FAST, PRIVATE, FLEXIBLE BLOCKCHAIN CONTRACTS

ZkVM
ZkVM is a multi-asset blockchain architecture with contracts and confidentiality.

It is designed to scale, it is fast, and it’s written in pure Rust.

TL;DR

github.com/stellar/slingshot
AGENDA

1. Explain the good parts.
2. Explain away the bad parts.
ZkVM is a unique combination of the best ideas from Bitcoin devs. It is a preview of what Bitcoin may look like in the future.
ZkVM ARCHITECTURE
**Transactions**

A transaction (Tx) is a program that transfers assets from inputs to outputs. Transactions can also issue arbitrary assets.
UTREEXO
Based on original proposal by Thaddeus Dryja

- Blockchain state
- Unspent outputs
- Spent output
- Spending proof
Pros
• Storage is free: simplifies protocol.
• More nodes can be full nodes.

Cons
• Every node has to update their utxo proofs.
• Extra bandwidth overhead (negligible with caching).
Transaction is a program, cryptographic proof and some metadata.
VM instantiated per transaction; discarded after tx is processed.

High-level instructions enforce network rules.

Not turing-complete by design.
Instructions build a constraint system (CS) on the fly. CS enforces both network rules and custom, per-contract rules. Single aggregated proof is used to verify all the constraints.
Program Execution

Transaction verification is **stateless**.

Created/deleted outputs are recorded in the **transaction log**.

Transactions log is **applied** to the blockchain state separately.
Each unspent output is a **contract** object.

Contract has **arbitrary payload** (assets, data) protected by a **predicate**. Saved via **output** instruction, loaded via **input** instruction.
Predicate is satisfied with either a \textit{signature}...
Predicate is satisfied with either a **signature** or a **sub-program**.
TAPROOT

Compresses contract logic into a **single public key**.
Either **sign** with $K$, or **reveal a branch** and execute it.

$$P = K + \text{hash}(K, R) \cdot B$$

*Based on original proposal by Gregory Maxwell*
## INSTRUCTIONS

<table>
<thead>
<tr>
<th>Stack</th>
<th>Variables</th>
<th>Constraints</th>
<th>Values</th>
<th>Contracts</th>
</tr>
</thead>
<tbody>
<tr>
<td>push:n:x</td>
<td>const</td>
<td>neg</td>
<td>issue</td>
<td>input</td>
</tr>
<tr>
<td>program:n:x</td>
<td>var</td>
<td>add</td>
<td>borrow</td>
<td>output:k</td>
</tr>
<tr>
<td>drop</td>
<td>alloc</td>
<td>mul</td>
<td>retire</td>
<td>contract:k</td>
</tr>
<tr>
<td>dup:k</td>
<td>mintime</td>
<td>eq</td>
<td>cloak:m:n</td>
<td>log</td>
</tr>
<tr>
<td>roll:k</td>
<td>maxtime</td>
<td>range:n</td>
<td></td>
<td>call</td>
</tr>
<tr>
<td></td>
<td>unblind</td>
<td>and</td>
<td></td>
<td>signtx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or</td>
<td></td>
<td>signid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>not</td>
<td></td>
<td>signtag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>verify</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### INSTRUCTIONS

<table>
<thead>
<tr>
<th>Stack</th>
<th>Variables</th>
<th>Constraints</th>
<th>Values</th>
<th>Contracts</th>
</tr>
</thead>
<tbody>
<tr>
<td>push:n:x</td>
<td>const</td>
<td>neg</td>
<td>issue</td>
<td>input</td>
</tr>
<tr>
<td>program:n:x</td>
<td>var</td>
<td>add</td>
<td>borrow</td>
<td>output:k</td>
</tr>
<tr>
<td>drop</td>
<td>alloc</td>
<td>mul</td>
<td>retire</td>
<td>contract:k</td>
</tr>
<tr>
<td>dup:k</td>
<td>mintime</td>
<td>eq</td>
<td>cloak:m:n</td>
<td>log</td>
</tr>
<tr>
<td>roll:k</td>
<td>maxtime</td>
<td>range:n</td>
<td></td>
<td>call</td>
</tr>
<tr>
<td></td>
<td>unblind</td>
<td>and</td>
<td></td>
<td>signtx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or</td>
<td></td>
<td>signid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>not</td>
<td></td>
<td>signtag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>verify</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bitcoin:** 88  
**Ethereum:** 77  
**Miniscript:** 26  
**ZkVM:** 33 instructions
Curve25519-Dalek  Vectorized elliptic curve operations.
Ristretto255  Safe prime order group.

Curve25519-Dalek  Vectorized elliptic curve operations.
Bulletproofs  Versatile zero-knowledge proof system.
Ristretto255  Safe prime order group.
Curve25519-Dalek  Vectorized elliptic curve operations.
# Cryptography Stack

- **Cloak**: Network rules.
- **Bulletproofs**: Versatile zero-knowledge proof system.
- **Ristretto255**: Safe prime order group.
- **Curve25519-Dalek**: Vectorized elliptic curve operations.
CRYPTOGRAPHY STACK

Cloak  Constraints  Network rules + custom rules.
Bulletproofs  Versatile zero-knowledge proof system.
Ristretto255  Safe prime order group.
Curve25519-Dalek  Vectorized elliptic curve operations.
CRYPTOGRAPHY STACK

**Instructions**  
Arithmetic + boolean operations.

**Cloak**  
Network rules + custom rules.

**Bulletproofs**  
Versatile zero-knowledge proof system.

**Ristretto255**  
Safe prime order group.

**Curve25519-Dalek**  
Vectorized elliptic curve operations.
CRYPTOGRAPHY STACK

Your protocol

Instructions

Cloak

Bulletproofs

Ristretto255

Curve25519-Dalek

Vaults, payment channels, order books, ...

Arithmetic + boolean operations.

Network rules + custom rules.

Versatile zero-knowledge proof system.

Safe prime order group.

Vectorized elliptic curve operations.
CRYPTOGRAPHY STACK

Your protocol

Instructions

Cloak Constraints

Bulletproofs

Ristretto255

Curve25519-Dalek

Vaults, payment channels, order books, ...

Arithmetic + boolean operations.

Network rules + custom rules.

Versatile zero-knowledge proof system.

Safe prime order group.

Vectorized elliptic curve operations.
CONSTRAINTS

Ad-hoc composition of arithmetic and boolean expressions:

\[
(P = B + R \cdot T) \text{ OR } (X = Y)
\]

R T \text{ mul } B \text{ add } P \text{ eq } X \text{ Y eq or verify}
Create variables from commitments, make expressions, form constraints and add them to the constraint system.
A variable defines a payment constraint with borrow + output.

Negative value is mixed with an actual payment in the cloak.
In ZkVM contracts imperatively express their requirements, entirely avoiding bugs like confused deputy problem.
ZkVM TRADEOFFS
ZkVM optimized for **financial uses**, not arbitrary computations:

- issuing tokens and fundraising,
- multi-party vaults,
- derivative instruments,
- payment channels.
Only zkSNARKs allow efficient compression (e.g. Coda).
SPV clients use $\approx 50x$ less traffic than full blocks.
Bootstrap from trusted source via Utreexo roots.
PRIVACY FEATURES

Private
Asset types
Asset quantities
Data parameters
In-transaction flow

Not private
Programs
Transaction graph
Privacy Features

Private
- Asset types
- Asset quantities
- Data parameters
- In-transaction flow

Not private
- Programs
- Transaction graph

Taproot reveals only in dispute and only a specific branch.
PRIVACY FEATURES

Private
- Asset types
- Asset quantities
- Data parameters
- In-transaction flow

Not private
- Programs
- Transaction graph

Taproot reveals only in dispute and only a specific branch.

CoinJoin scales better. Hiding UTXO links requires $O(n)$ storage (nullifiers).
PERFORMANCE

1 Fast  

<1 ms per output (up to 1000 tx/sec).

- vectorized implementation of Curve25519,
- signature aggregation,
- state of the art multi-scalar multiplication,
- \( \approx 1.5 \text{ Kb/proof}, \text{ marginal cost 0.2–0.5 Kb/transfer}. \)
PERFORMANCE

1. Fast  
   <1 ms per output (up to 1000 tx/sec).

2. Always fast  
   Custom constraints are relatively cheap.
   - rangeproofs for output values bear most of the cost,
   - signatures and custom constraints: 1-5% overhead.
PERFORMANCE

1. Fast  <1 ms per output (up to 1000 tx/sec).

2. Always fast  Custom constraints are relatively cheap.

3. Scales with privacy  Aggregation saves space and time.
   - proof size is $\log(N)$, marginal cost goes to zero,
   - larger batches of ECC operations take $N/\log(N)$ time.
PERFORMANCE

1. Fast < 1 ms per output (up to 1000 tx/sec).

2. Always fast Custom constraints are relatively cheap.

3. Scales with privacy Aggregation saves space and time.

4. Free storage Utreexo makes storage costs negligible.
   - storage costs $\log(N)$ (≈1 kilobyte without caching),
   - bandwidth overhead is 5-10% with caching (+ tens of megabytes)
SMALL AND SAFE

Small, pure-Rust codebase:

- 6K LOC   zkvm + utreexo + blockchain (w/o consensus)
- 7K LOC   schnorr + musig + keytree + bulletproofs
- 14K LOC  curve25519 + ristretto255

Assumptions:

- ECDLP on Curve25519
- Keccak (SHAKE128) is a random oracle
LEARN MORE & PARTICIPATE

Code and specs:
github.com/stellar/slingshot

See also:
github.com/dalek-cryptography/bulletproofs
ristretto.group
merlin.cool
THANK YOU

Oleg Andreev
@oleganza

Project Slingshot is sponsored by Stellar Development Foundation.