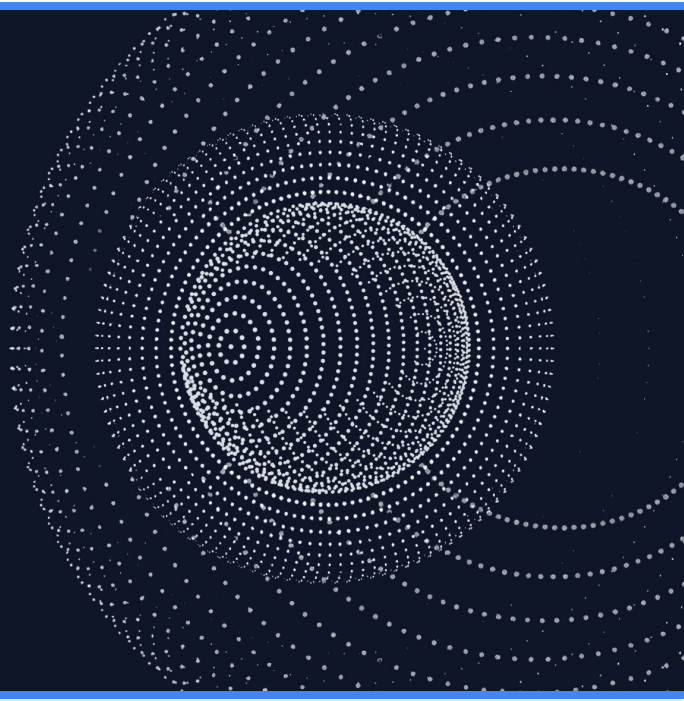
Fairgate

BitVMX and Fairgate's Evolving Ecosystem of Protocols



Sergio Demian Lerner | CTO, Fairgate Labs

NOV 16, 2025

Why should Bitcoin 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.

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)

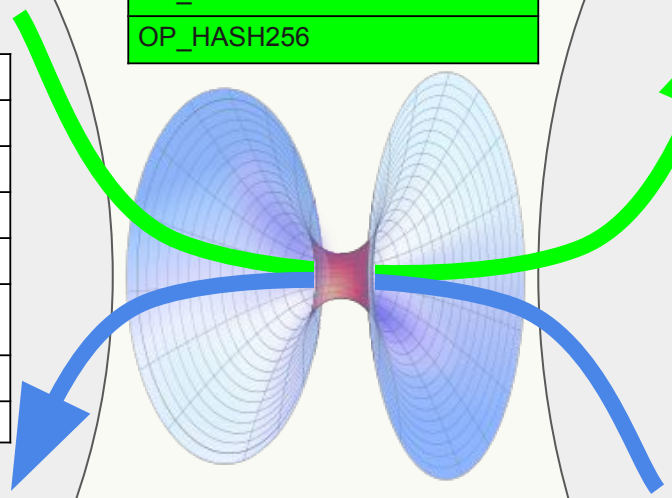
- OP_BOOLAND
- OP_BOOLOR
- OP_NUMEQUAL
- OP_NUMEQUALVERIFY
- OP_NUMNOTEQUAL
- OP_LESSTHAN
- OP_GREATERTHAN
- OP_LESSTHANOREQUAL
- OP_GREATERTHANOREQUAL
- OP_MIN
- OP_MAX
- OP_WITHIN

- 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

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

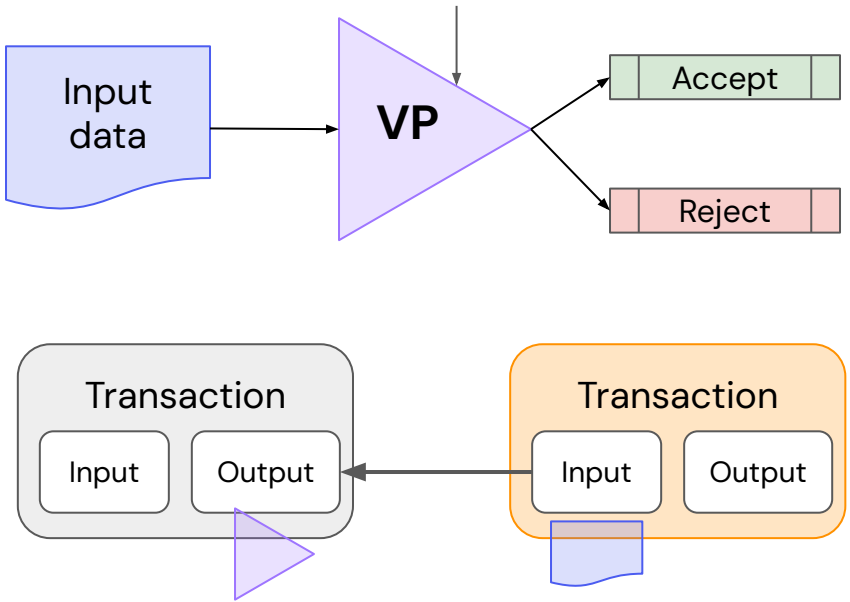
Fairgate

Computation on Bitcoin

Patrick

Prover

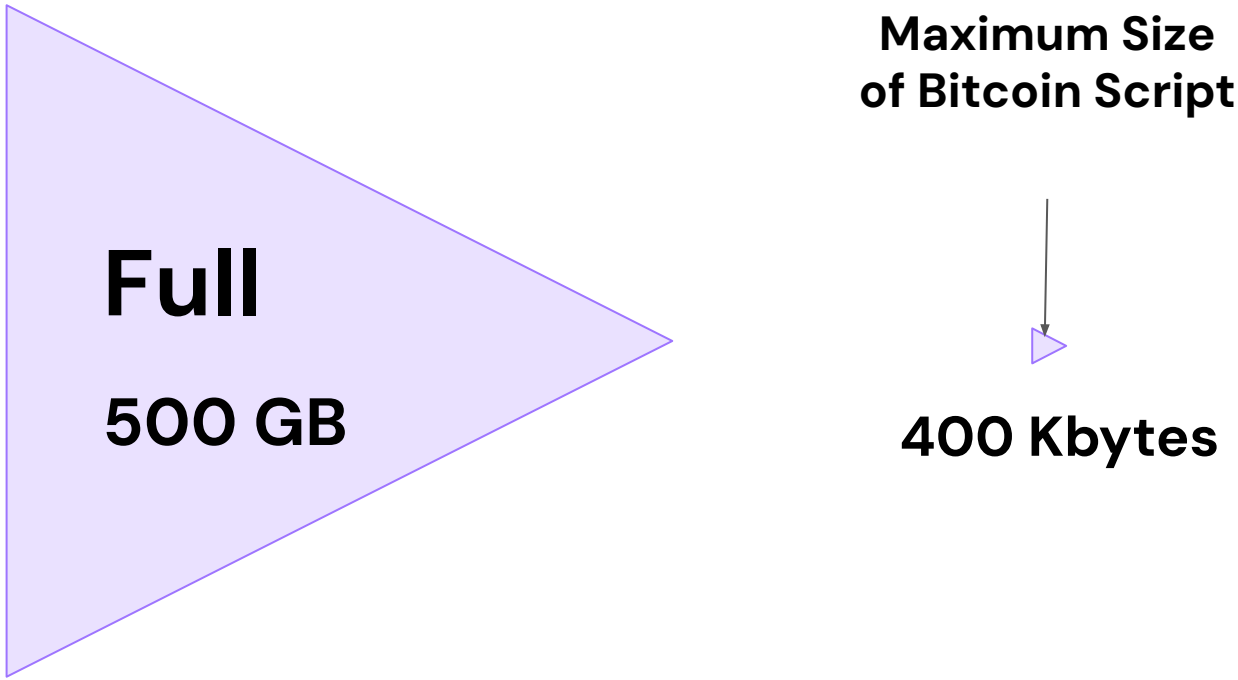
Agreed **Verification Program (VP)**



Valerie

Observer

Script Size of Sidechain/Rollup Peg-out Verification

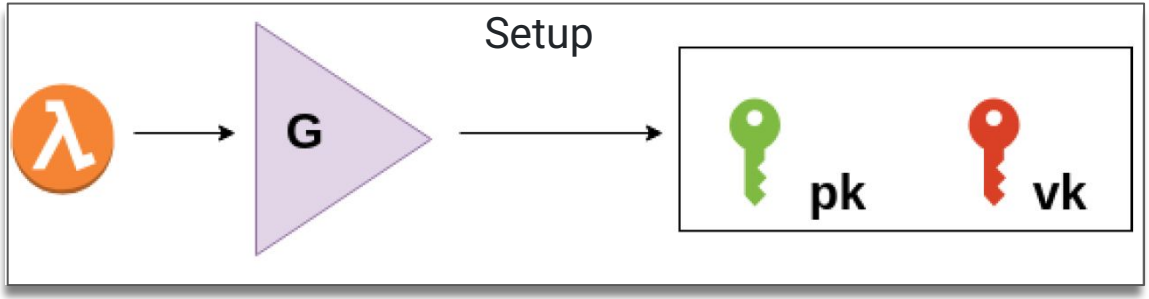


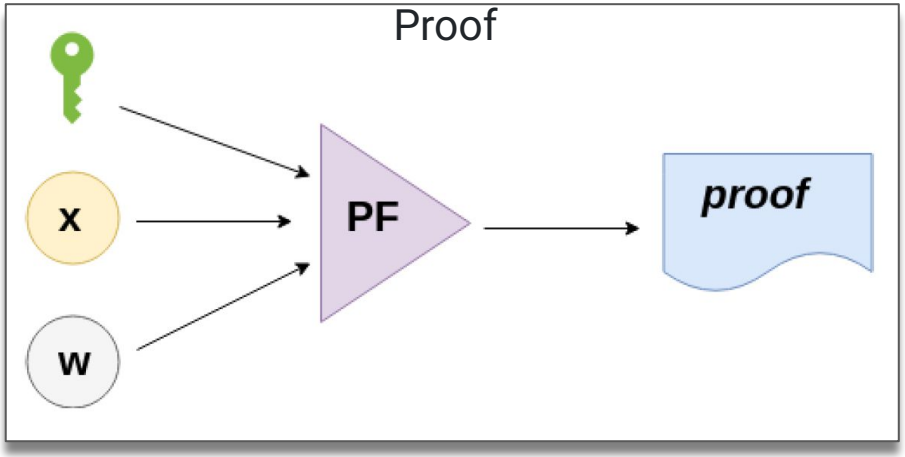
SNARKs

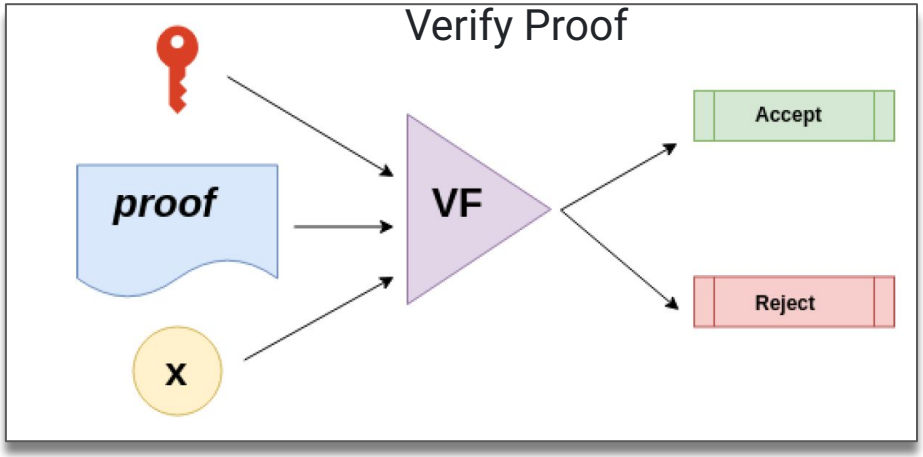
Zero-Knowledge Succinct Non-Interactive Argument of Knowledge.

Proof

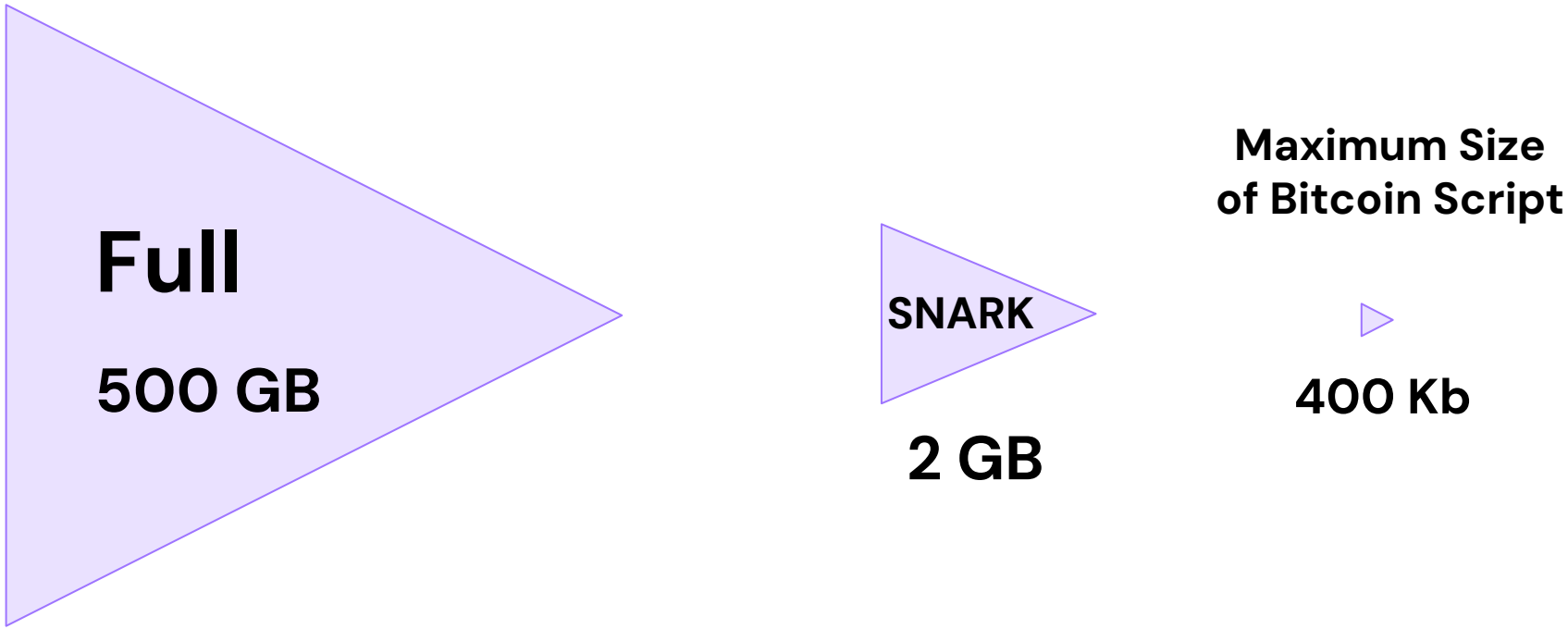
SNARKs


Setup
λ
G
pk
vk


Proof
x
w
PF
proof


Verify Proof
proof
x
VF
Accept
Reject

Script Size of Sidechain/Rollup Peg-out Verification

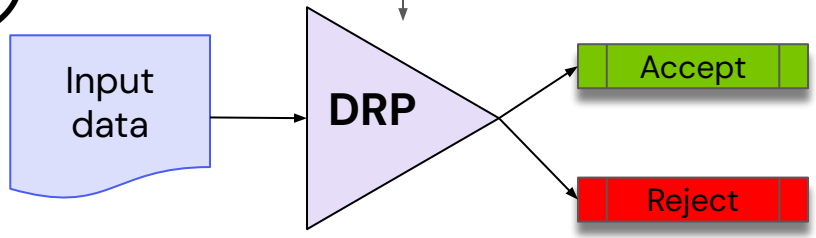


Disputable Computation Model

Patrick

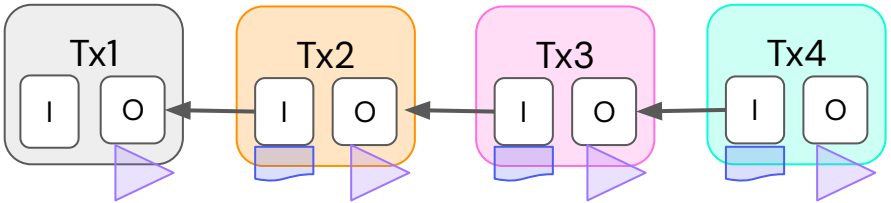
Prover/ Responder

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)**

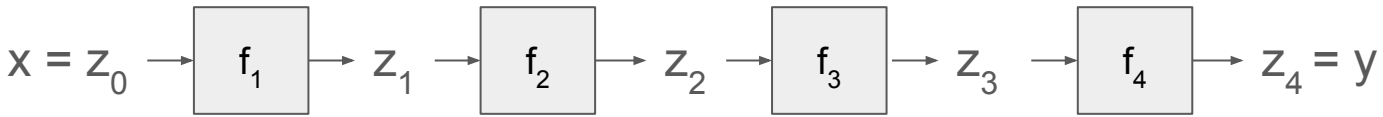
BitVM Method (pre-GC)

- 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

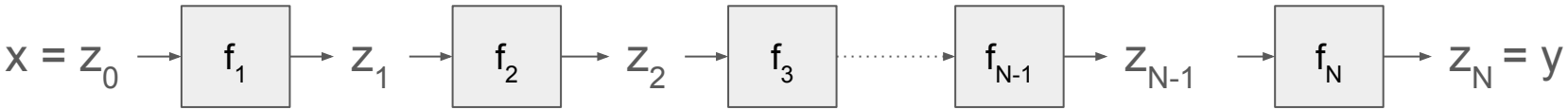
Example Program: f(x) = ((x+ 5)*6*2)-11

Input: x=0

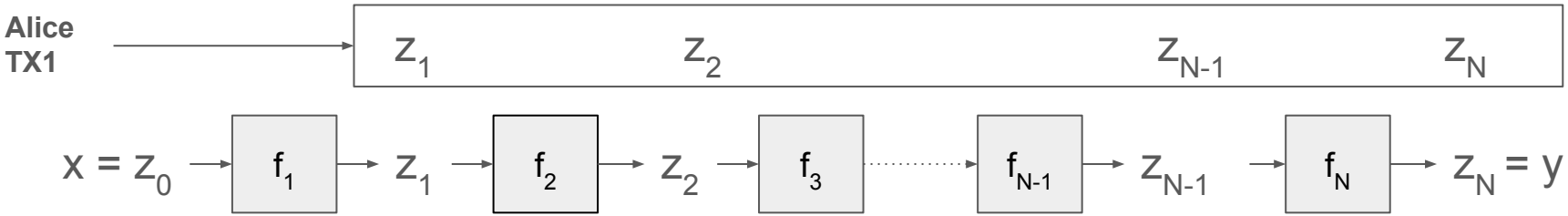


x = 0 + 5 (=5) * 6 (=30) *2 (=60) - 11 = 49

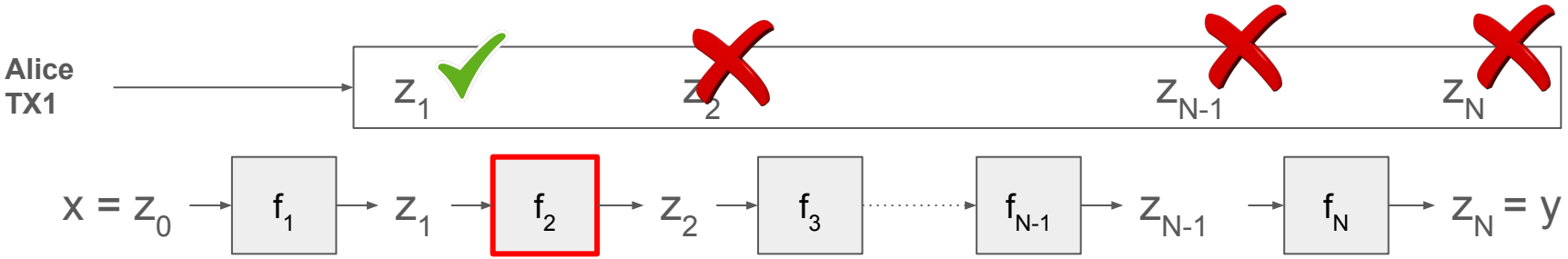
BitVM2 Method



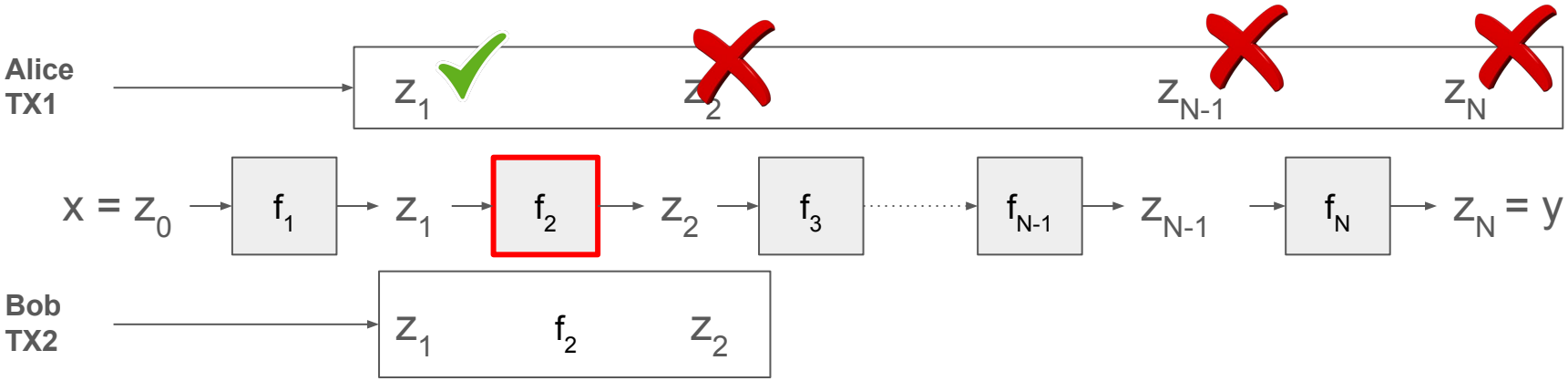
BitVM2 Method



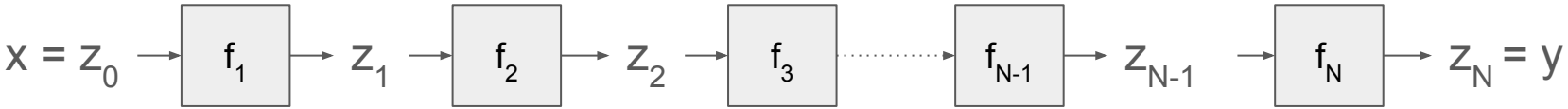
BitVM2 Method



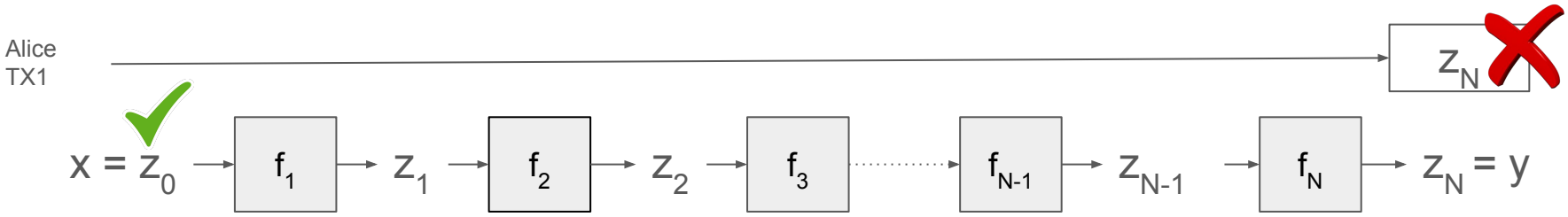
BitVM2 Method



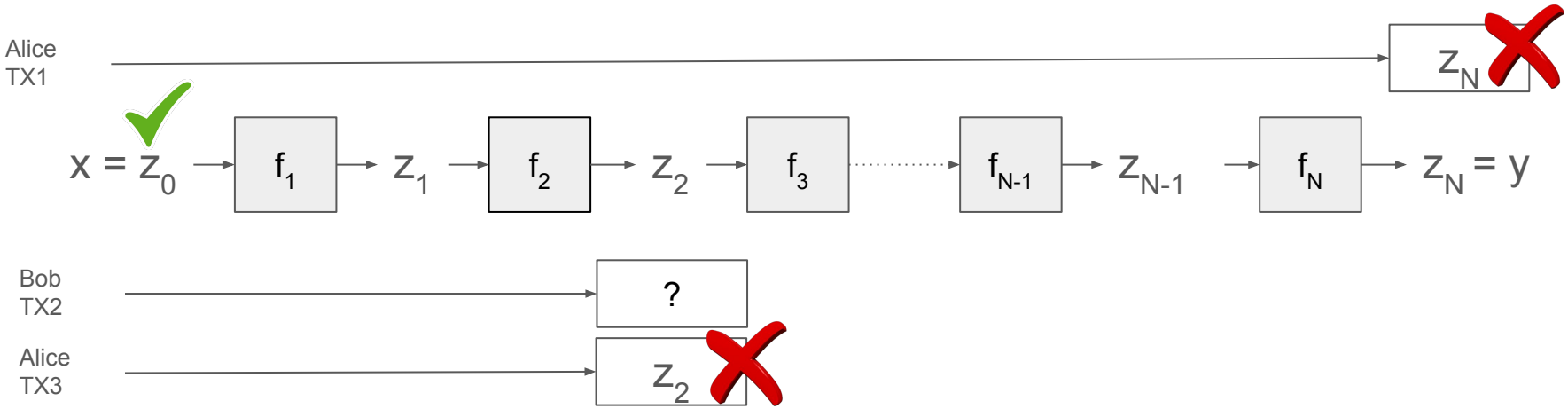
BitVMX-CPU Method



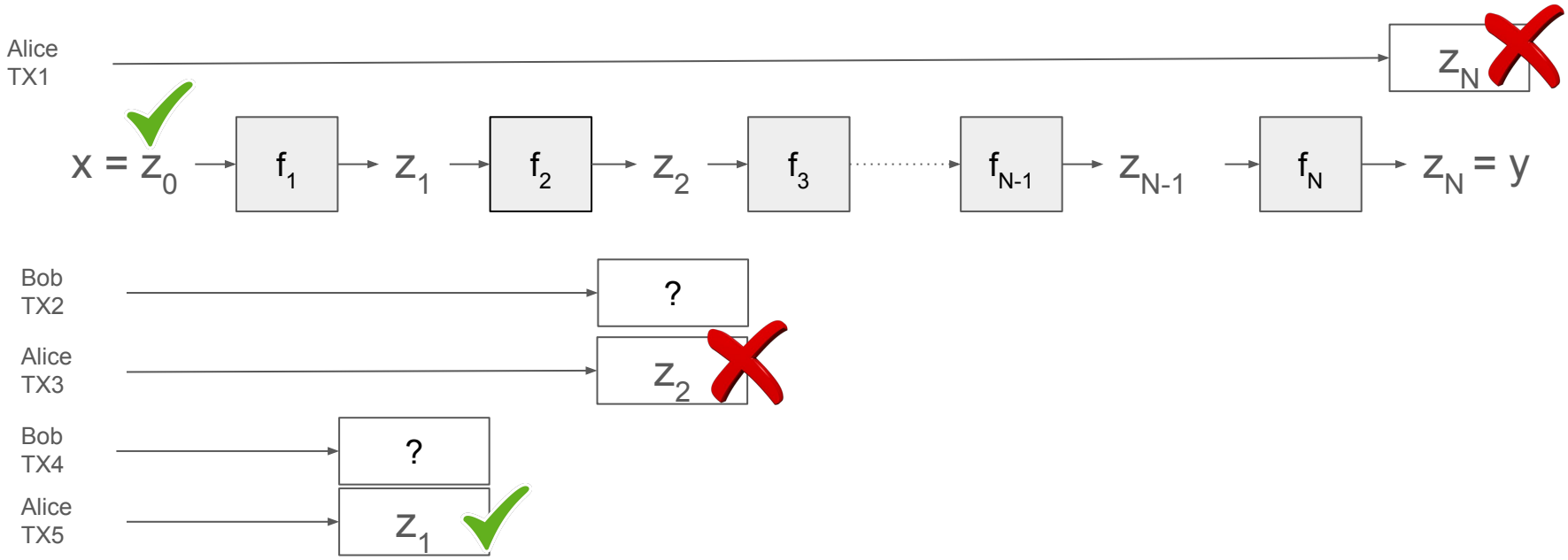
BitVMX-CPU Method



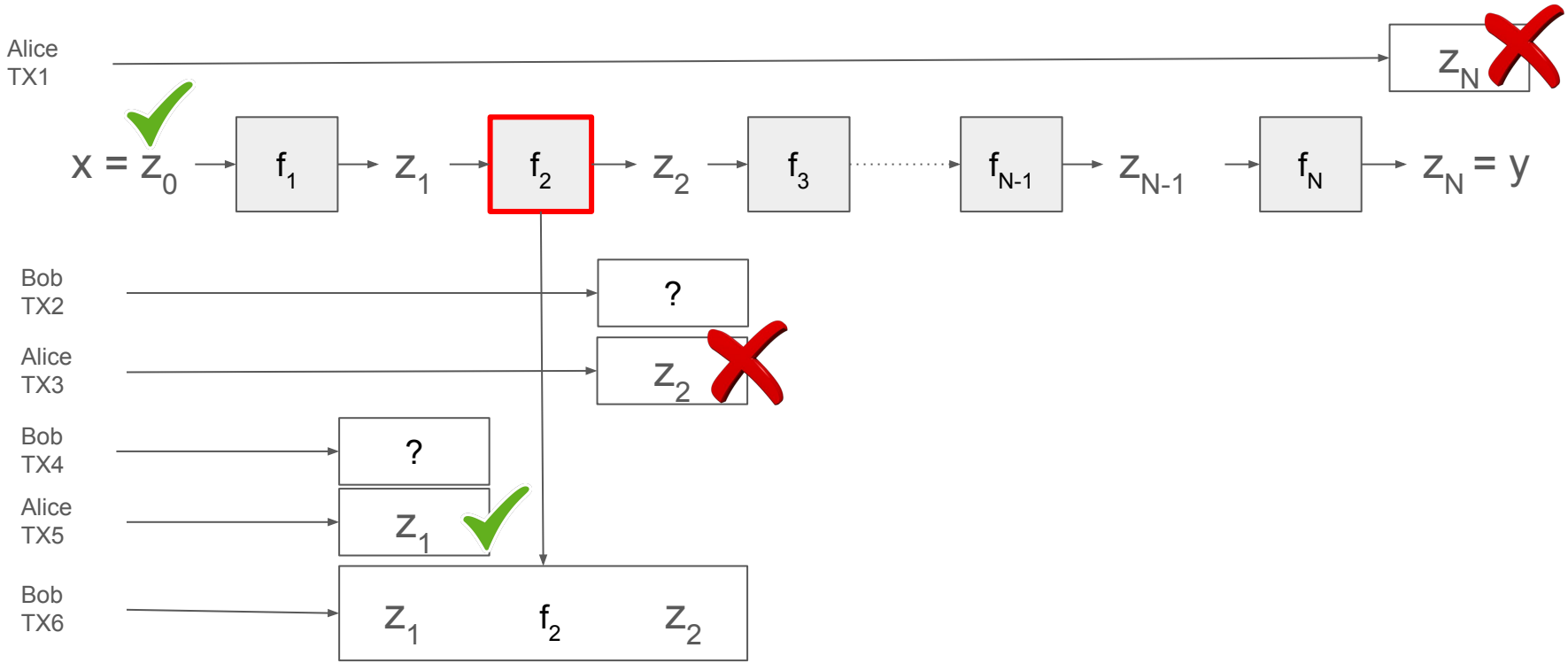
BitVMX-CPU Method



BitVMX-CPU Method



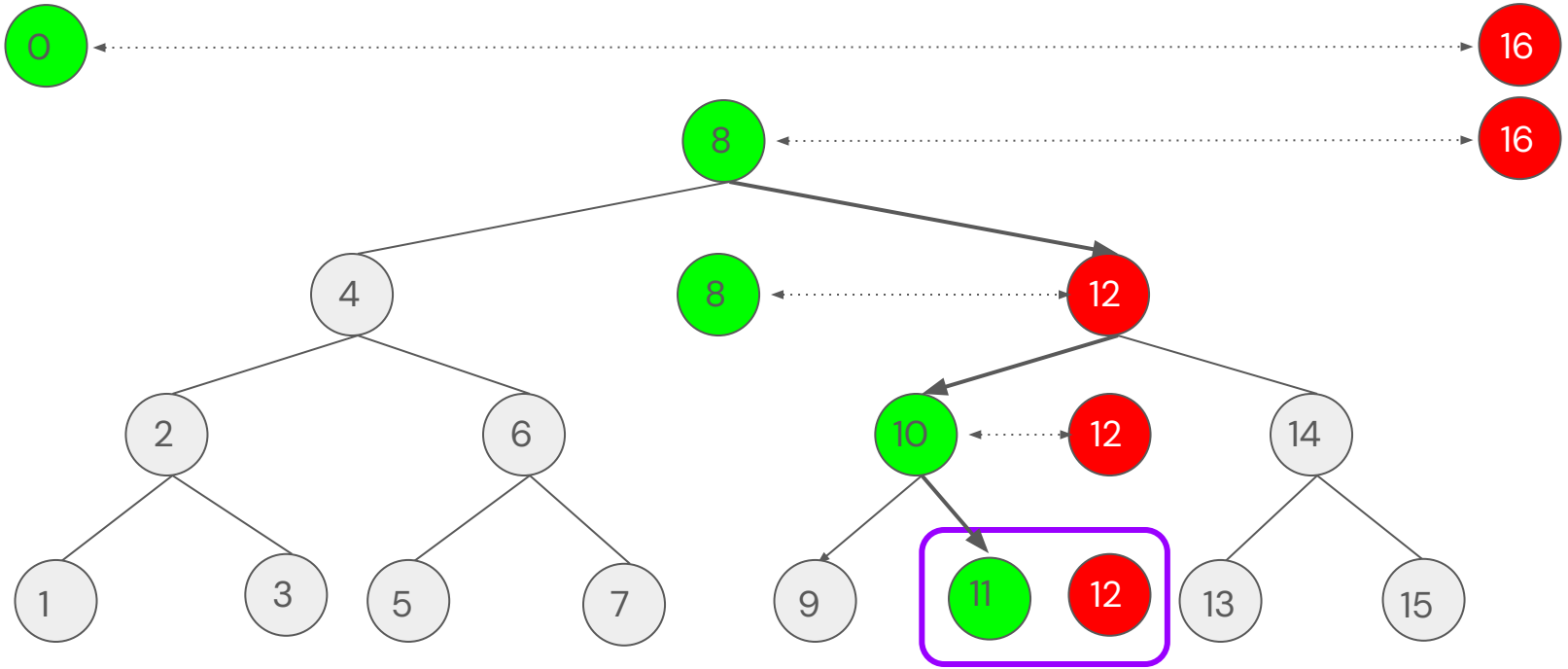
BitVMX-CPU Method



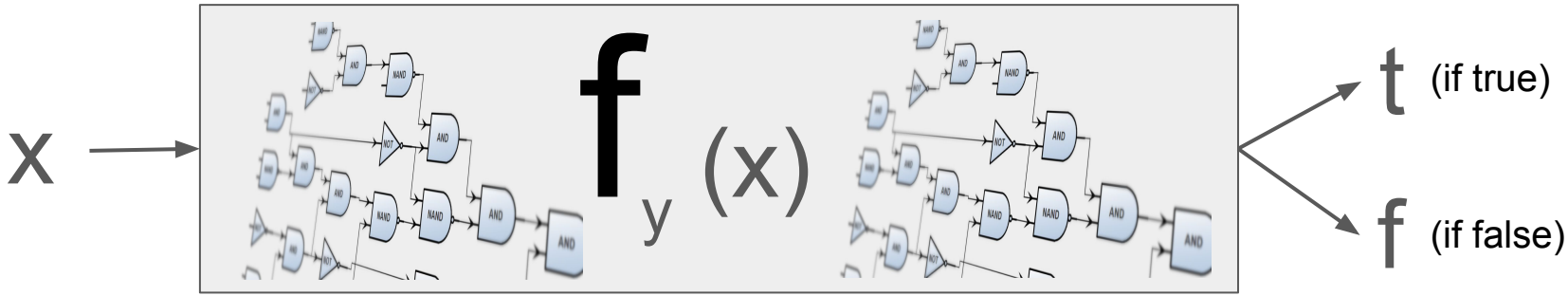
Fairgate

BitVMX

Malfunction Binary Search



BitVMX-CPU: BitVM-Garbled Circuits Paradigm

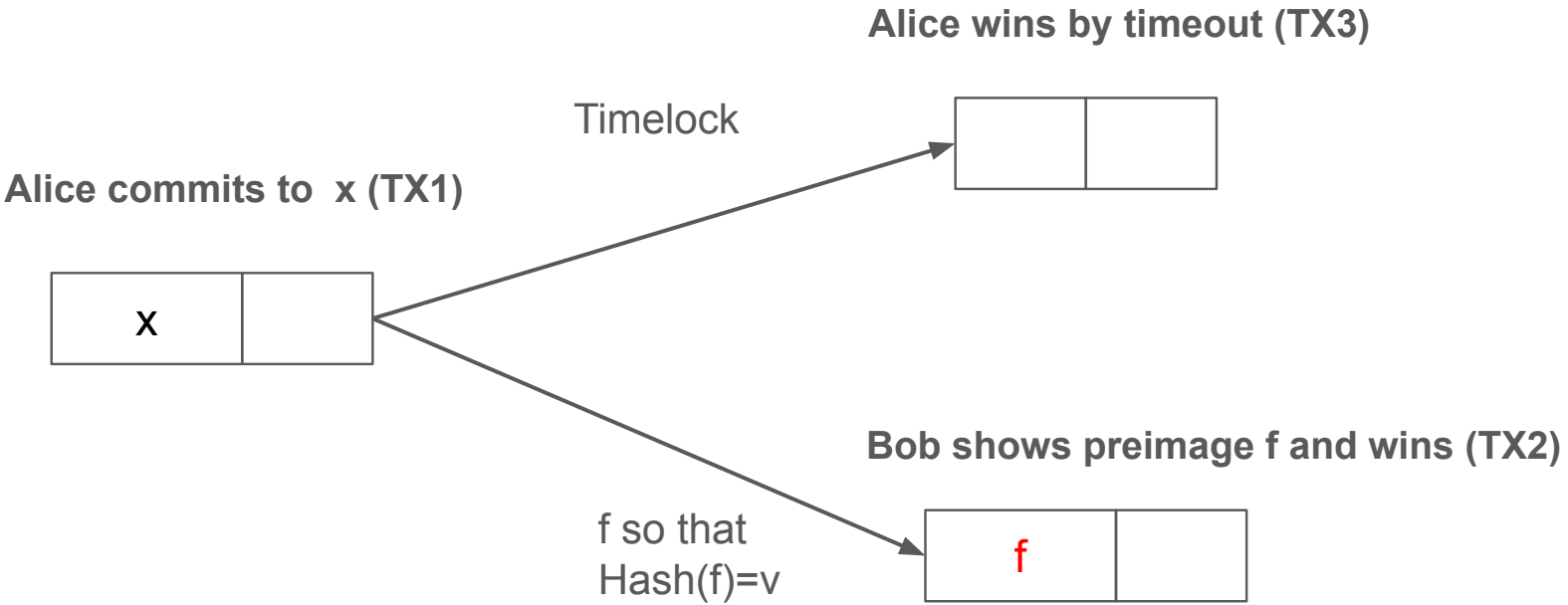


Conditional
Disclosure of Secrets

Alice
TX1 → X

Bob
TX2 → f

HTLCs in BitVM-Garbled Circuits



Program Sizes to Verify a Sidechain Peg-out

Scripts to Verify	Script+Midstate+Input Size
1000 PoW confirmation blocks (e.g. RSK)	~500 GB
SNARK	2 GB
BitVM2	4 MB
BitVMX-CPU	400 Kbytes
Garbled Circuits	4 Kbytes

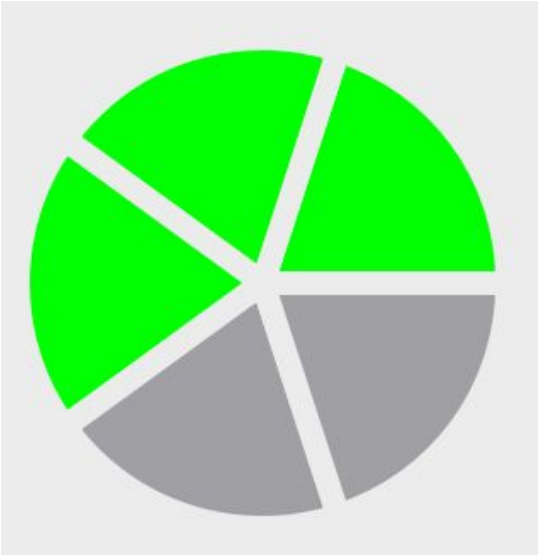
BitVM Limitations

- 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

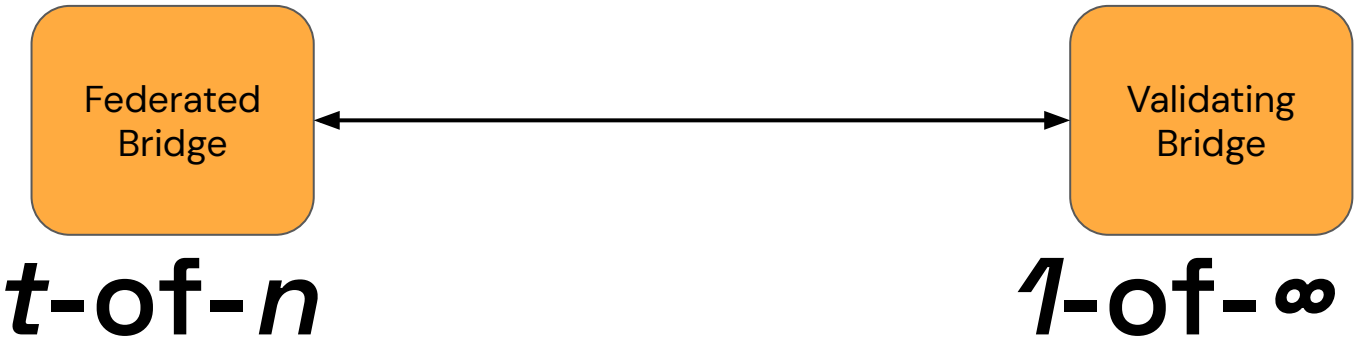
Federated Model (t-of-N)



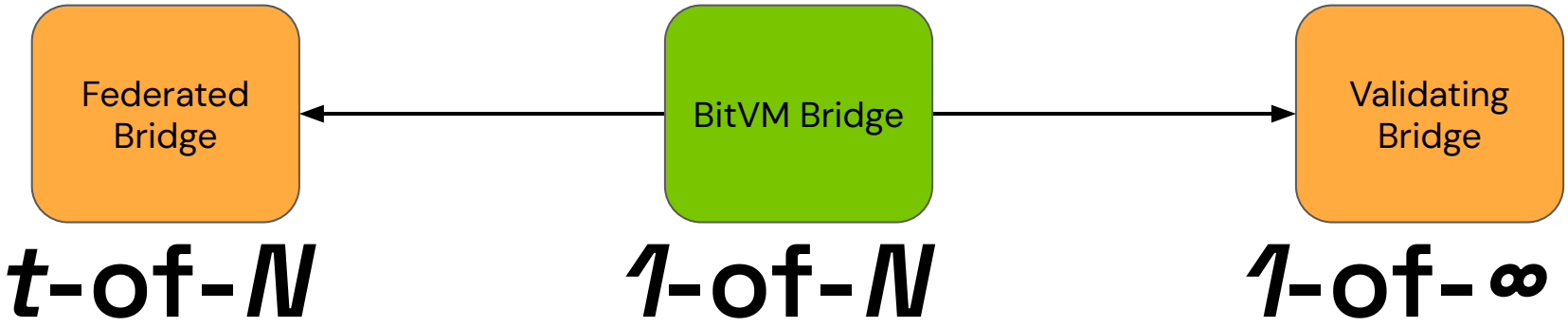
All federated bridges are based on the t-of-N security model. This model assumes that t members of a security committee of N are honest, in order to protect the funds. Normally, t represents the majority.

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

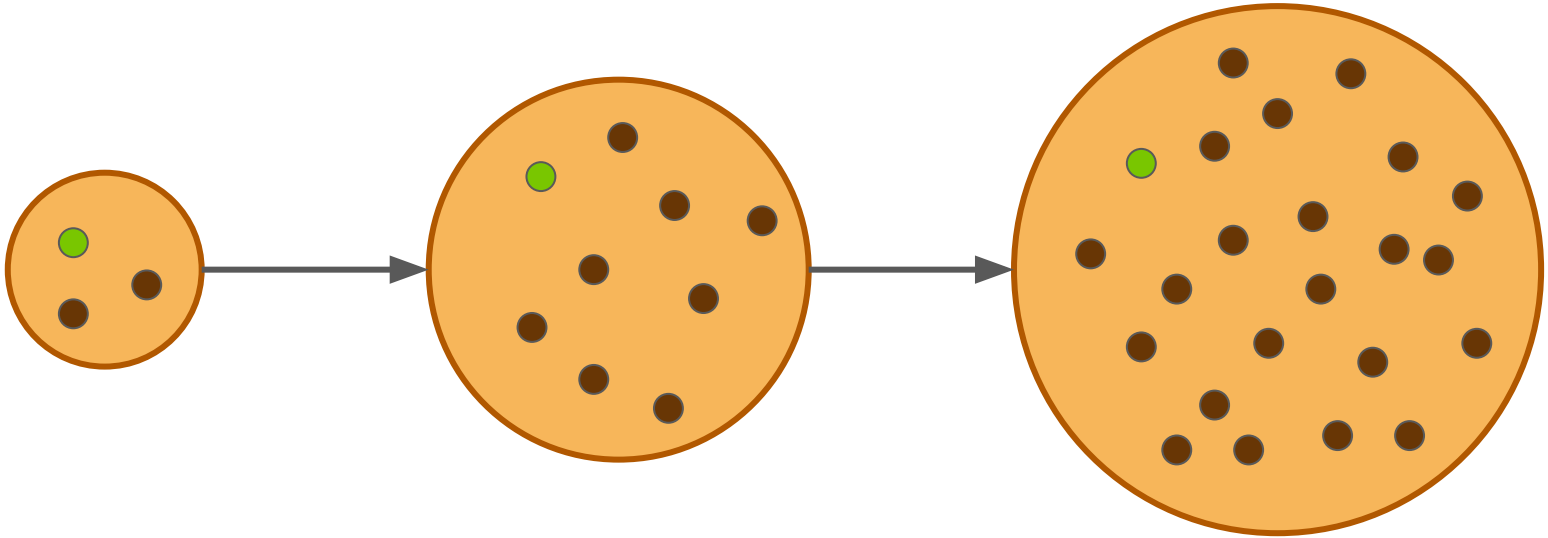
From *t*-of-*n* to *1*-of-∞



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



Openness Without Security Dilution



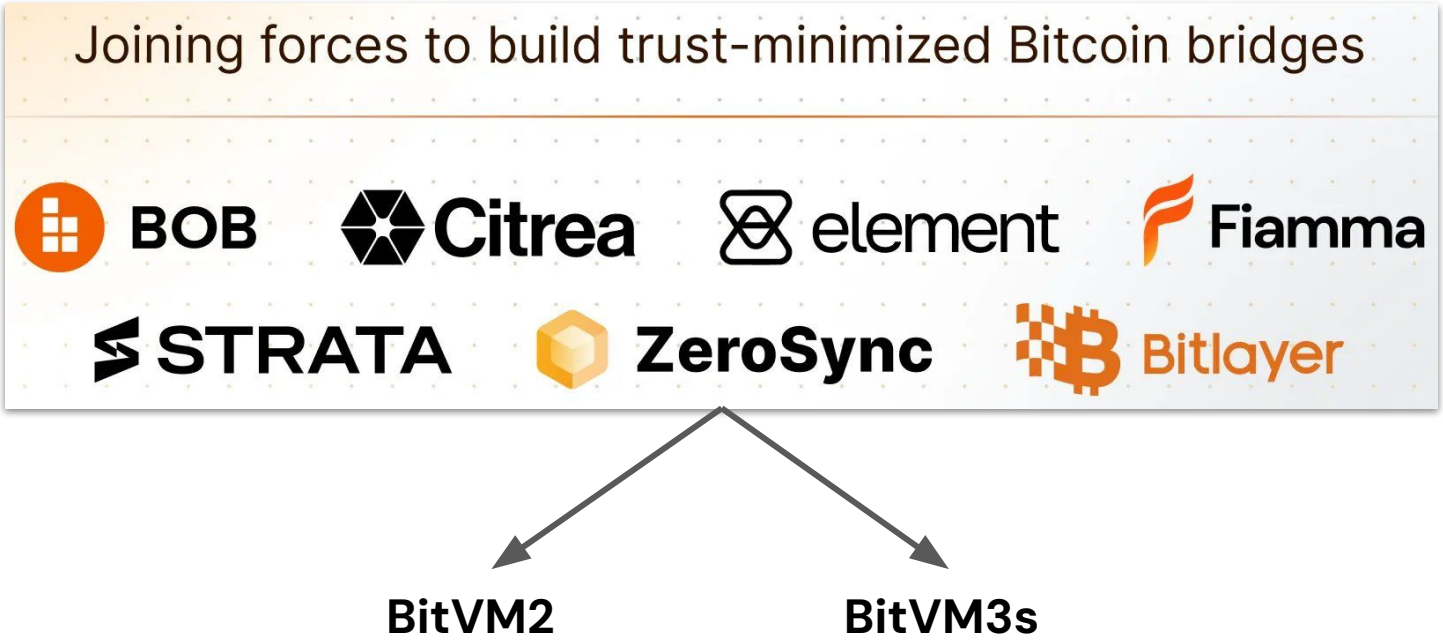
- Honest party
- Party of unknown behaviour

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 verifying computation on Bitcoin**.

Rootstock Labs

FairGate LABS

BitVMX FORCE

FOUNDING MEMBERS

Rootstock Labs

Fairgate

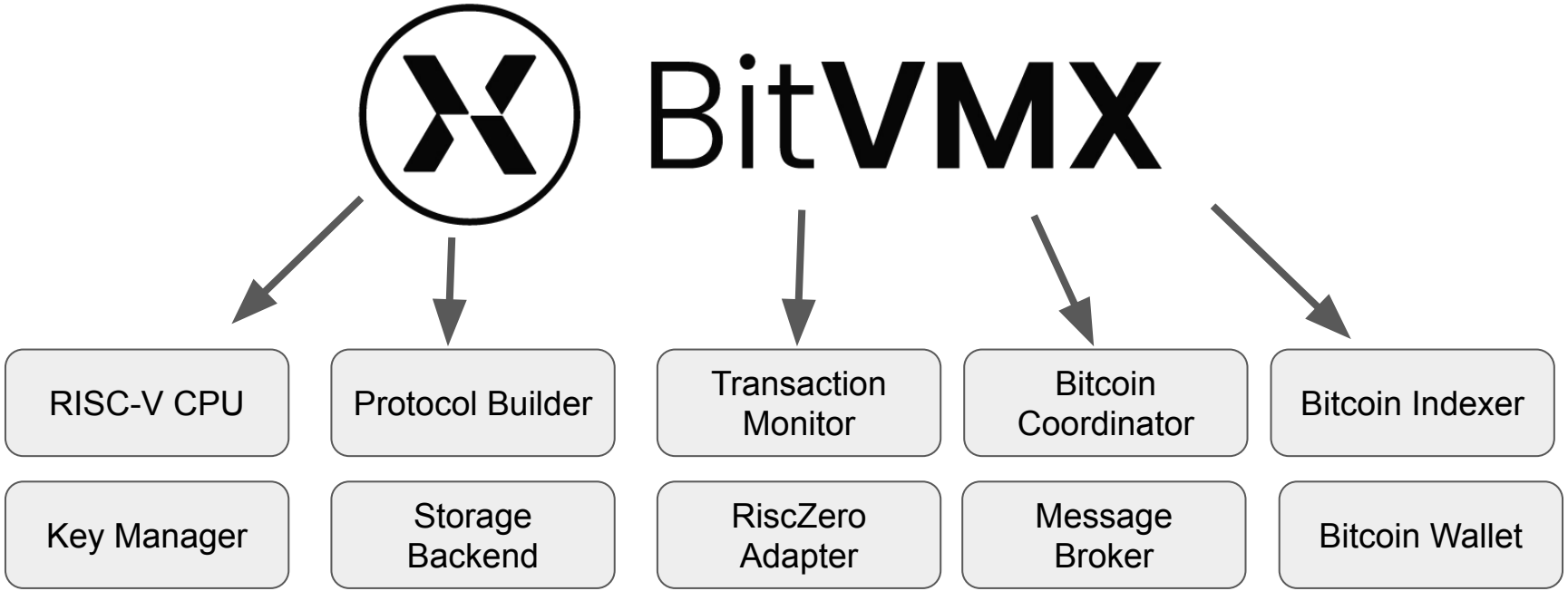
INPUT | OUTPUT

BitVMX-CPU

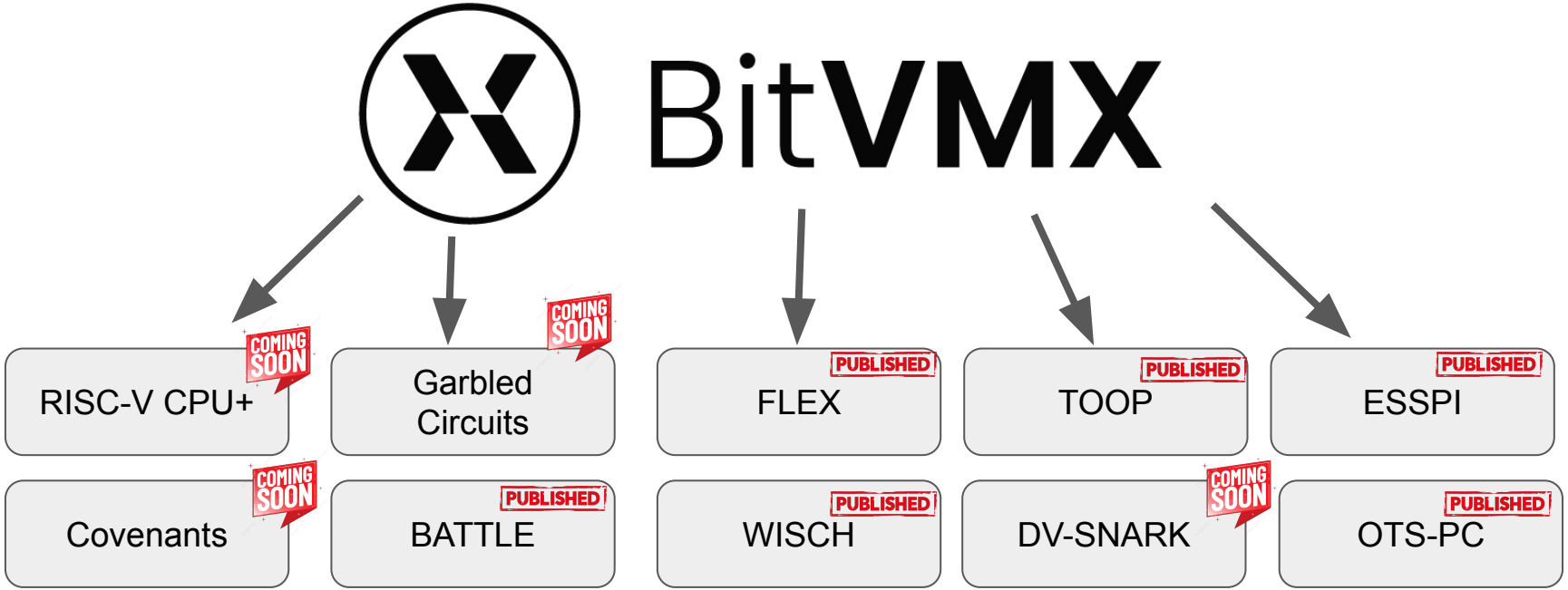
BitVMX Platform

BitVMX-GC

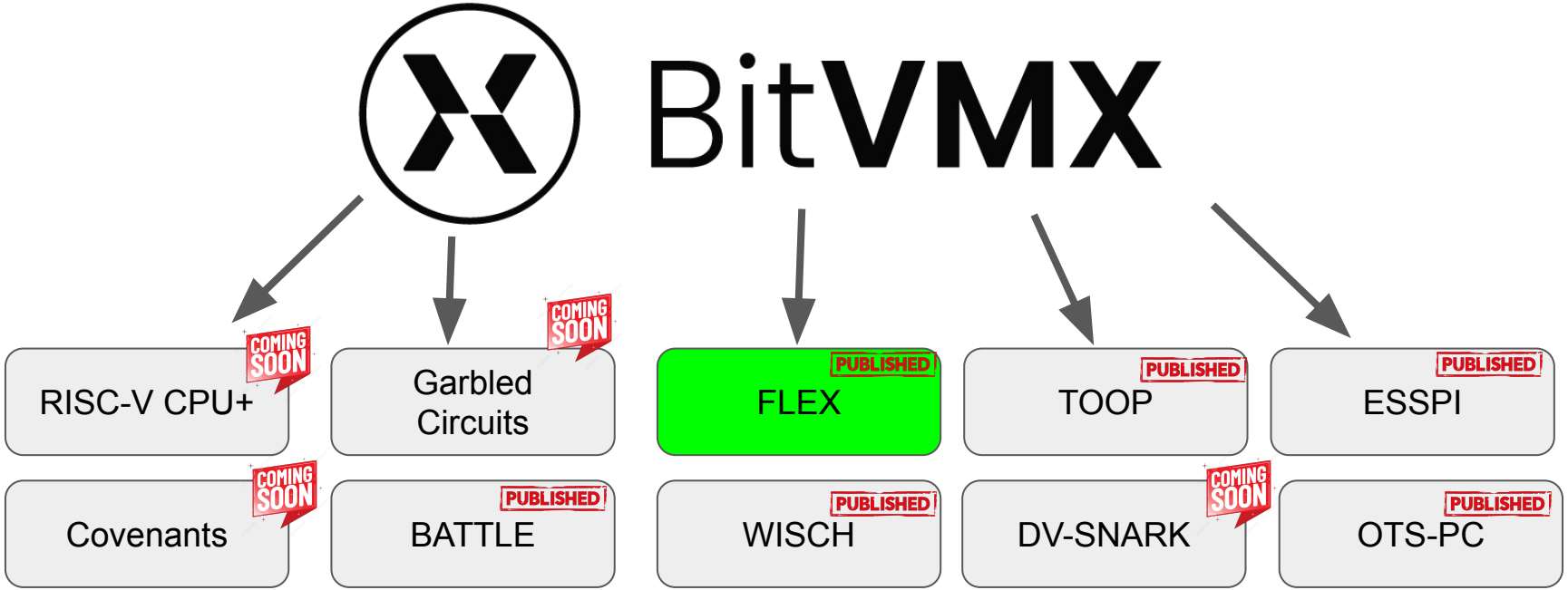
Current Components in BitVMX Platform



New Components Coming to the BitVMX Family 2026



New Components Coming to the BitVMX Family 2026



BitVMX FLEX

Fairgate

The role of security bonds on bitcoin protocols

1. Offsetting expected gains for a cheater in a probabilistic repeatable game
2. Reimburse dispute costs to honest parties (cheater pays)
3. To discourage griefing attacks.

The financial costs of security bonds

1. Tied to the opportunity cost
2. Grows exponential with respect to lock time (compound yield)
3. It will only increase with the growth of the bitcoin DeFi ecosystem.

Why BitVM Bridges use static bonds?

1. We need some form of security deposit
2. It’s simpler to use static bonds.
3. Emulated covenants “require” sourcing from fixed UTXOs to have known tx-ids for pre-signing.

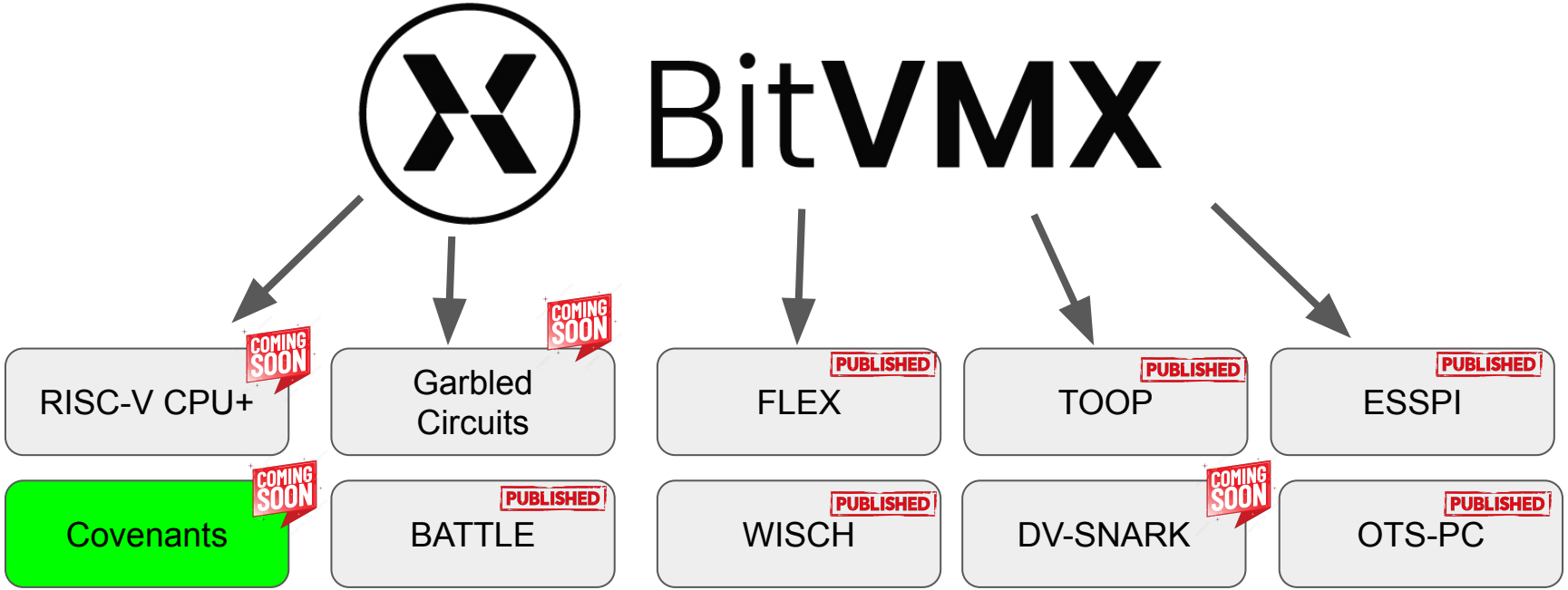
What’s wrong with static bonds?

1. **Duration**: Bonds stay for the life-cycle of the asset or role.
2. **Scalability**: Total bond requirements grows **quadratic** with the number of operators (in BitVM2, grows linearly)
3. **Bad Trade-offs**: Or, we can reduce bond requirements but affect security.
4. **Hard to build committees**: strong disincentive for becoming member
5. **High bonds impacts decentralization**

Solution: BitVMX **FLEX**

- Capital-Efficient Disputable computing protocols with On-Demand Security Bonds
- Builds on top of BitVMX Garbled Circuits
- Uses two GCs per dispute
- Properties:
  - Bond funding UTXOs are not required to be known during setup.
  - In a bridge, security deposits are required only during peg-out period for operators / watchtowers

New Components Coming to the BitVMX Family 2026



BitVMX **Covenants**

Fairgate

Use cases for Bitcoin Covenants

Covenants enable setting constraints on the transactions that spend funds. This can materially improve both scalability and security.

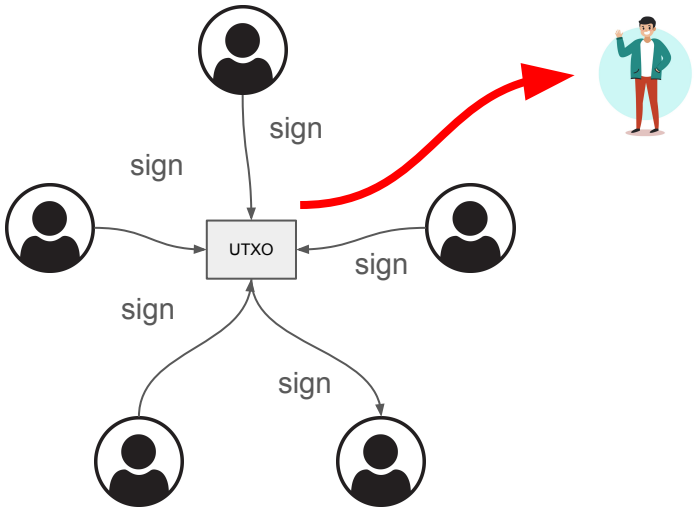
1. Turing complete contracts
2. Vaults with revocation keys
3. More decentralized L2 bridges
4. DAOs

Soft-Fork Proposals

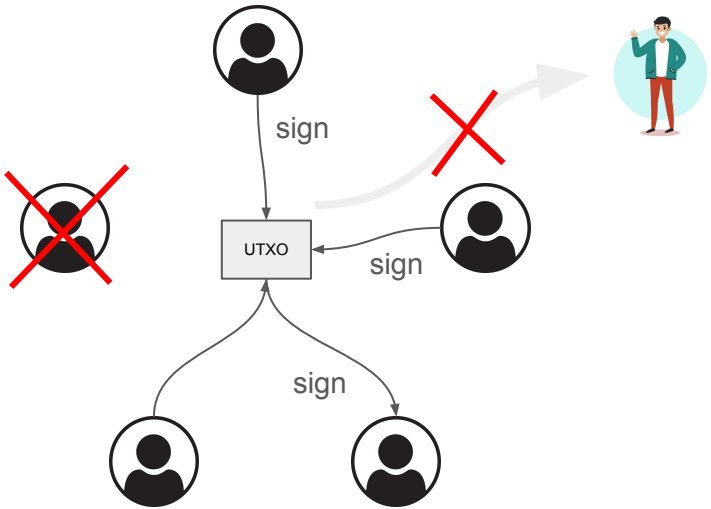
1. OP_CHECKOUTPUTVERIFY
2. OP_CHECKOUTPUTSHASHVERIFY
3. OP_CHECKTEMPLATEVERIFY
4. SIGHASH_NOINPUT/ANYPREVOUTS and privkey selection trick
5. OP_SUBSTR / OP_CAT and CHECKSIG trick

Bridging: N-of-N Multisig Paradigm

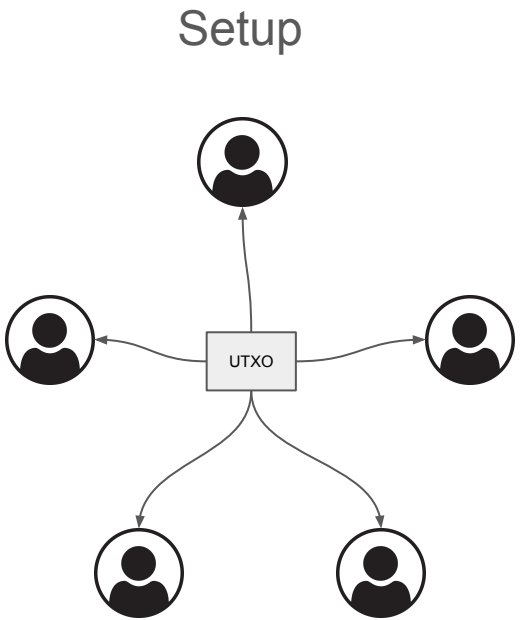
Pay to user (happy path)



Unhappy path ?

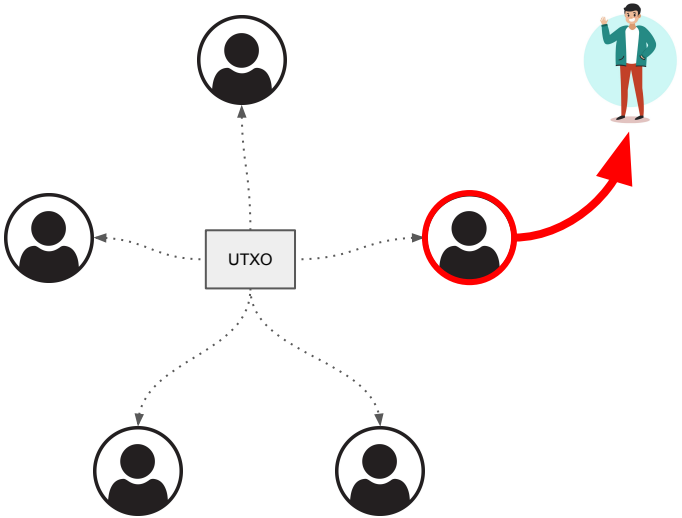


Current Fronting-Reimbursement Paradigm

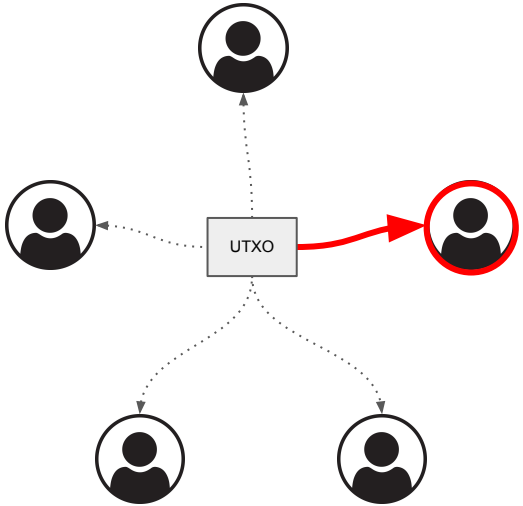


Current Fronting-Reimbursement Paradigm

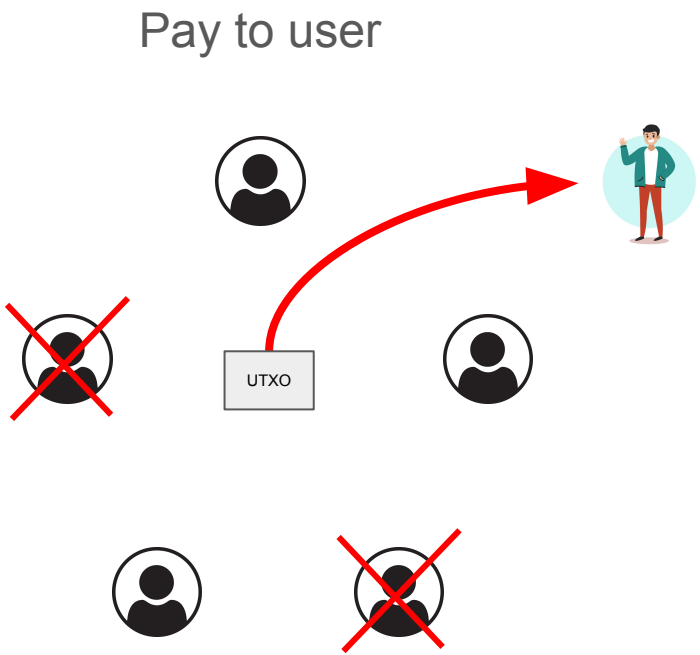
(1) Front



(2) Reimburse



Covenant Paradigm



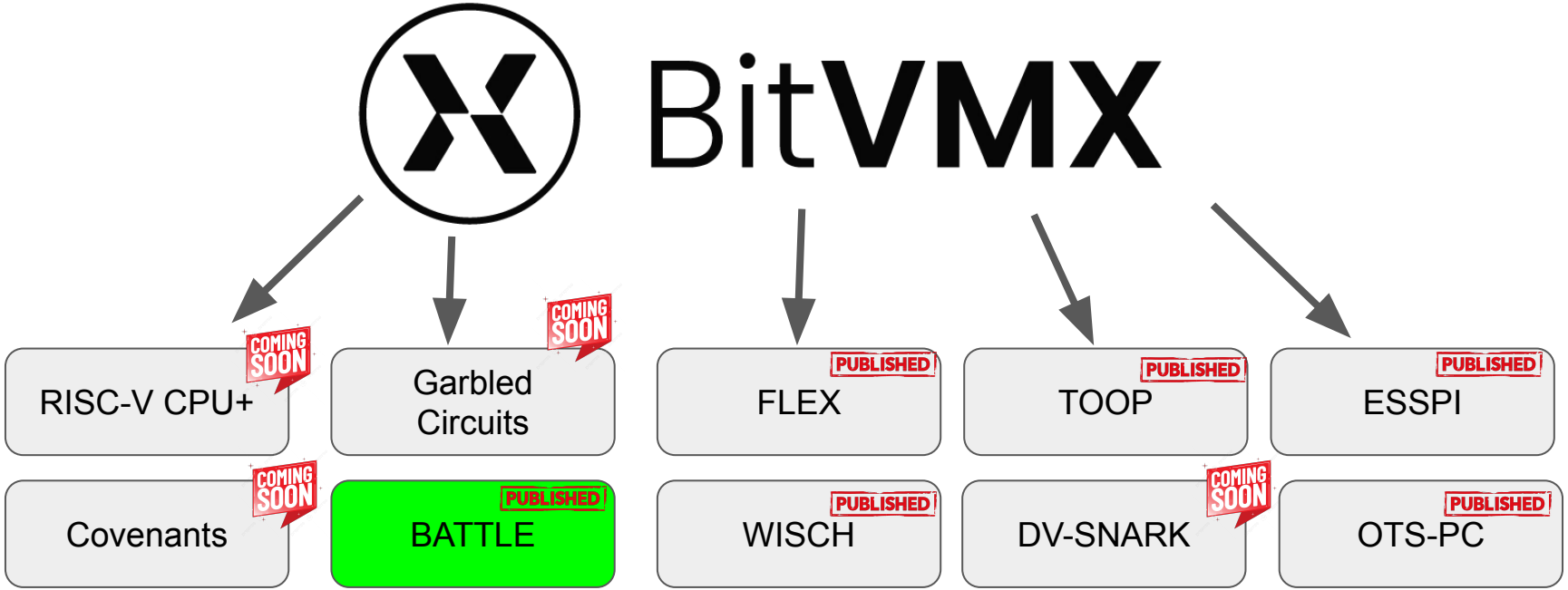
BitVMX Covenants

1. Allow a committee C of N parties controlling a UTXO U to dynamically build and execute a transaction T that consumes U, so that:
    a. T has specific properties (e.g. destination address, amount)
    b. T can be proven correct by an operator
    c. T has a change output that returns part of the funds to the committee
    d. If a set of parties $S \subsetneq C$ are unavailable, the protocol anyway executes T.
2. Secure if one of the N members is honest

Expected Covenants Use Cases

- Layer 2 Peg-outs without Fronting and Reimbursement of Funds
- Bitcoin DAOs (using recursive covenants)
    - Unused funds moved to a committee of available parties.

New Components Coming to the BitVMX Family 2026



BitVMX **BATTLE**

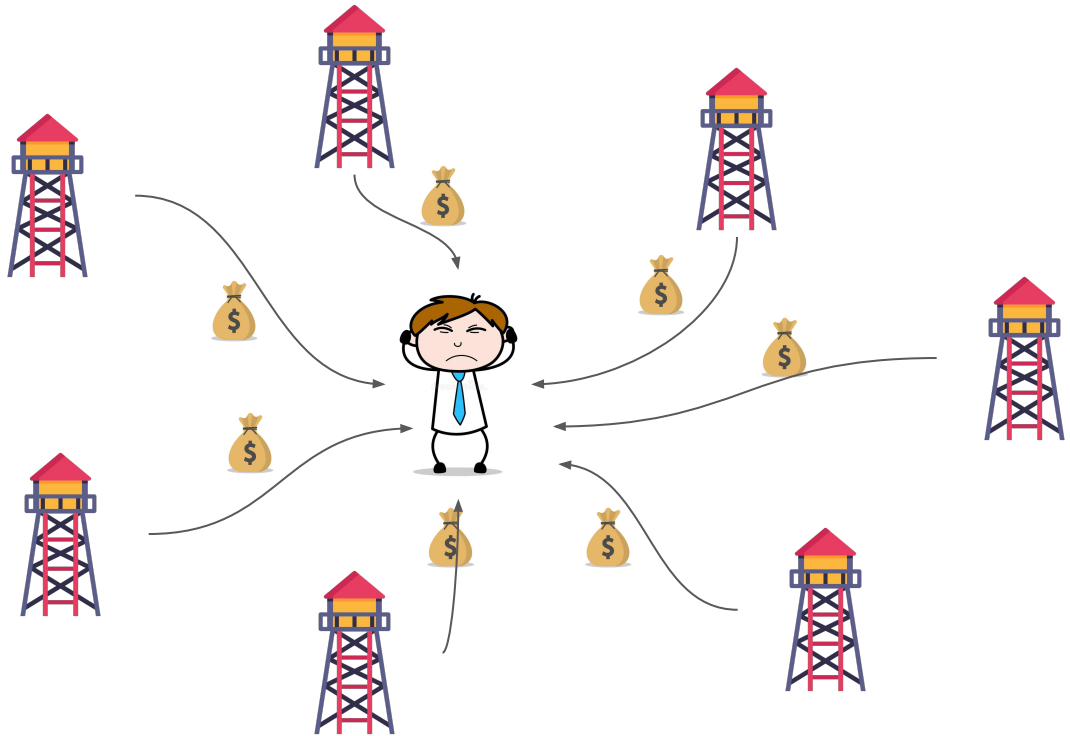
Fairgate

Problem

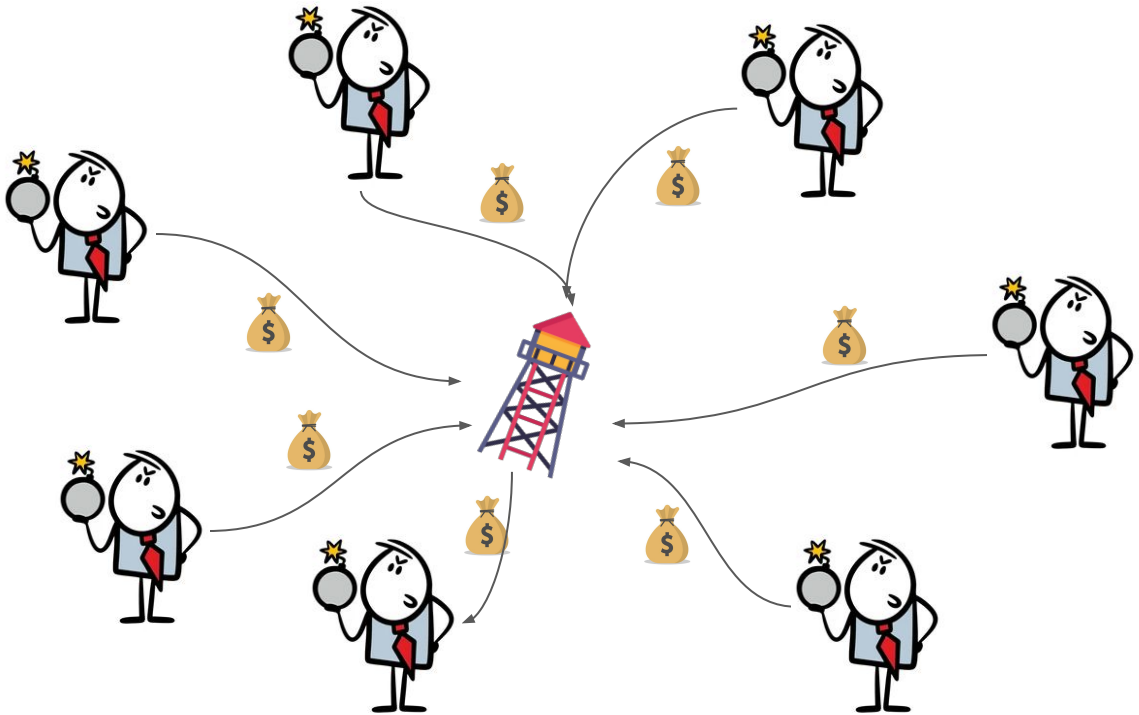
- Existent BitVM bridges that accept counter-proofs require:
  - (1) the pre-deposit of security bonds which grow with N, generally $O(N^2)$
  - (2) the pre-generation and signing of large transaction DAGs, whose size grows with N, generally $O(N^3)$.

- The larger the N, the more decentralized the bridge can be.
- Most BitVM-based protocols are restricted to $N < 20$

PROBLEM 1: Watchtowers against one Claimant



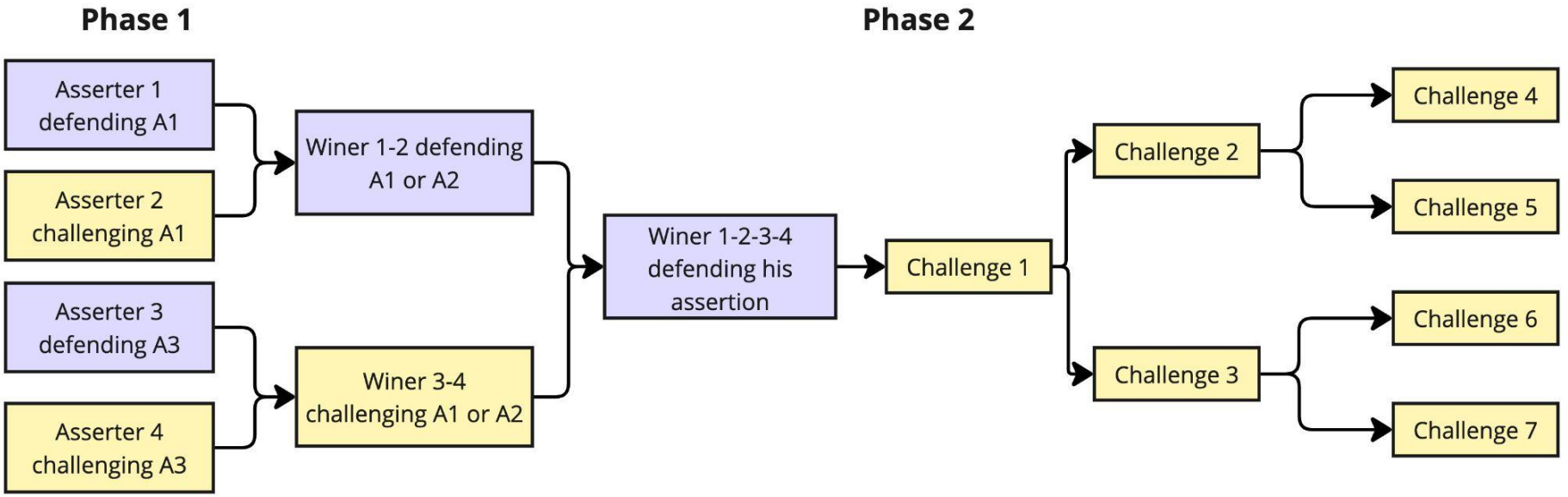
PROBLEM 2: Claimants against one Watchtower



Properties of BATTLE for Bitcoin

- Solves N-party Disputes in O(Log(N)) time
- Requires O(1) amount in security bonds
- Security bonds are only required on dispute with FLEX
- Each party performs $O(N^2)$ signatures
- Each party stores $O(N^2)$ data (partial TX DAG) even if DAG size is $O(N^3)$.
- Practical for $N \approx 1000$ operators
- GC table size is the main constraint

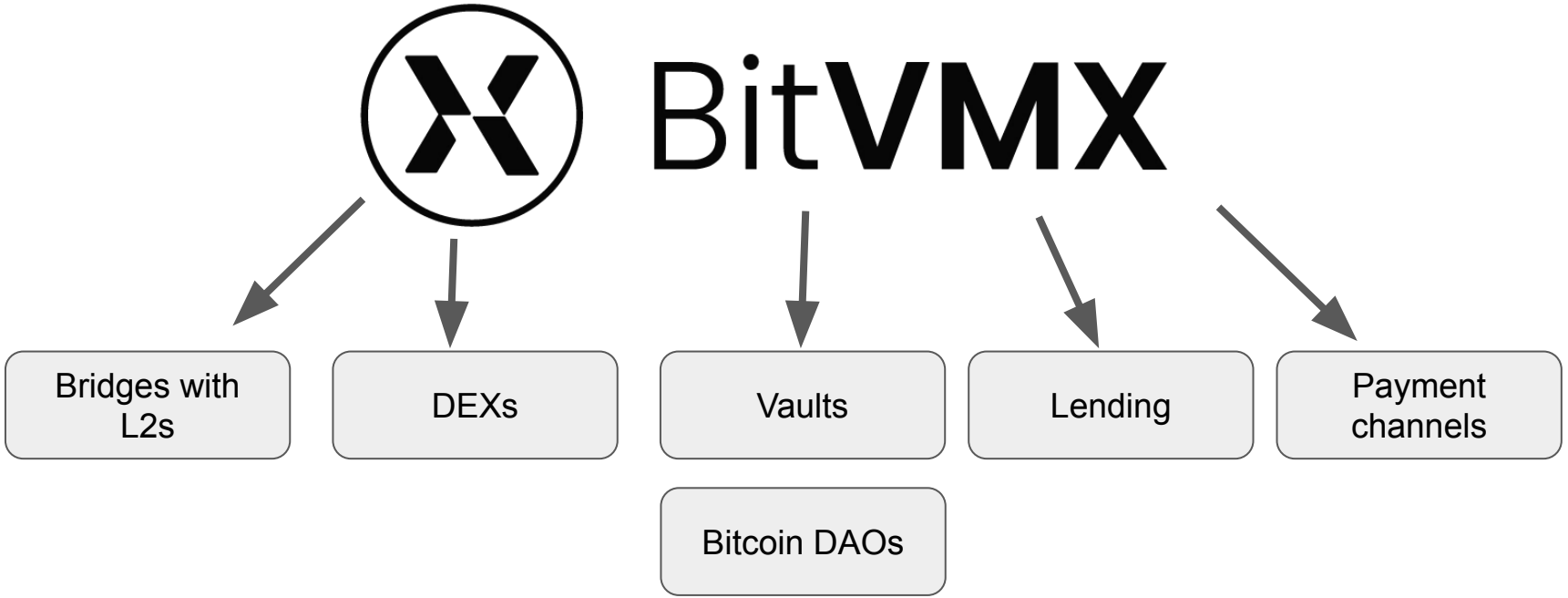
BATTLE Protocol Phases



Conflicting assertions compete.

Winning assertion competes with challengers

New Possible Applications



News on BitVM/BitVMX and L2s

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

Visit BitVMX.org and Fairgate.io !

Q&A