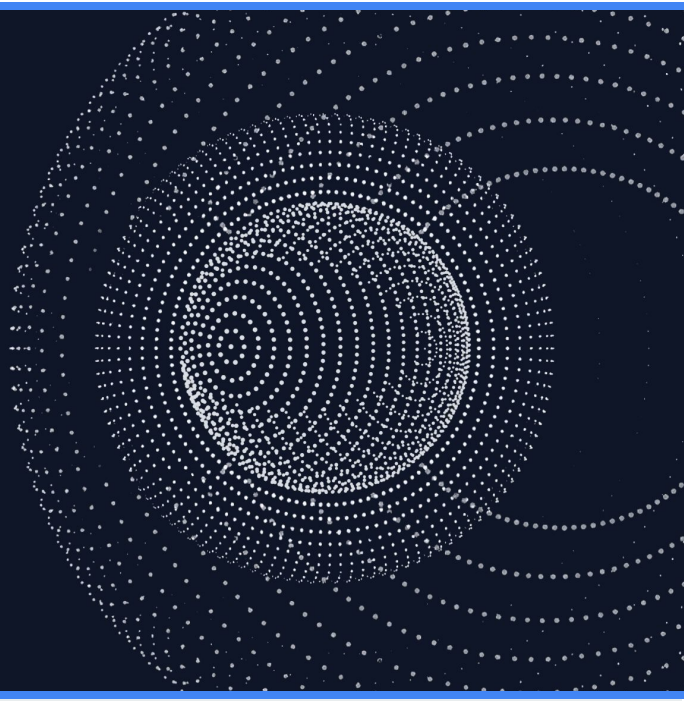
Fairgate

# BitVMX and Fairgate's Evolving Ecosystem of Protocols



**Sergio Demian Lerner** | CTO, Fairgate Labs

NOV 2025

# Why should a Blockchain Compute?

- Smart Wallets
- DEXes
- DAOs
- Computation Outsourcing
- Fair games of private strategy supporting gambling
- Trading Responsible Vulnerabilities Disclosures with ZK-Proofs
- Improved Lightning Network Channels
- More decentralized bridges to sidechains and rollups
- Verify rollup state transitions
- Enable new innovative L2s

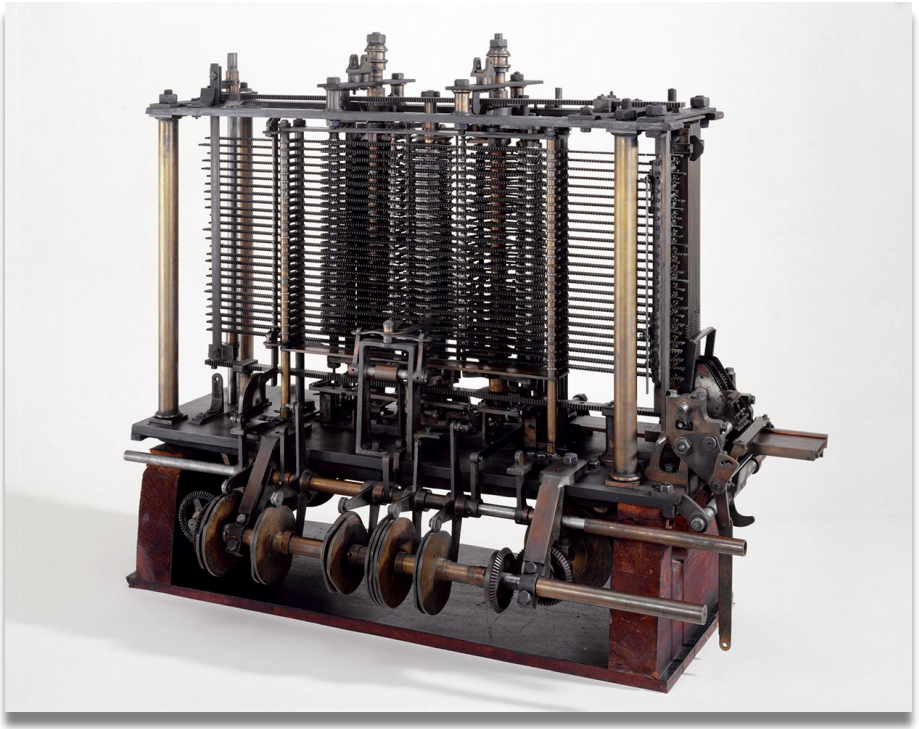
# Functional Escape Velocity

**Vitalik's Definition:**
State of a blockchain in which its computational capacity is sufficient to support all necessary applications without requiring significant architectural revisions or major changes.

**Definition for Bitcoin:**
Minimum amount of computation a blockchain must support to allow its native token to be moved in a decentralized manner to L2s that support all applications.

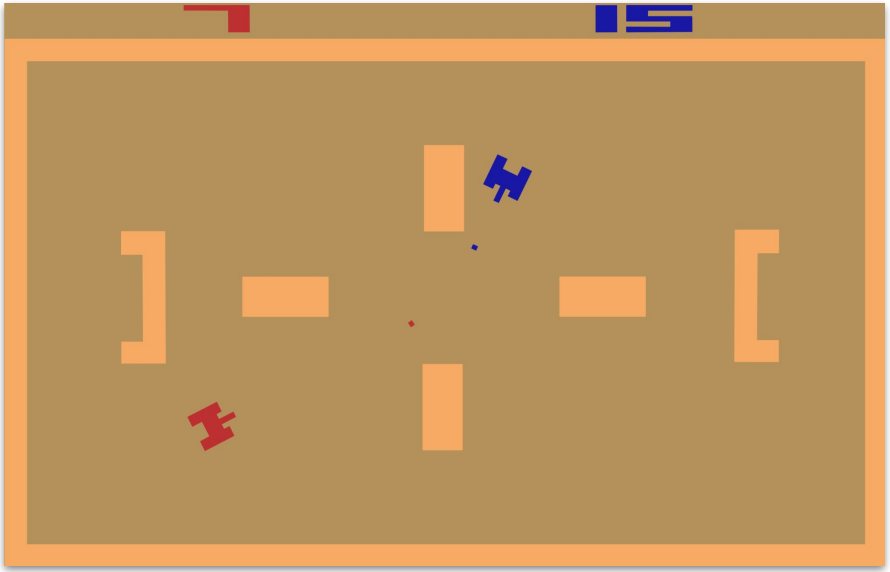
# What can we compute on Bitcoin?

# What Is This Machine?



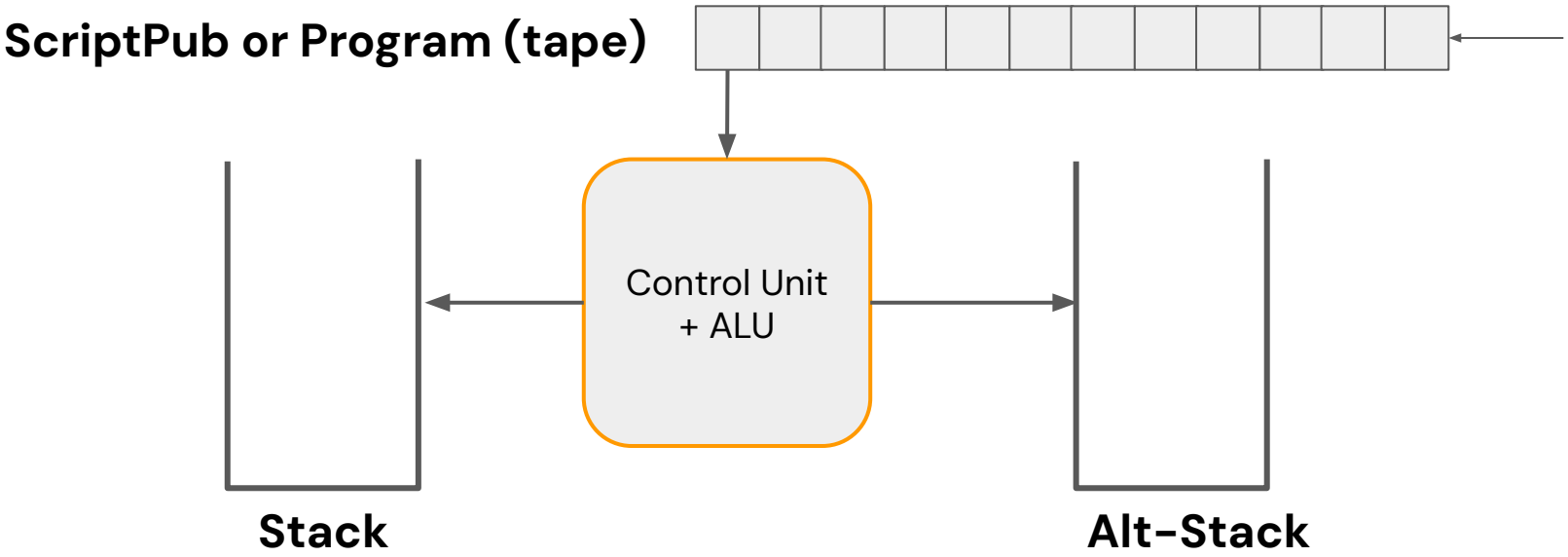
**RAM: 16 Kb  Ops: +-**

# And Who Remembers This ?




**Processor: 6507** **RAM: 128 bytes** **ROM: up to 16 Kb** **Ops: +-x/**

# Bitcoin Script as a Computer (2009)




**0 RAM  Stack: 5 Mb  Tape: 1 Mb  Ops: + - x - CHECKSIG**

# The temporary change to script.cpp on August 15th, 2010

misc changes

```
94                 if (opcode == OP_CAT ||
95                     opcode == OP_SUBSTR ||
96                     opcode == OP_LEFT ||
97                     opcode == OP_RIGHT ||
98                     opcode == OP_INVERT ||
99                     opcode == OP_AND ||
100                    opcode == OP_OR ||
101                    opcode == OP_XOR ||
102                    opcode == OP_2MUL ||
103                    opcode == OP_2DIV ||
104                    opcode == OP_MUL ||
105                    opcode == OP_DIV ||
106                    opcode == OP_MOD ||
107                    opcode == OP_LSHIFT ||
108                    opcode == OP_RSHIFT)
109                    return false;
110
```

# OPCODES

## Constants

| |
|---|
| OP_0, OP_FALSE |
| N/A |
| OP_PUSHDATA1 |
| OP_PUSHDATA2 |
| OP_PUSHDATA4 |
| OP_1NEGATE |
| OP_1, OP_TRUE |
| OP_2-OP_16 |

## Bitwise logic

| |
|---|
| OP_INVERT |
| OP_AND |
| OP_OR |
| OP_XOR |
| OP_EQUAL |
| OP_EQUALVERIFY |

## Cryptographic

| |
|---|
| OP_RIPEMD160 |
| OP_SHA1 |
| OP_SHA256 |
| OP_HASH160 |
| OP_HASH256 |
| OP_CODESEPARATOR |
| OP_CHECKSIG |
| OP_CHECKSIGVERIFY |
| OP_CHECKMULTISIG |
| OP_CHECKMULTISIGVERIFY |
| OP_CHECKSIGADD |

## Flow control

| |
|---|
| OP_NOP |
| OP_IF |
| OP_NOTIF |
| OP_ELSE |
| OP_ENDIF |
| OP_VERIFY |
| OP_RETURN |

## Arithmetic 2

| |
|---|
| OP_BOOLAND |
| OP_BOOLOR |
| OP_NUMEQUAL |
| OP_NUMEQUALVERIFY |
| OP_NUMNOTEQUAL |
| OP_LESSTHAN |
| OP_GREATERTHAN |
| OP_LESSTHANOREQUAL |
| OP_GREATERTHANOREQUAL |
| OP_MIN |
| OP_MAX |
| OP_WITHIN |

## Splice

| |
|---|
| OP_CAT |
| OP_SUBSTR |
| OP_LEFT |
| OP_RIGHT |
| OP_SIZE |

## Locktime

| |
|---|
| OP_CHECKLOCKTIMEVERIFY (previously OP_NOP2) |
| OP_CHECKSEQUENCEVERIFY (previously OP_NOP3) |

## Arithmetic

| |
|---|
| OP_1ADD |
| OP_1SUB |
| OP_2MUL |
| OP_2DIV |
| OP_NEGATE |
| OP_ABS |
| OP_NOT |
| OP_0NOTEQUAL |
| OP_ADD |
| OP_SUB |
| OP_MUL |
| OP_DIV |
| OP_MOD |
| OP_LSHIFT |
| OP_RSHIFT |

## Stack

| |
|---|
| OP_TOALTSTACK |
| OP_FROMALTSTACK |
| OP_IFDUP |
| OP_DEPTH |
| OP_DROP |
| OP_DUP |
| OP_NIP |
| OP_OVER |
| OP_PICK |
| OP_ROLL |
| OP_ROT |
| OP_SWAP |
| OP_TUCK |
| OP_2DROP |
| OP_2DUP |
| OP_3DUP |
| OP_2OVER |
| OP_2ROT |
| OP_2SWAP |

Removed August, 2010

# Emulation OP_MUL with OP_ADD

## 32-Bit Multiplication in 3575 bytes of Bitcoin Script

# Small Script

- OP_IFDUP
- OP_DEPTH

- OP_0, OP_FALSE
- OP_PUSHDATA1
- OP_1NEGATE
- OP_1, OP_TRUE
- OP_2-OP_16

- OP_EQUAL
- OP_EQUALVERIFY

- OP_IF
- OP_NOTIF
- OP_VERIFY

- OP_CHECKLOCKTIMEVERIFY (previously OP_NOP2)
- OP_CHECKSEQUENCEVERIFY (previously OP_NOP3)

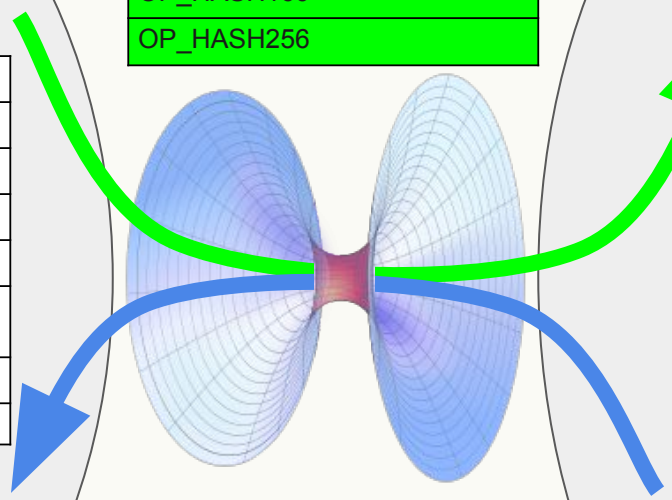
- OP_BOOLAND
- OP_BOOLOR
- OP_NUMEQUAL
- OP_NUMEQUALVERIFY
- OP_NUMNOTEQUAL
- OP_LESSTHAN
- OP_GREATERTHAN
- OP_LESSTHANOREQUAL
- OP_GREATERTHANOREQUAL
- OP_MIN
- OP_MAX
- OP_WITHIN

- OP_1ADD
- OP_1SUB
- OP_NEGATE
- OP_ABS
- OP_NOT
- OP_0NOTEQUAL
- OP_ADD
- OP_SUB

- OP_CAT

Hashing

- OP_RIPEMD160
- OP_SHA1
- OP_SHA256
- OP_HASH160
- OP_HASH256



**OP_SIZE**

- OP_CHECKSIG
- OP_CHECKMULTISIG

- OP_SUBSTR
- OP_LEFT
- OP_RIGHT

# Big Script

- OP_PUSHDATA2
- OP_PUSHDATA4

- OP_CODESEPARATOR
- OP_CHECKSIGVERIFY
- OP_CHECKMULTISIGVERIFY

# Bitcoin vs Babbage vs Atari 2600

| Computer | Working memory (e.g. RAM) for arithmetic operations | Program Memory (e.g. ROM) | Supported Arithmetic operations |
|---|---|---|---|
| Babbage analytical engine (1833) | **~16 Kbytes** | - | + - |
| Atari 2600 (1977) | 128 bytes | **16 Kbytes** | + - x / |
| Bitcoin 0.1 (2009) | infinite | 33 Mbytes | + - x / |
| Bitcoin Script Pre-August-2010 | 20K bytes stack<br>(4x5K or 1000 x 20) | 33 Mbytes | + - x / |
| B.script Post-Aug-2010 Pre-segwit | 4K bytes stack<br>(1000 x 4) | 100 Kbytes | + - |
| Bitcoin script now (emulated mul/div, 9.4x expansion in SNARK verifier) | 4K bytes stack<br>(less than B.a.e) | 10 Kbytes<br>(less than Atari) | - x / |

# Computing on a Blockchain

- **Offchain Trusted Computation**:
  - TEE, Federated, Hyperledger Avalon
- **Onchain Non-TC Space-bounded computation**
  - Bitcoin, Litecoin
- **Onchain Metered Computation**
  - Rootstock y Ethereum.
  - All work onchain
- **Proof of Computation Integrity**
  - SNARKs, STARKs, ZK-Rollups.
  - Most work off-chain, some work on-chain.
- **Disputable Computation**
  - TrueBit, Optimistic Rollups, **BitVM.**
  - Even less work onchain.

# BitVM

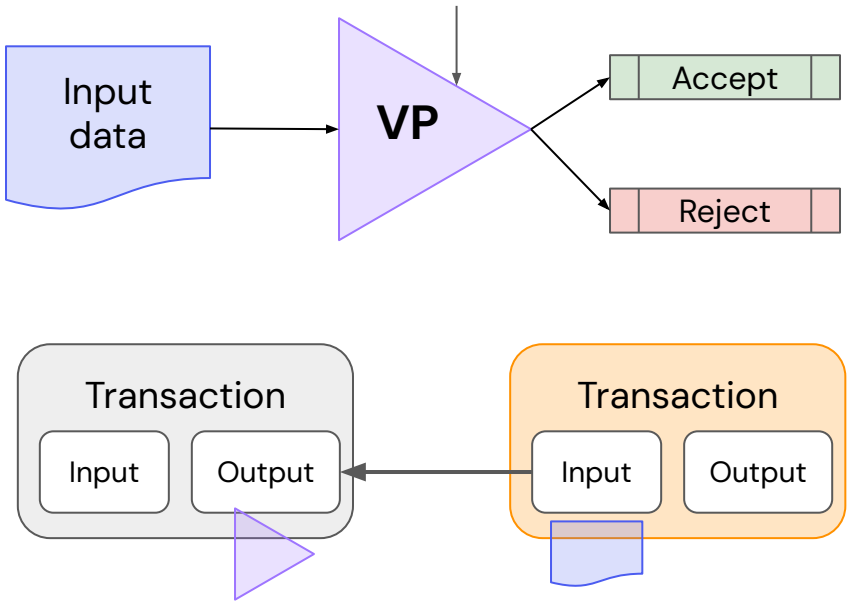
Fairgate

# Computation on Bitcoin

Patrick

**Prover**

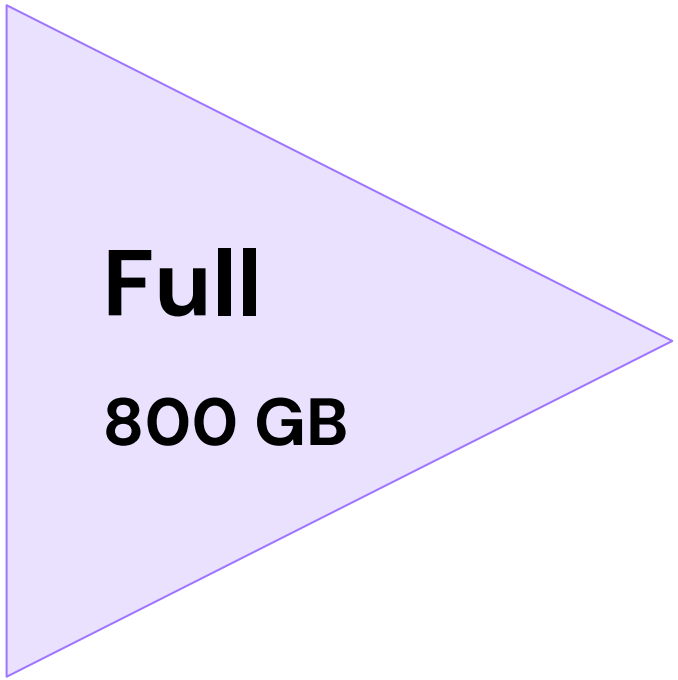
Agreed **Verification Program (VP)**




Valerie

**Observer**

# Script Size of Sidechain/Rollup Peg-out Verification




**Maximum Size of Bitcoin Script**

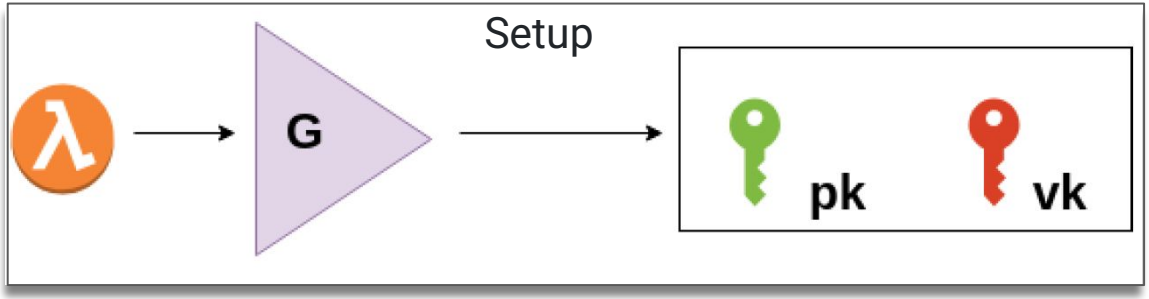
**400 Kbytes**

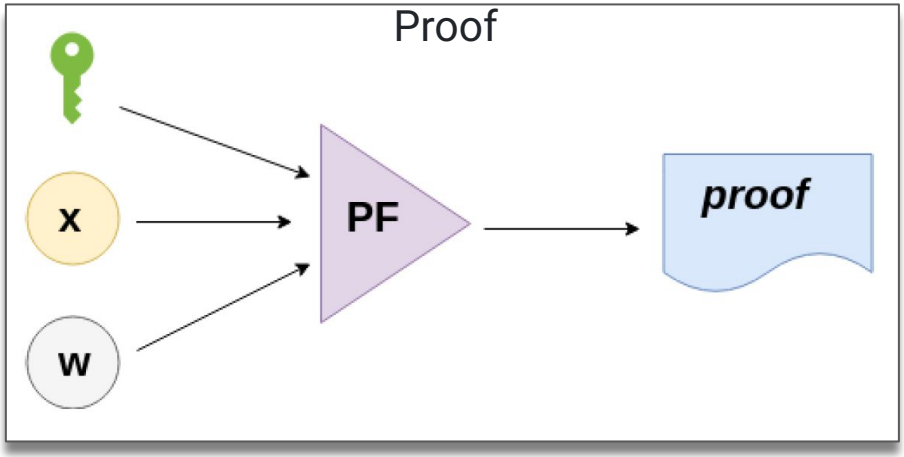
# SNARKs

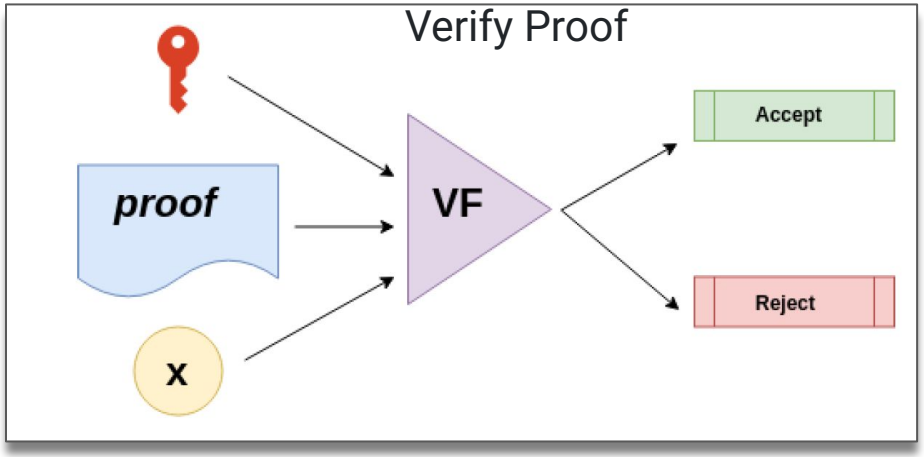
Zero-Knowledge Succinct Non-Interactive Argument of Knowledge.


Proof

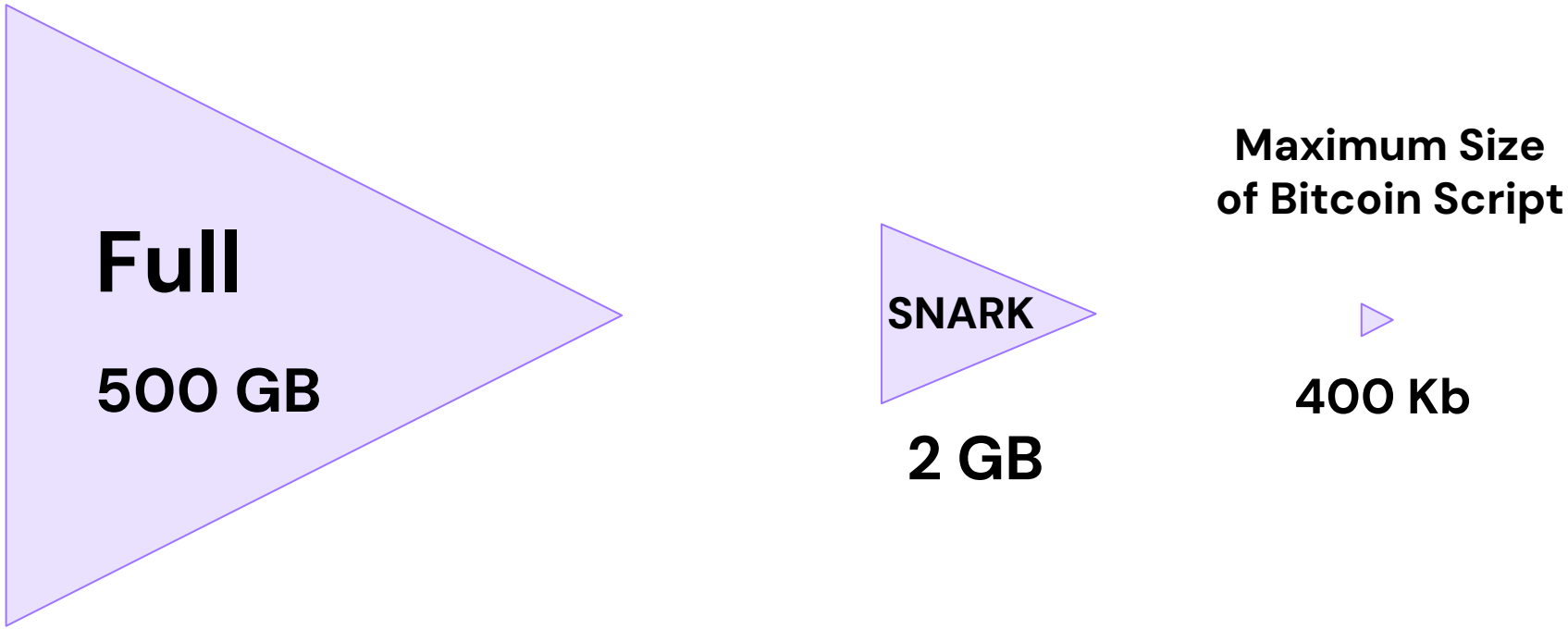
# SNARKs


Setup
G
pk
vk


Proof
x
w
PF
proof


Verify Proof
proof
x
VF
Accept
Reject

# Script Size of Sidechain/Rollup Peg-out Verification

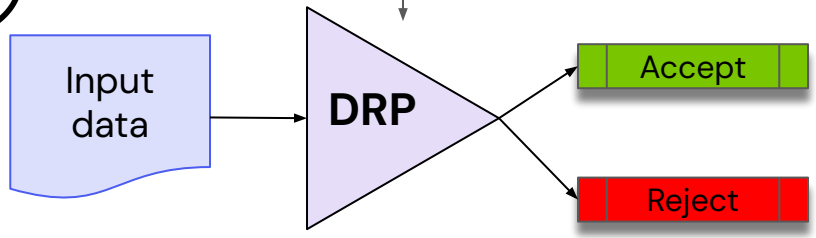


# Disputable Computation Model

Patrick

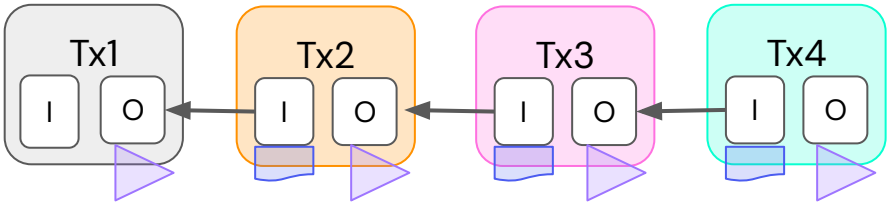

**Prover/ Responder**

Agreed Interactive **Dispute Resolution Protocol (DRP)**




Valerie


**Verifier / Challenger**



# Previous Ideas that Enabled BitVM

- Optimistic Protocols with Fraud Proofs**
  - Blockstream sidechains (2014)
  - TrueBit (Ethereum)
  - Optimistic rollups (Ethereum)
- Verification of Lamport/Winternitz signatures on Bitcoin
- Transferring state between Bitcoin transactions using Lamport Signatures.

# BitVM-like Protocols

- BitVM0 (tapleaf circuits)
- BitVM1 CPU (deprecated)
- BitVM2 (1 round, finalized but deprecated)
- BitVM3 (garbled circuits with reuse,  broken)
- BitVM3s (Disposable GC, almost practical)
- **BitVMX-CPU** (RISC-V, BETA)
- **BitVMX-GC (Mixed Arithmetic Garbled Circuits)**

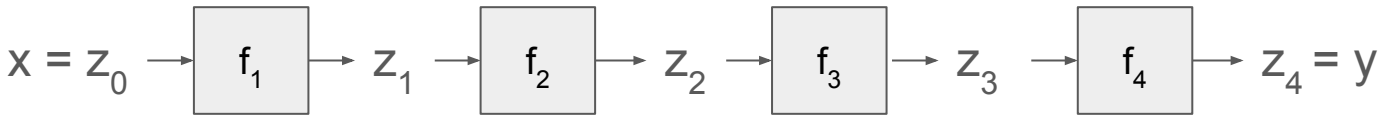
# BitVM Method (pre-GC)

- We have a function f(x) that we want to compute on Bitcoin.
- We divide the function into N “smaller” functions such that...
  $y = f(x) = f_N ( \ldots. f_2( f_1(x) )$
- O sea:
  - $f_1(x) = z_1$
  - $f_2(z_1) = z_2$
  - $f_3(z_2) = z_3$
  - …
  - $f_N(z_{N-1}) = z_N = y$
- Let’s assume that the intermediate states( $z_i$ are small.

# BitVM Method

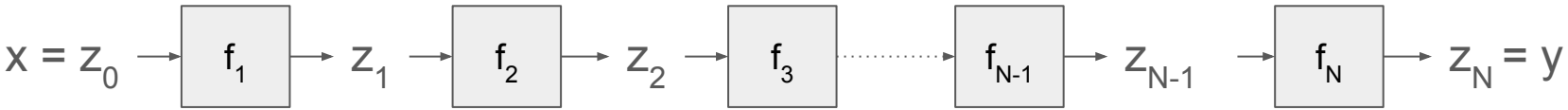
Example Program: f(x) = ((x+ 5)*6*2)-11

Input: x=0

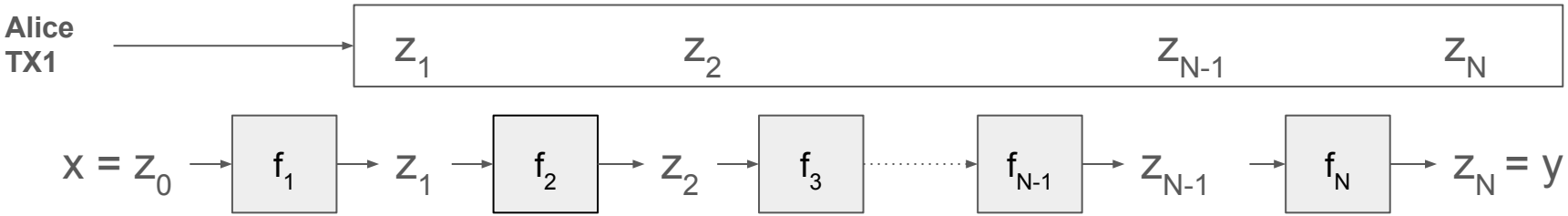



x = 0 + 5 (=5) * 6 (=30) *2 (=60) - 11 = 49

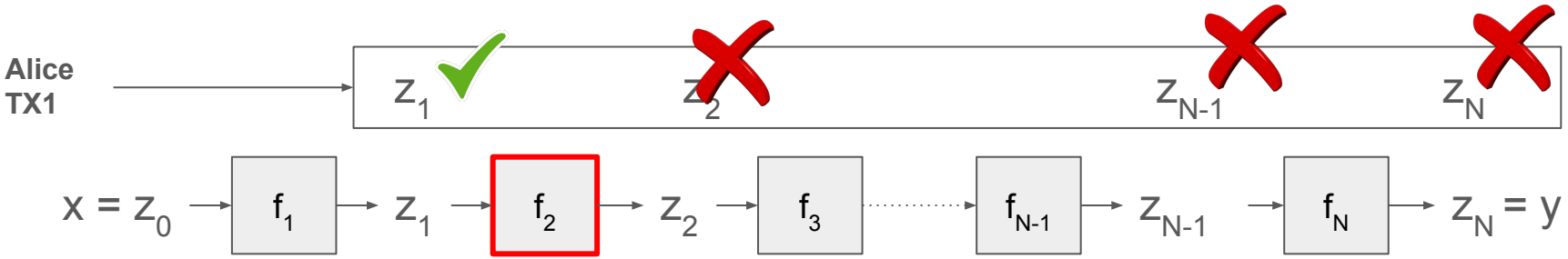
# BitVM2 Method



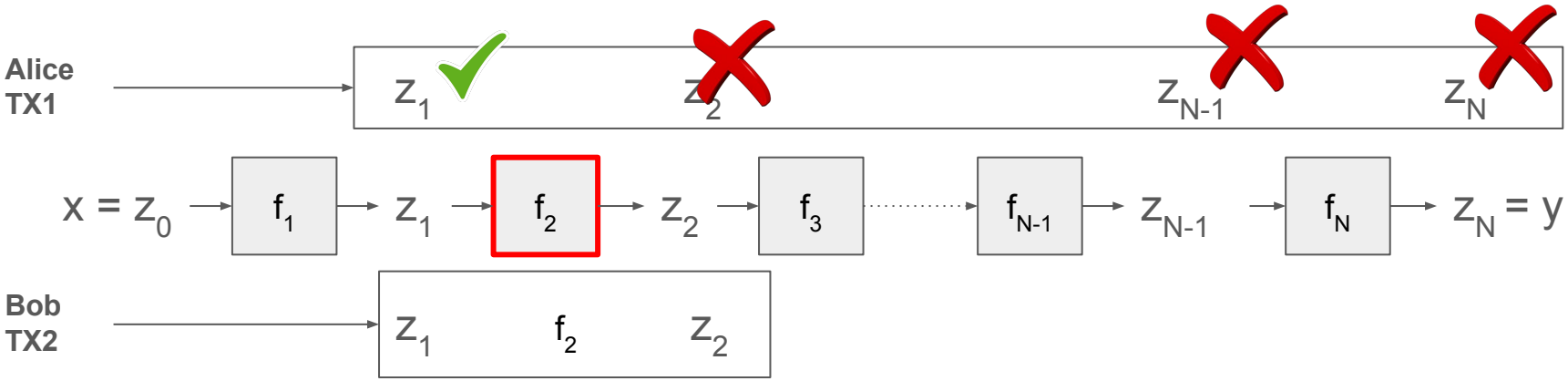
# BitVM2 Method



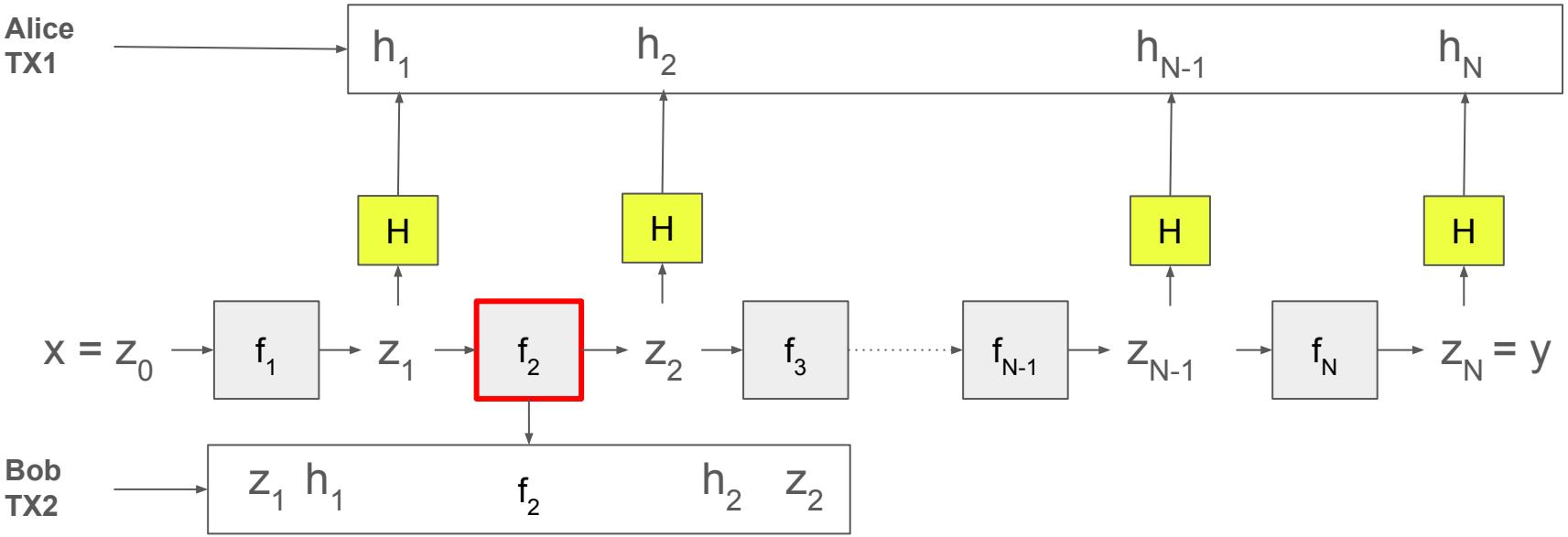
# BitVM2 Method



# BitVM2 Method

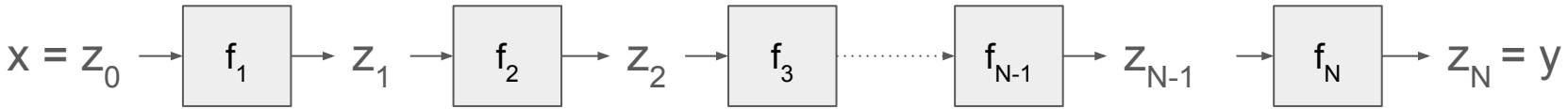


# When the midstate $z_i$ is large

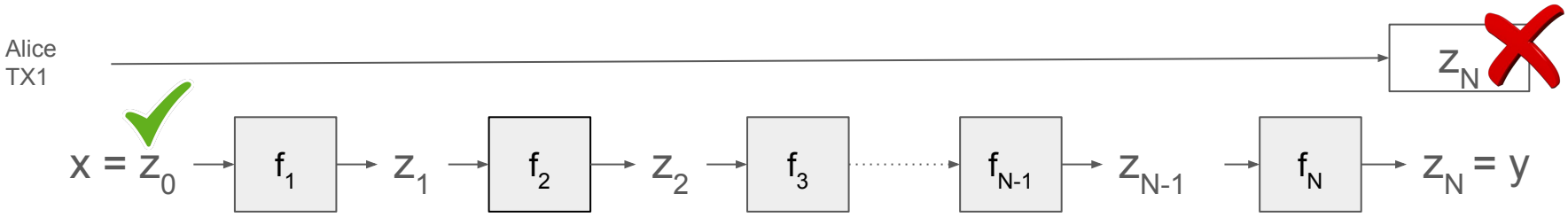



Script: (1) CheckSig($h_1$), (2) $h_1$ == H($z_1$)?, (3) CheckSig($h_2$), (4) $h_2$ == H($z_2$)?, (5) Execute t = $f_2(z_1)$, (6) t == $z_2$ ?

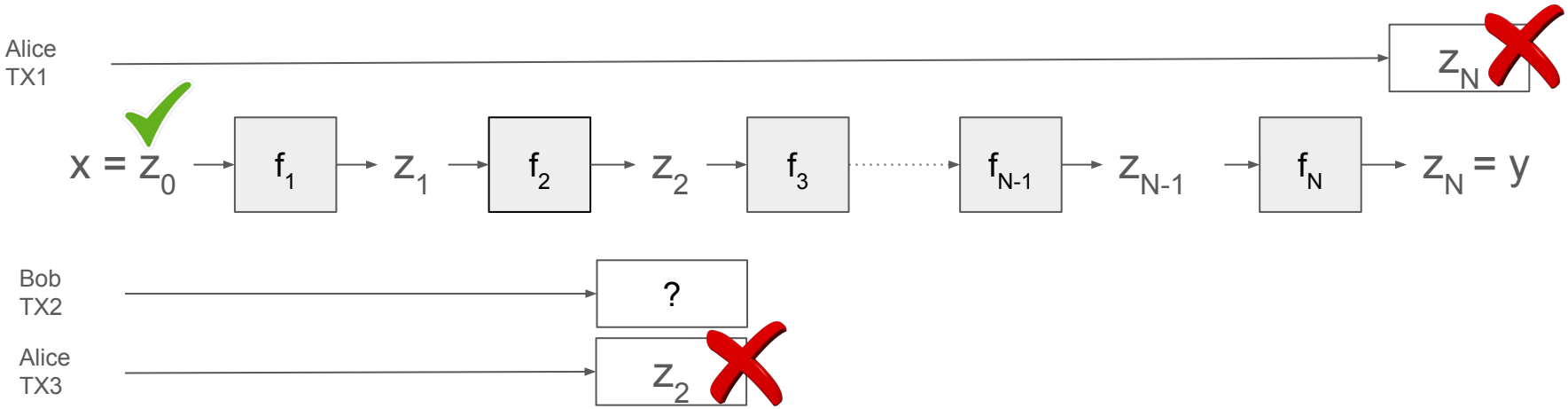
# BitVMX Method



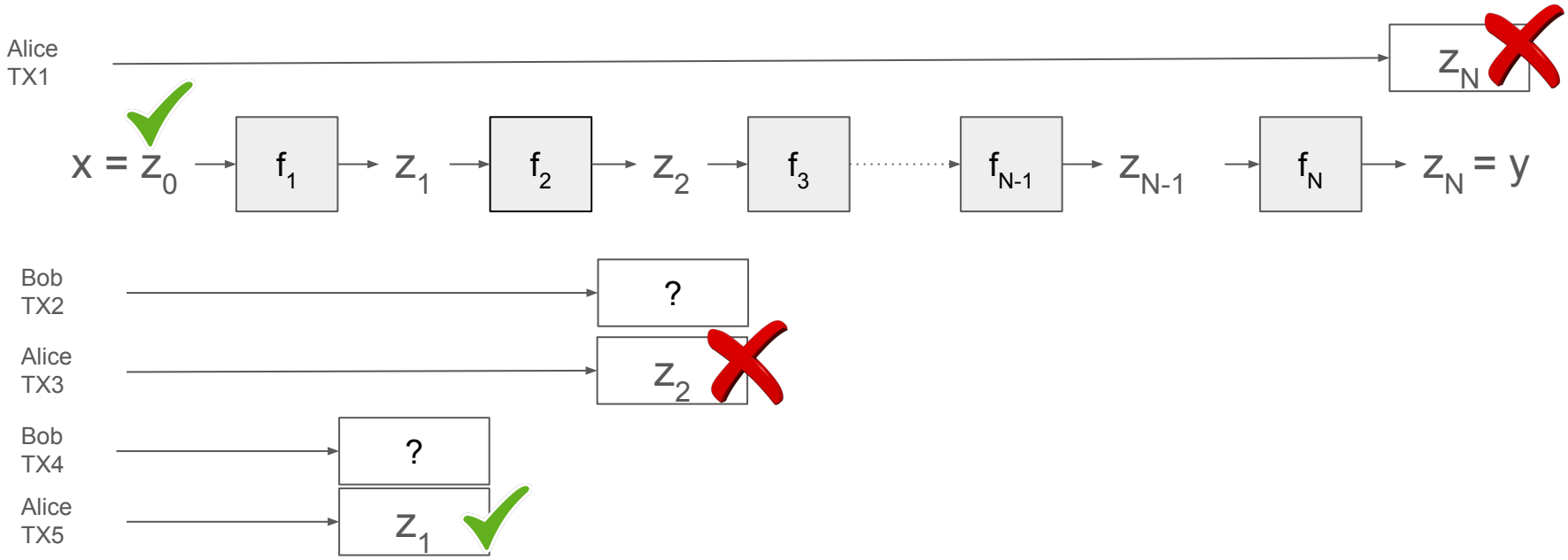
# BitVMX Method



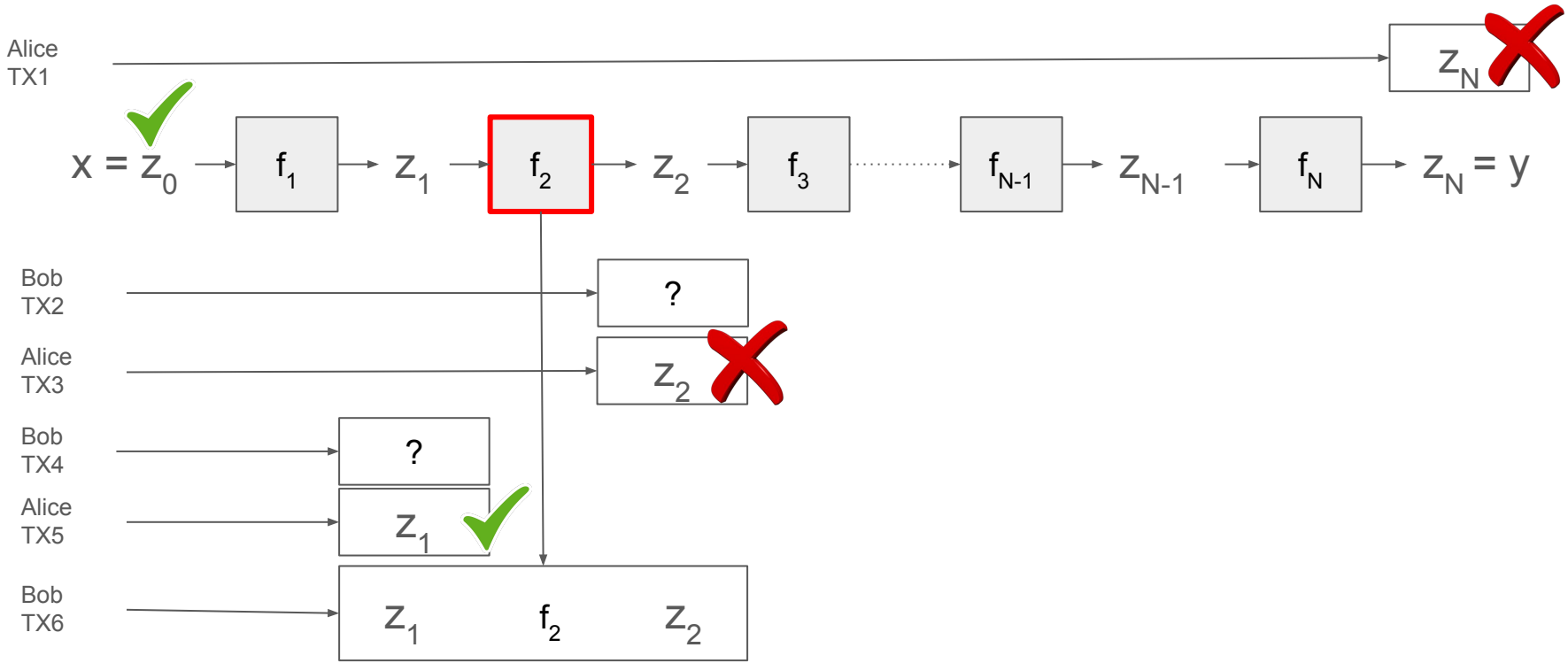
# BitVMX Method



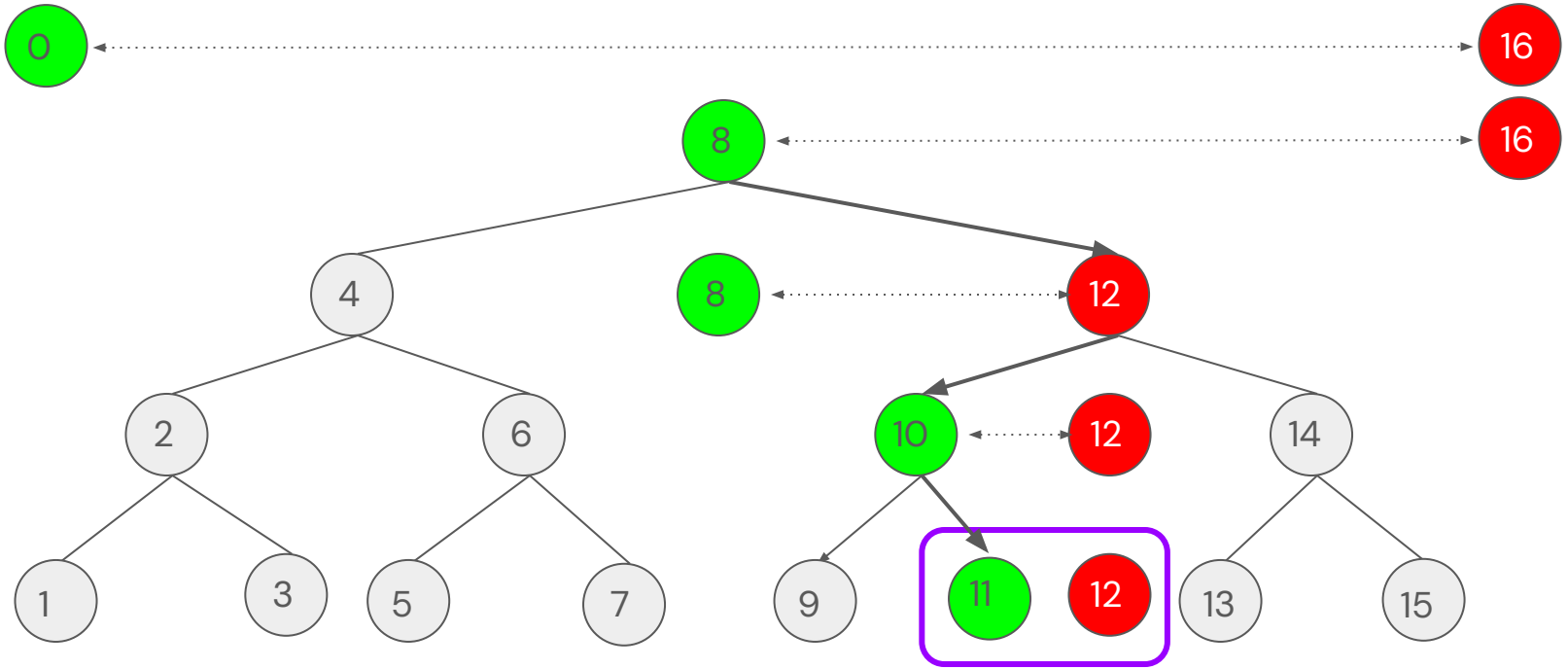
# BitVMX Method



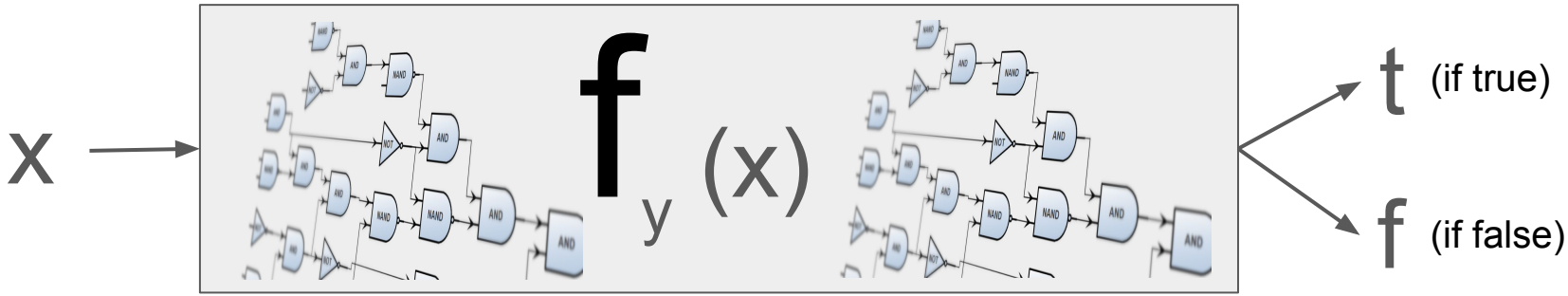
# BitVMX Method



# Malfunction Binary Search



# BitVM-Garbled Circuits Paradigm

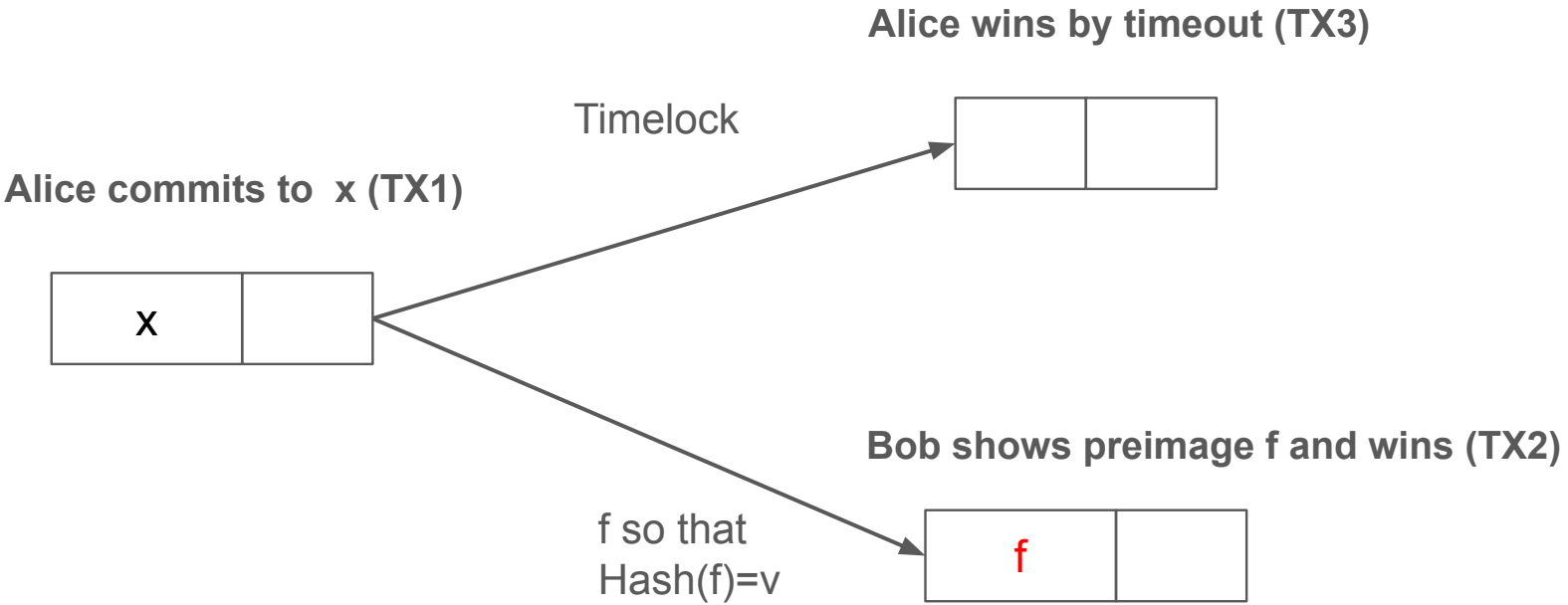



Conditional
Disclosure of Secrets

**Alice**
**TX1** → X

**Bob**
**TX2** → f

# HTLCs in BitVM-Garbled Circuits



# Program Sizes to Verify a Sidechain Peg-out

| Scripts a verificar | Tamaño |
|---|---|
| Todo | 800 GB |
| SNARK | 8 GB |
| BitVM2 | 4 MB |
| BitVMX-CPU | 400 Kbytes |
| Garbled Circuits | 4 Kbytes |

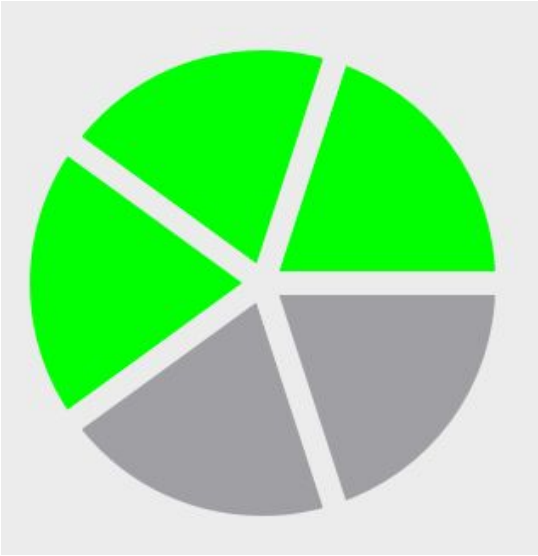
# BitVM Limitations

- Protocol between N known participants
- Security condition: 1 of N is honest
- No contextual information (time, block hash)
- Minimum computation cost is non-negligible
- Persistent storage cost is non-negligible
- Builds immutable transaction graphs to emulate covenants
- It is not possible to pay third parties directly; possible alternatives include:
  - Repayments
  - TOOP
  - Trust-minimized Covenants (coming soon…)

# 1 of N Honesty

Fairgate

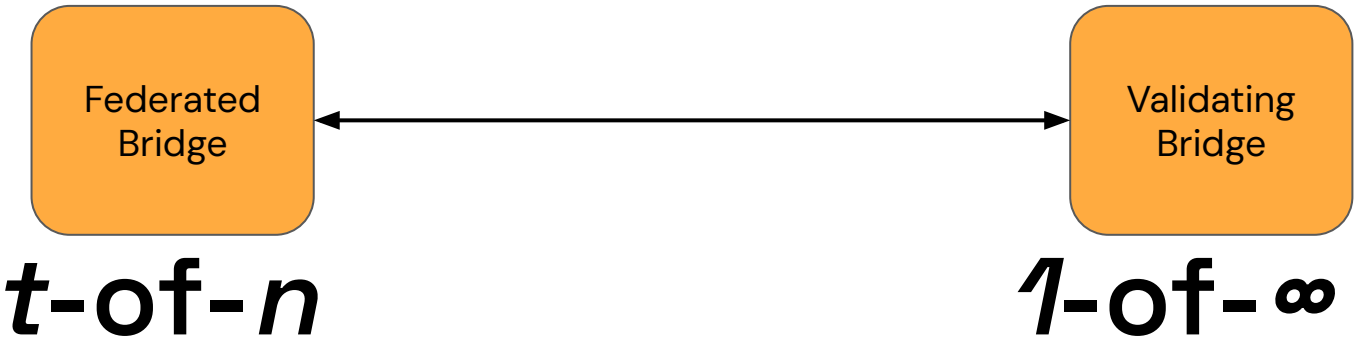
# Federated Model (t-of-N)



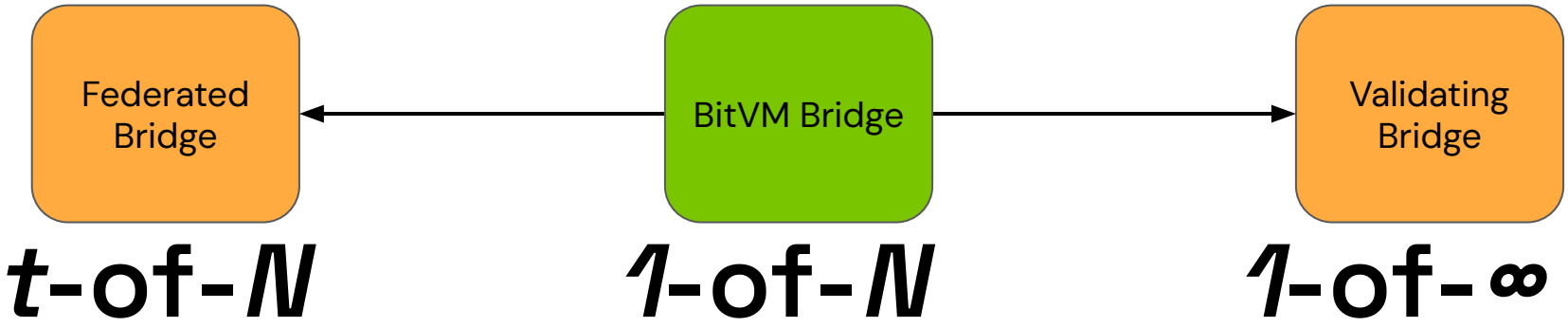
All federated bridges are based on the t-of-N security model. This model assumes that t members of a security committee of N are honest, in order to protect the funds. Normally, t represents the majority.

The committee cannot be permissionless: the anonymous addition of new members would dilute security through a Sybil attack.

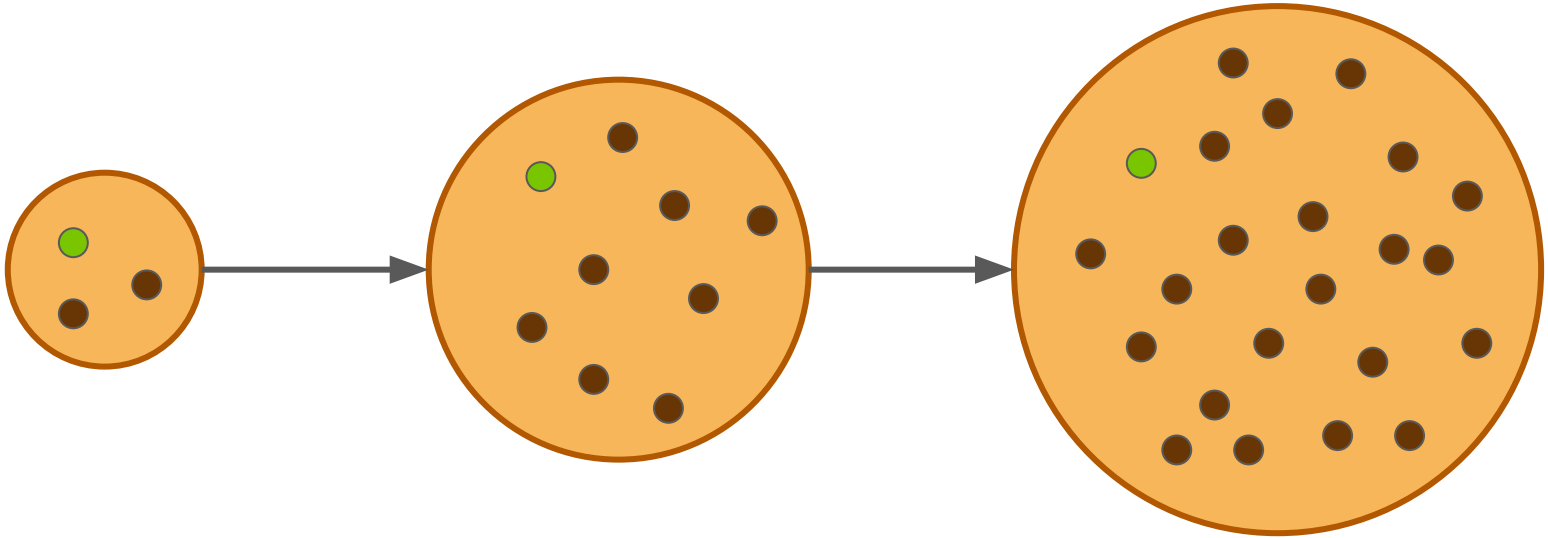
# From *t*-of-*n* to *1*-of-∞



# New *1*-of-*n* honesty assumption



# Openness Without Security Dilution



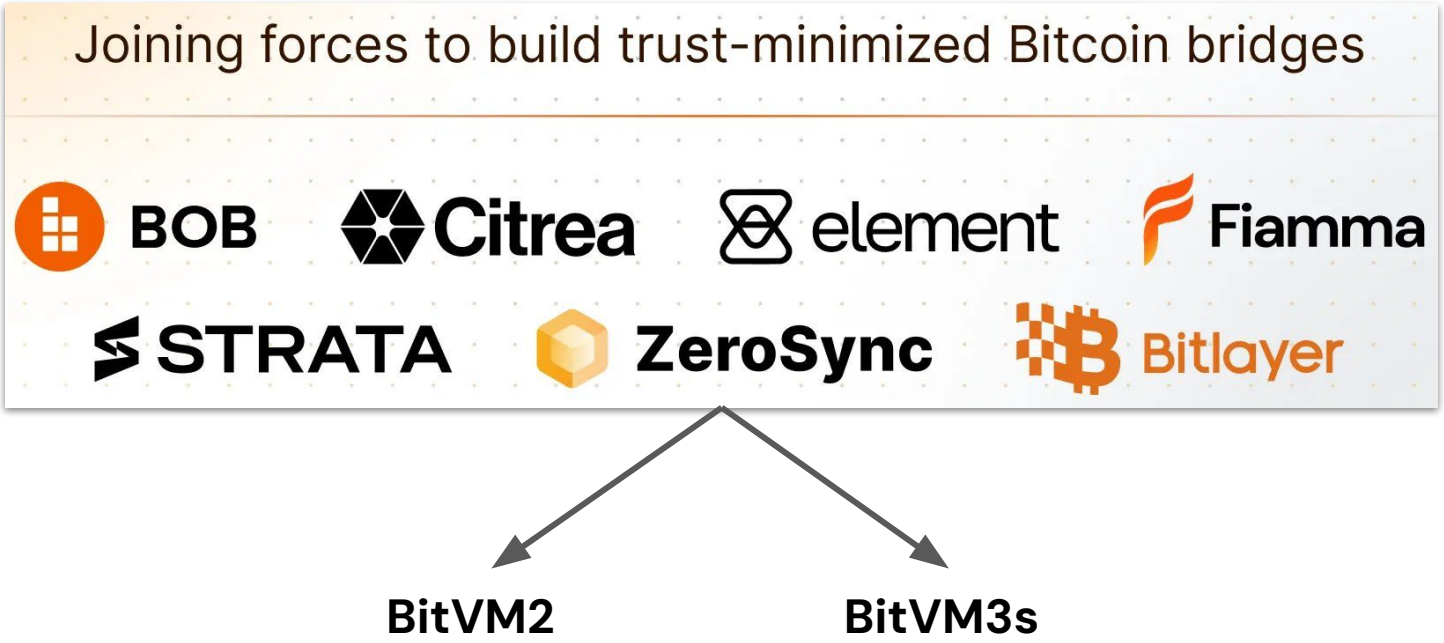
- Honest party
- Party of unknown behaviour

# BitVM 2025

Fairgate

# BitVM/BitVMX Adoption on Bitcoin L2s

# BitVM Alliance



# **BitVMX**: A Platform for Universal Computation on Bitcoin

**BitVMX** is a cutting-edge framework designed to optimistically execute arbitrary programs on Bitcoin, leveraging the N-party disputable computation paradigm introduced by BitVM.

With its foundation in secure, extensible, and open-source principles, **BitVMX paves the way for running any CPU on Bitcoin**.

Rootstock Labs

FairGate LABS

# BitVM FORCE


FOUNDING MEMBERS

Rootstock Labs

Fairgate

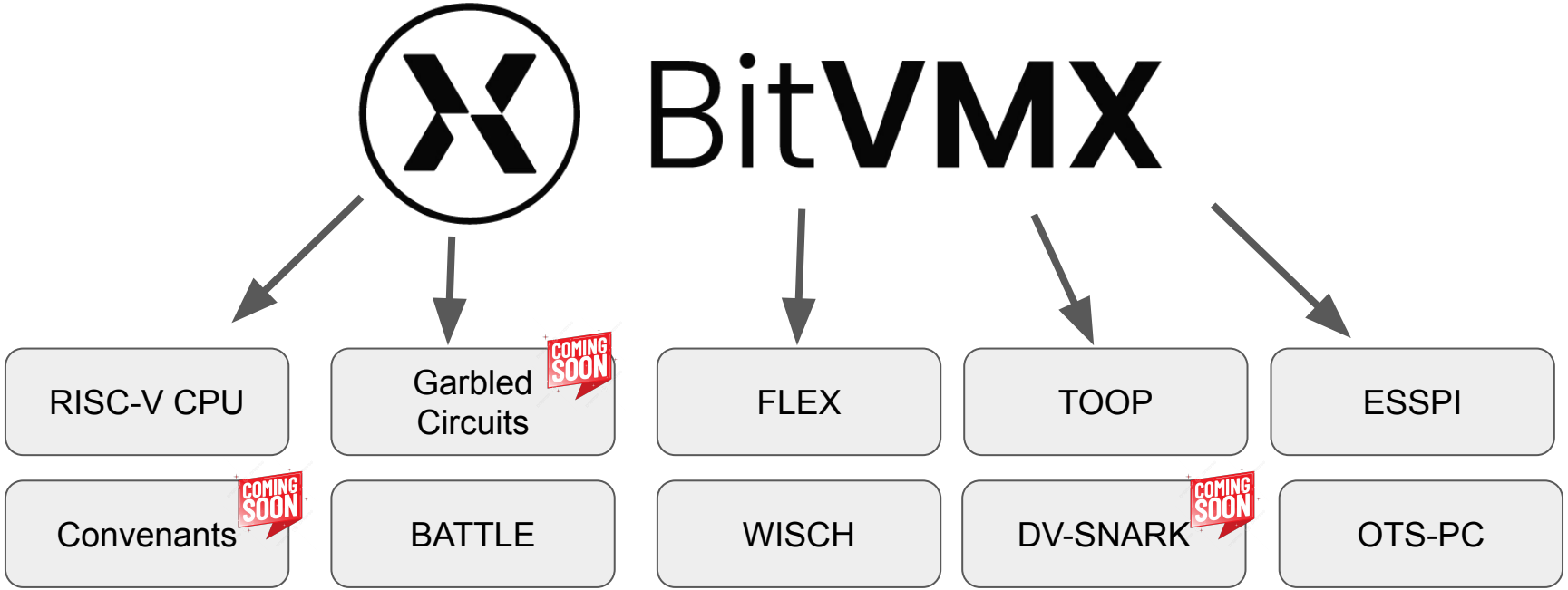
INPUT | OUTPUT

BitVMX-CPU

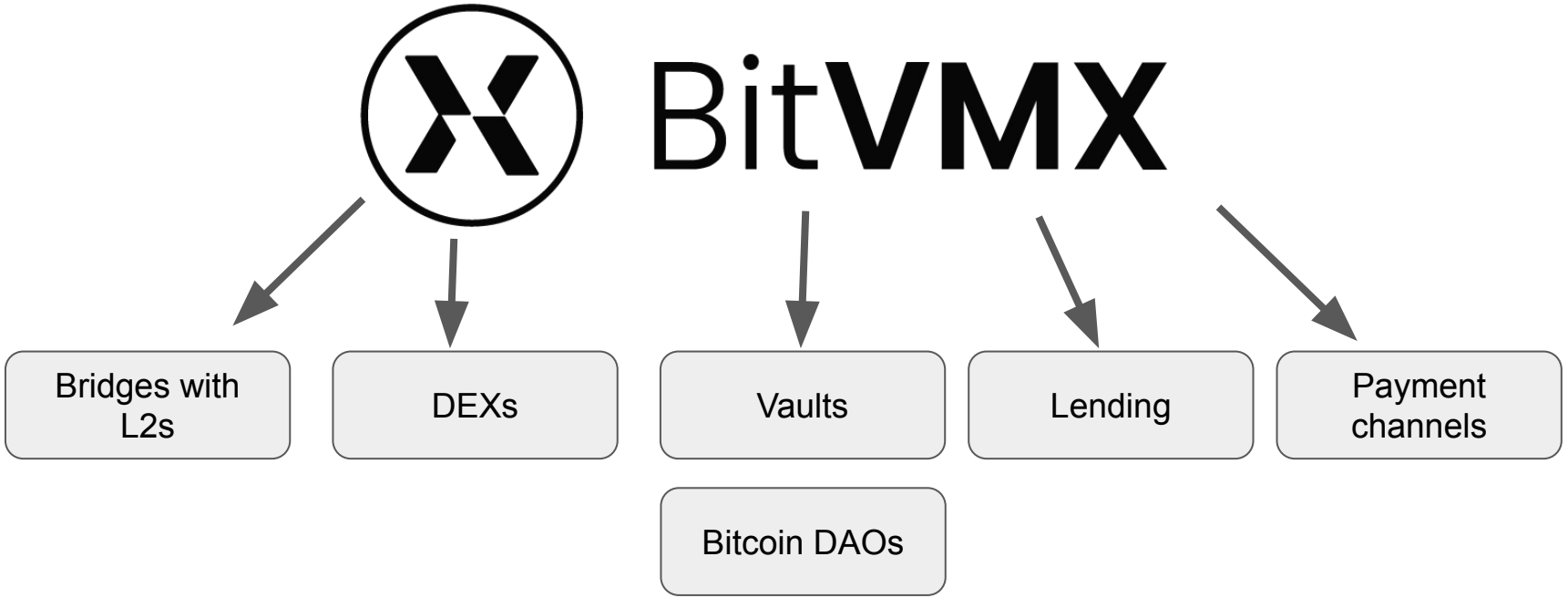
BitVMX Platform

BitVMX-GC

# New Components to the BitVMX Family

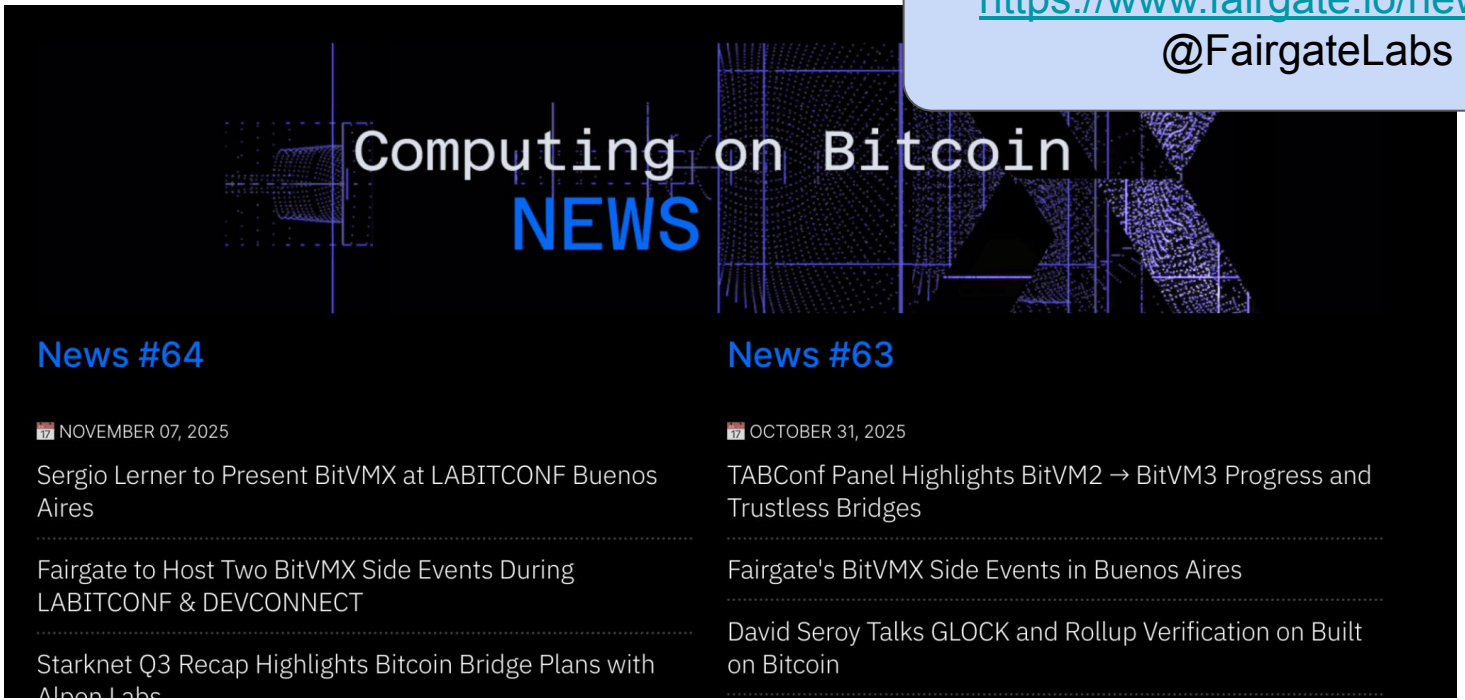


# New Possible Applications



# BitVM/BitVMX in the news

https://www.fairgate.io/newsletters/
@FairgateLabs



# Visit BitVMX.org !

# Q&A


# BitVM
# in the News

Fairgate

# BitVM/BitVMX in the news

COINTELEGRAPH
The future of money

BRAYDEN LINDREA

OCT 10, 2023

## Bitcoiner drops BitVM paper — Bringing Ethereum-like contracts to Bitcoin

The author of the white paper, Robin Linus, based BitVM's architecture on Ethereum's optimistic rollups with fraud proofs and recent Merkle tree developments.

Fairgate

# BitVM/BitVMX in the news

CoinDesk

Technology

## Bitcoin Might Get Ethereum-Style Smart Contracts Under 'BitVM' Plan

A goal of Robin Linus's design for "BitVM" is to enable Turing-complete Bitcoin contracts – making the blockchain programmable, similar to a computer – without making the network more complicated for other users.

By Jamie Crawley Oct 11, 2023 at 12:08 p.m. Updated Oct 23, 2023 at 1:21 p.m.

Fairgate

# BitVM/BitVMX in the news


Fairgate

# BitVM/BitVMX in the news

COINTELEGRAPH
The future of money

JESSE COGHLAN

FEB 22, 2024

## Citrea raises $2.7M in seed funding to launch Bitcoin ZK-rollup

Chainway Labs' ZK-rollup project Citrea secured initial funding to bring the solution to market, which could allow for NFTs and blockchain games on Bitcoin.

Fairgate

# BitVM/BitVMX in the news

CoinDesk

Technology

## Alpen Labs Looks to Scale Bitcoin With Zero-Knowledge Proofs Using $10.6M Funding

Alpen Labs has emerged from stealth mode following the funding round, having spent the last year developing Bitcoin rollup infrastructure to bring smart contract functionality to the network

By Jamie Crawley Apr 10, 2024 at 12:11 p.m. Updated Apr 10, 2024 at 12:14 p.m.

Fairgate

# BitVM/BitVMX in the news


Fairgate

# BitVM/BitVMX in the news


## Bitcoin Layer 2 Rootstock Verifies Zero-Knowledge SNARK (on Bitcoin!)

Jamie Crawley
Updated July 26, 2024

Fairgate

# BitVM/BitVMX in the news

CoinDesk

Tech

## Bitcoin Ordinals Can Now Be Bridged to Cardano Through BitVMX

The on-chain transaction between Bitcoin and Cardano was facilitated by BitVMX, an interoperability protocol built using the BitVM computing paradigm

By Jamie Crawley | Edited by Parikshit Mishra

Updated Jun 9, 2025, 10:43 a.m. Published May 28, 2025, 6:48 a.m.

Fairgate