# FAST, PRIVATE, FLEXIBLE BLOCKCHAIN CONTRACTS

# ZkVM

*Oleg Andreev*
*Tel Aviv, Israel*
*September 11-12, 2019*

# TL;DR

*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**.*

*github.com/stellar/slingshot*

# AGENDA

1. *Explain the good parts.*
2. *Explain away the bad parts.*

# IS IT ABOUT BITCOIN?

*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.*

*bitcoin*

*zcash* *ethereum*

***payment channels*** ***musig***

*mimblewimble* *monero* ***coinjoin***

*bls signatures* ***linear types*** *txo mmr*

***object capabilities*** *ring signatures*

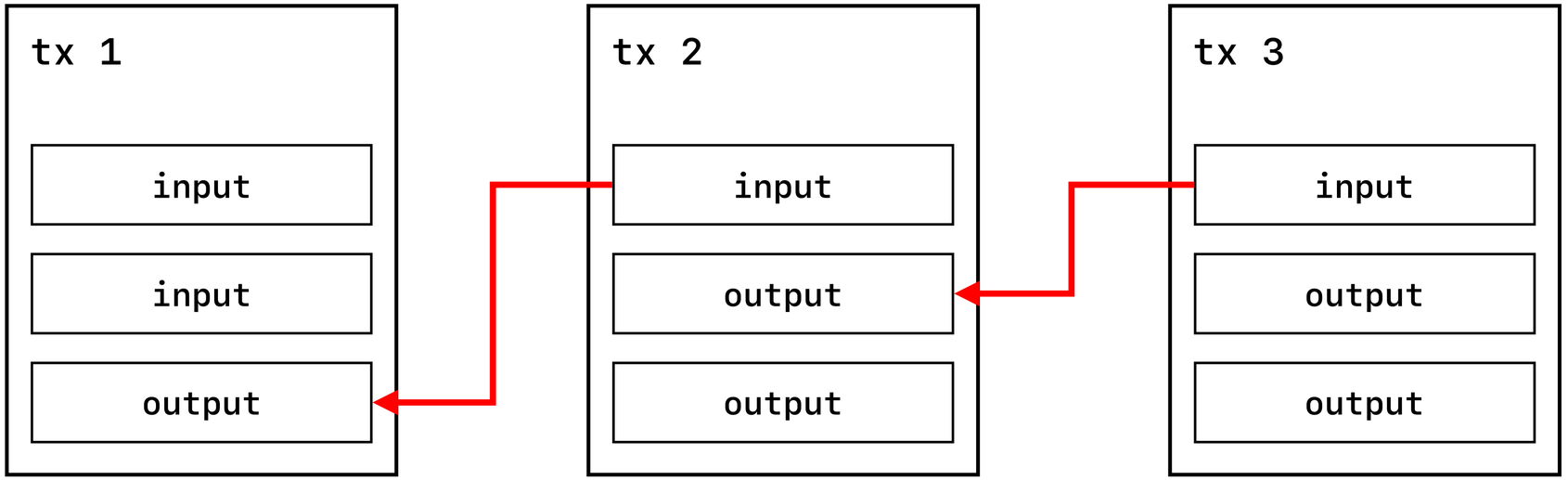
*zksnarks* ***utreexo*** ***bulletproofs***

*recursive snarks* ***taproot***

***ristretto***

# ZkVM ARCHITECTURE

# TRANSACTIONS

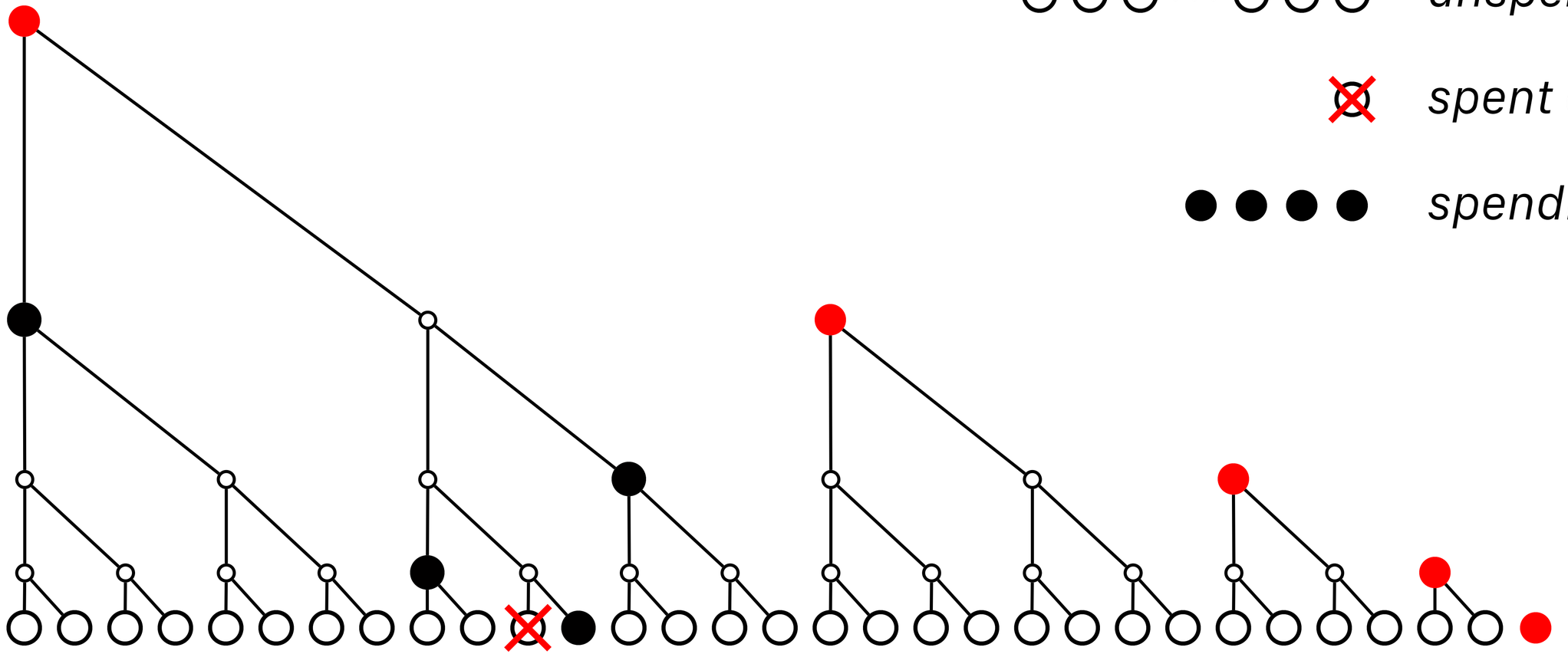



*Tx = program that transfers assets from **inputs** to **outputs**.*

*Transactions can also **issue** arbitrary assets.*

# UTREEXO

*based on original proposal by Thaddeus Dryja*



*blockchain state*

*unspent outputs*

*spent output*

*spending proof*

# UTREEXO

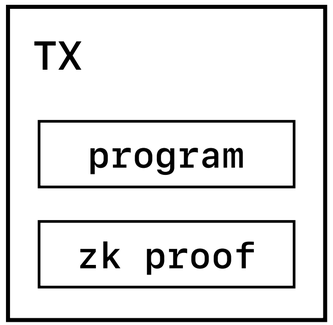
***Pros***

- *Storage is free: simplifies protocol.*
- *More nodes can be full nodes.*

***Cons***

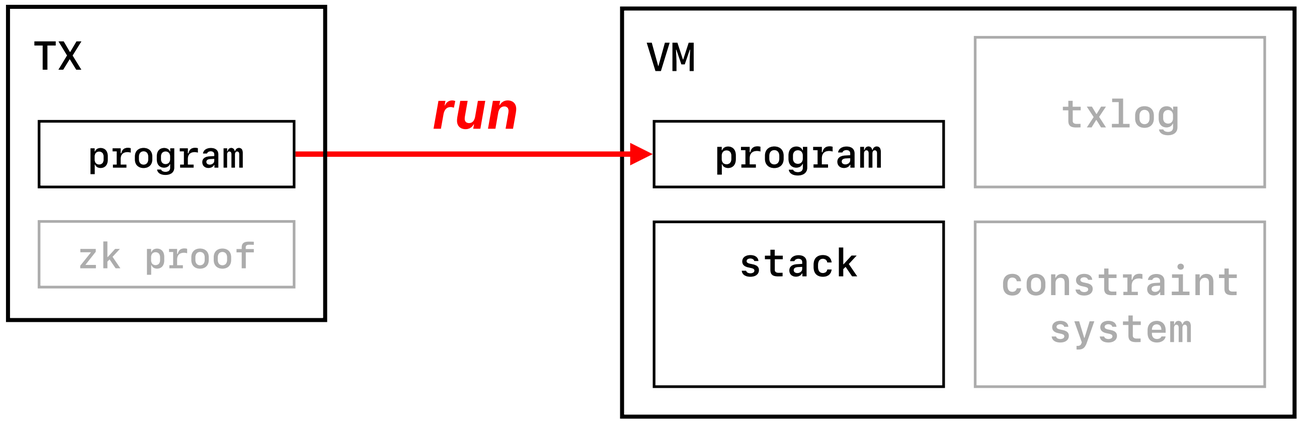
- *Every node has to update their utxo proofs.*
- *Extra bandwidth overhead (negligible with caching).*

# PROGRAM EXECUTION




*Transaction is a **program**, cryptographic **proof** and some metadata.*

# PROGRAM EXECUTION

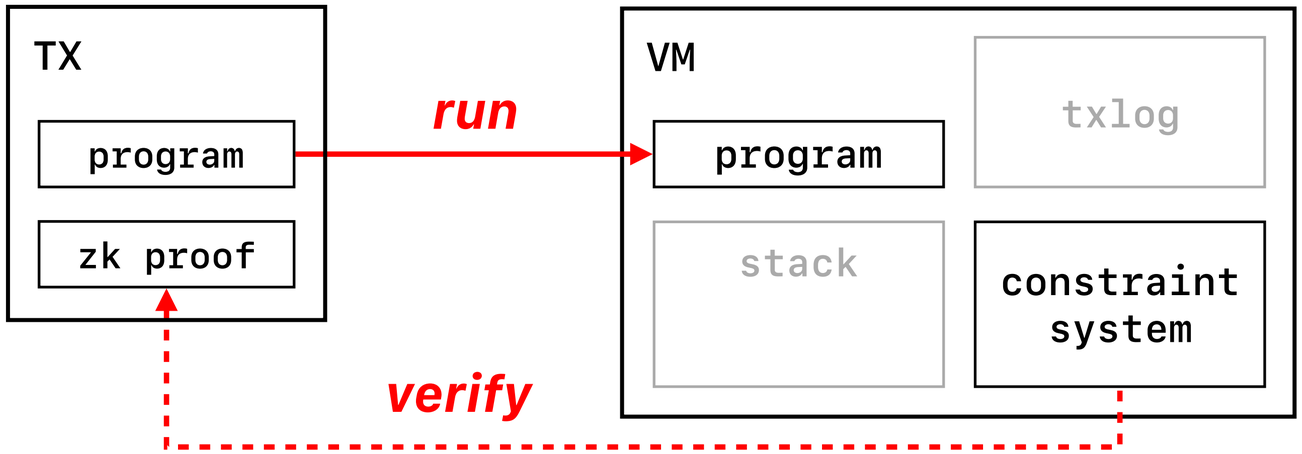



*VM instantiated **per transaction**; discarded after tx is processed.*

*High-level instructions **enforce network rules**.*

***Not turing-complete** by design.*

# PROGRAM EXECUTION

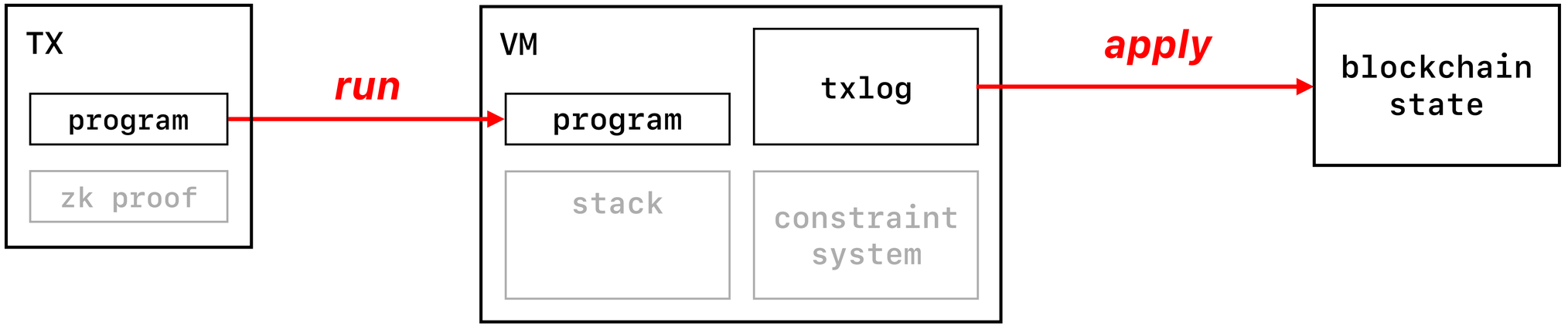



*Instructions build a constraint system (CS)* ***on the fly****.*

*CS enforces both* ***network rules*** *and custom,* ***per-contract rules****.*

***Single aggregated proof*** *is used to verify all the constraints.*

# PROGRAM EXECUTION

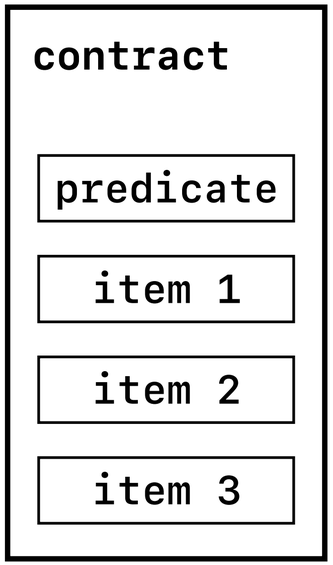



*Transaction verification is* ***stateless.***

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

*Transactions log is* ***applied*** *to the blockchain state separately.*

# CONTRACTS

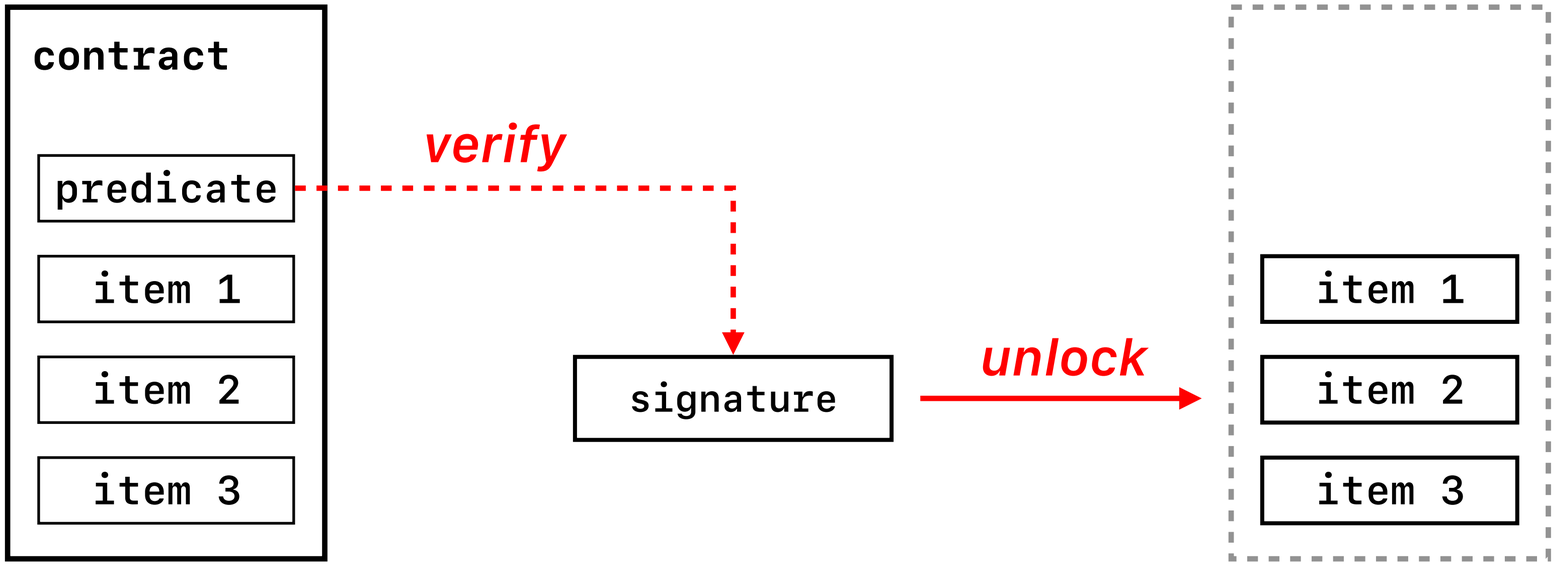



*Each unspent output is a **contract** object.*

*Contract has **arbitrary payload** (assets, data) protected by a **predicate**.*

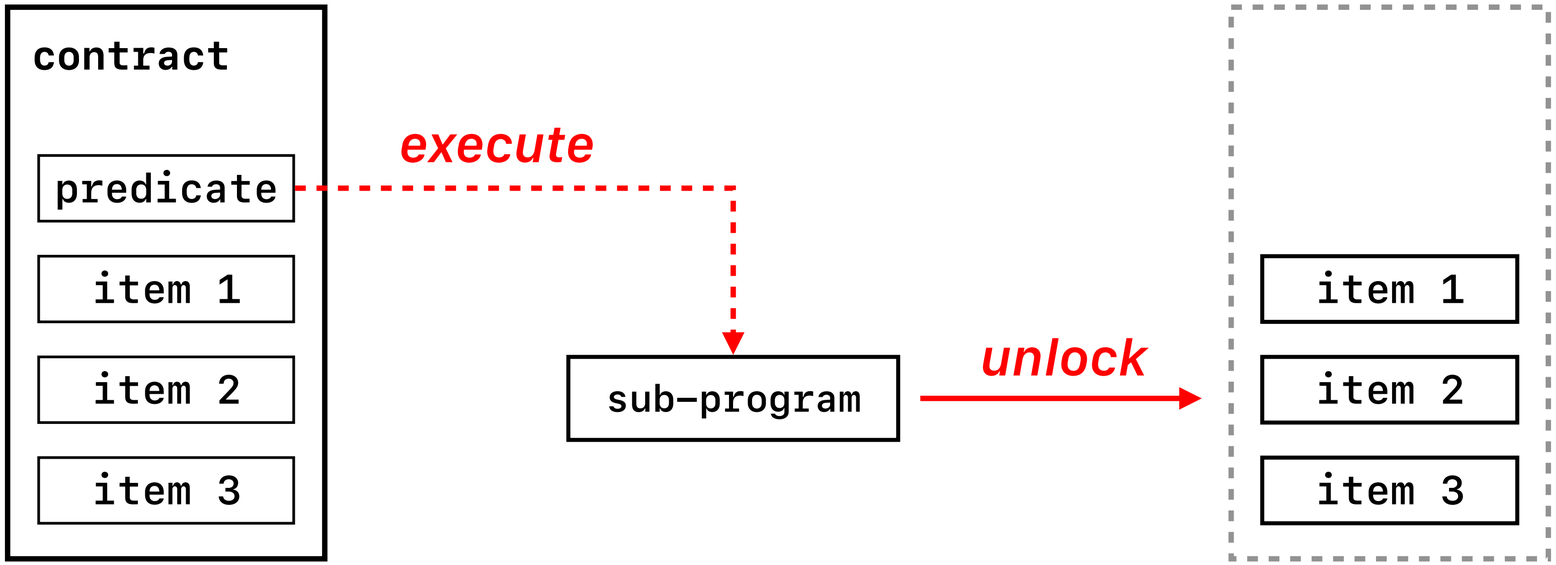
*Saved via* `output` *instruction, loaded via* `input` *instruction.*

# CONTRACTS




*Predicate is satisfied with either a **signature**...*

# CONTRACTS



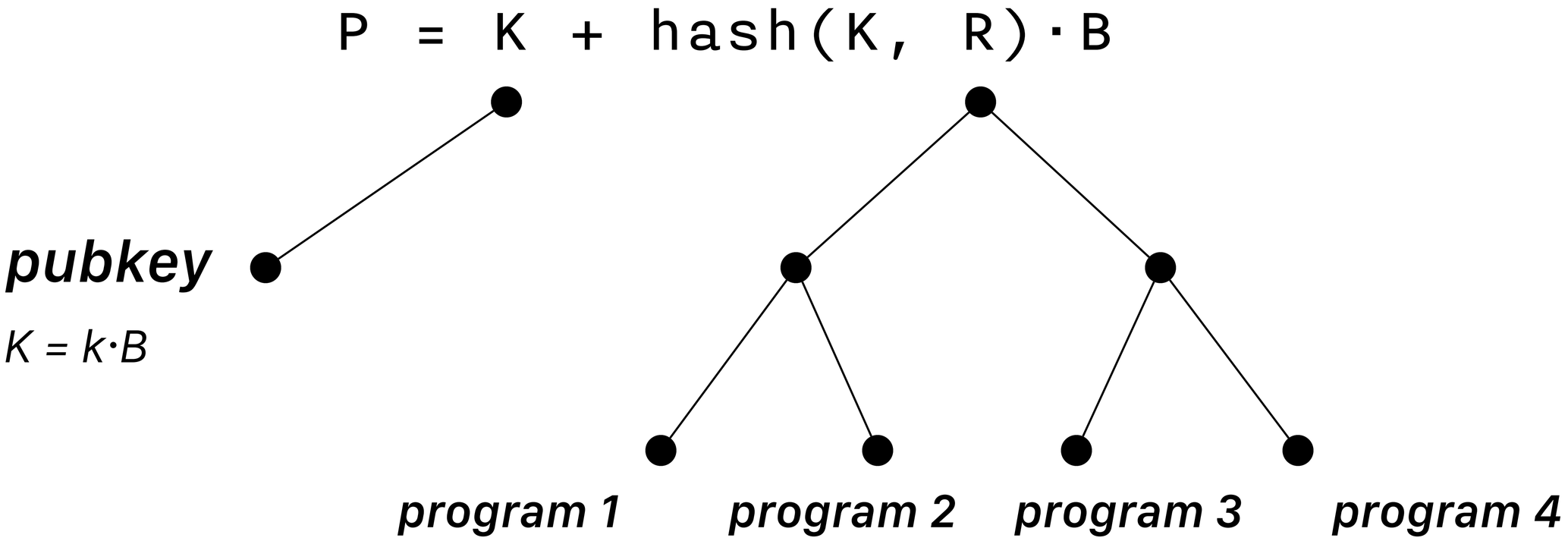

*Predicate is satisfied with either a **signature** or a **sub-program**.*

# TAPROOT

*based on original proposal by Gregory Maxwell*

*Compresses contract logic into a **single public key**.*

*Either **sign** with K, or **reveal a branch** and execute it.*



# INSTRUCTIONS

## Stack

```
push:n:x
program:n:x
drop
dup:k
roll:k
```

## Variables

```
const
var
alloc
mintime
maxtime
unblind
```

## Constraints

```
neg
add
mul
eq
range:n
and
or
not
verify
```

## Values

```
issue
borrow
retire
cloak:m:n
```

## Contracts

```
input
output:k
contract:k
log
call
signtx
signid
signtag
```

# INSTRUCTIONS

| *Stack* | *Variables* | *Constraints* | *Values* | *Contracts* |
|---|---|---|---|---|
| `push:n:x` | `const` | `neg` | `issue` | `input` |
| `program:n:x` | `var` | `add` | `borrow` | `output:k` |
| `drop` | `alloc` | `mul` | `retire` | `contract:k` |
| `dup:k` | `mintime` | `eq` | `cloak:m:n` | `log` |
| `roll:k` | `maxtime` | `range:n` | | `call` |
| | `unblind` | `and` | | `signtx` |
| | | `or` | | `signid` |
| | | `not` | | `signtag` |
| | | `verify` | | |

---

*Bitcoin: 88* *Ethereum: 77* *Miniscript: 26* *ZkVM: **33** instructions*

# CRYPTOGRAPHY STACK

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

# CRYPTOGRAPHY STACK

| | |
|---|---|
| *Ristretto255* | *Safe prime order group.* |
| *Curve25519-Dalek* | *Vectorized elliptic curve operations.* |

# CRYPTOGRAPHY STACK

| | |
|---|---|
| *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.* |

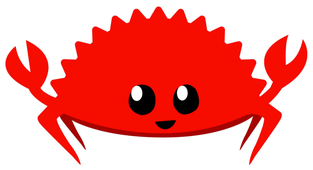
# CRYPTOGRAPHY STACK

| | |
|---|---|
| *Instructions* | *Arithmetic + boolean operations.* |
| *Cloak Constraints* | *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* | *Vaults, payment channels, order books, ...* |
| *Instructions* | *Arithmetic + boolean operations.* |
| *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



pure Rust

| | |
|---|---|
| *Your protocol* | *Vaults, payment channels, order books, ...* |
| *Instructions* | *Arithmetic + boolean operations.* |
| *Cloak Constraints* | *Network rules + custom rules.* |
| *Bulletproofs* | *Versatile zero-knowledge proof system.* |
| *Ristretto255* | *Safe prime order group.* |
| *Curve25519-Dalek* | *Vectorized elliptic curve operations.* |

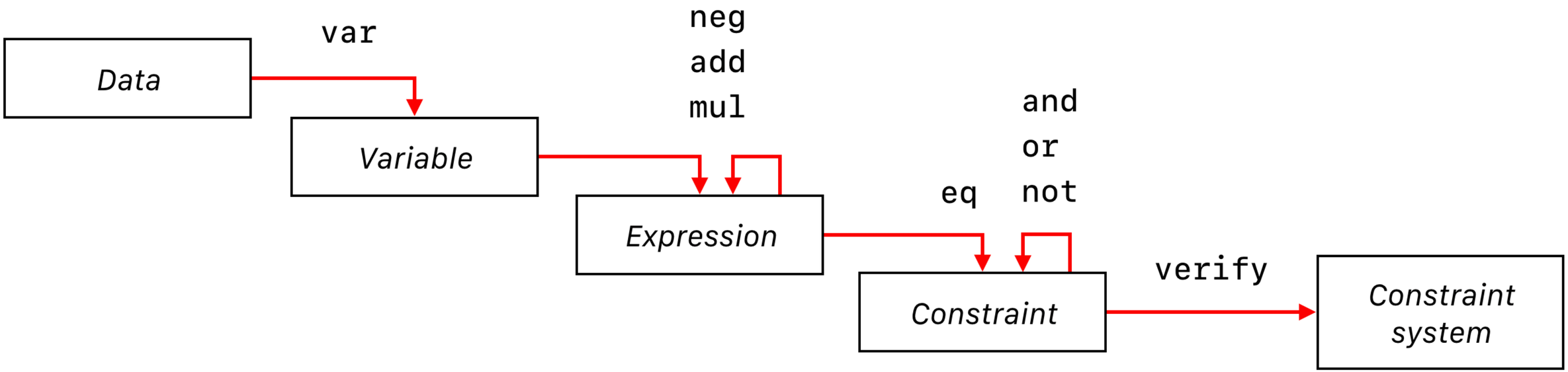
# CONSTRAINTS

*Ad-hoc composition of **arithmetic** and **boolean** expressions:*

```
(P = B + R·T) OR (X = Y)
```

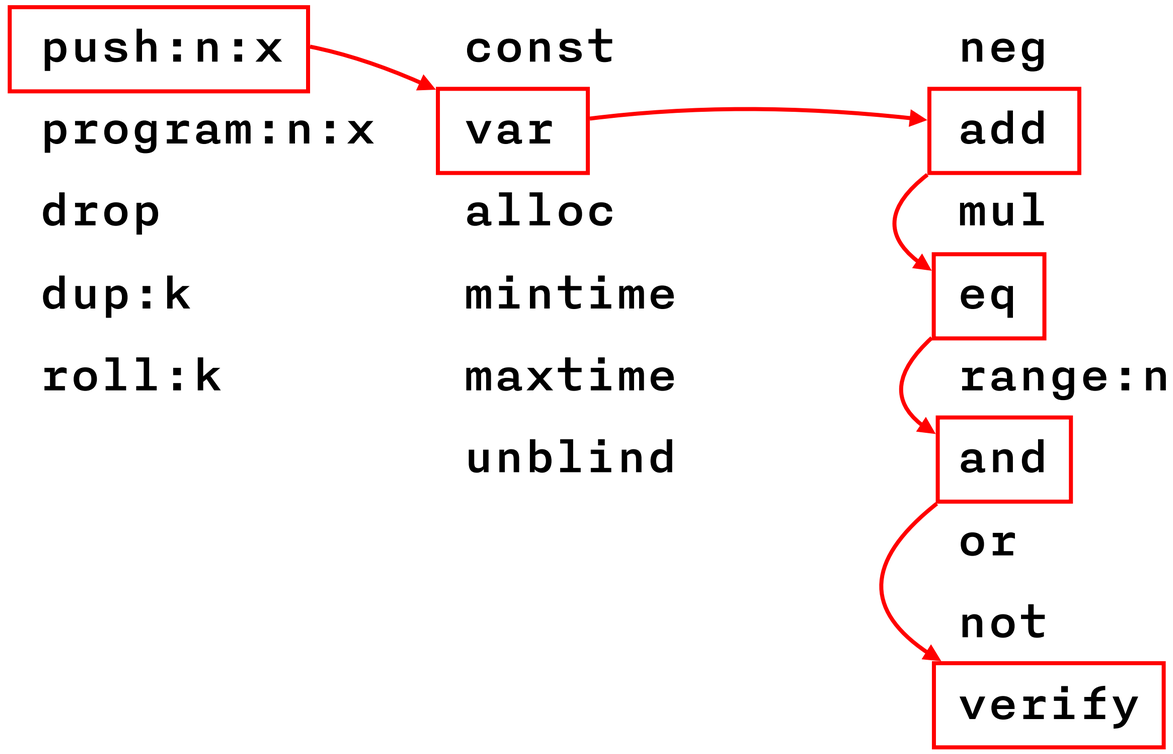
```
R T mul B add P eq X Y eq or verify
```

# CONSTRAINTS



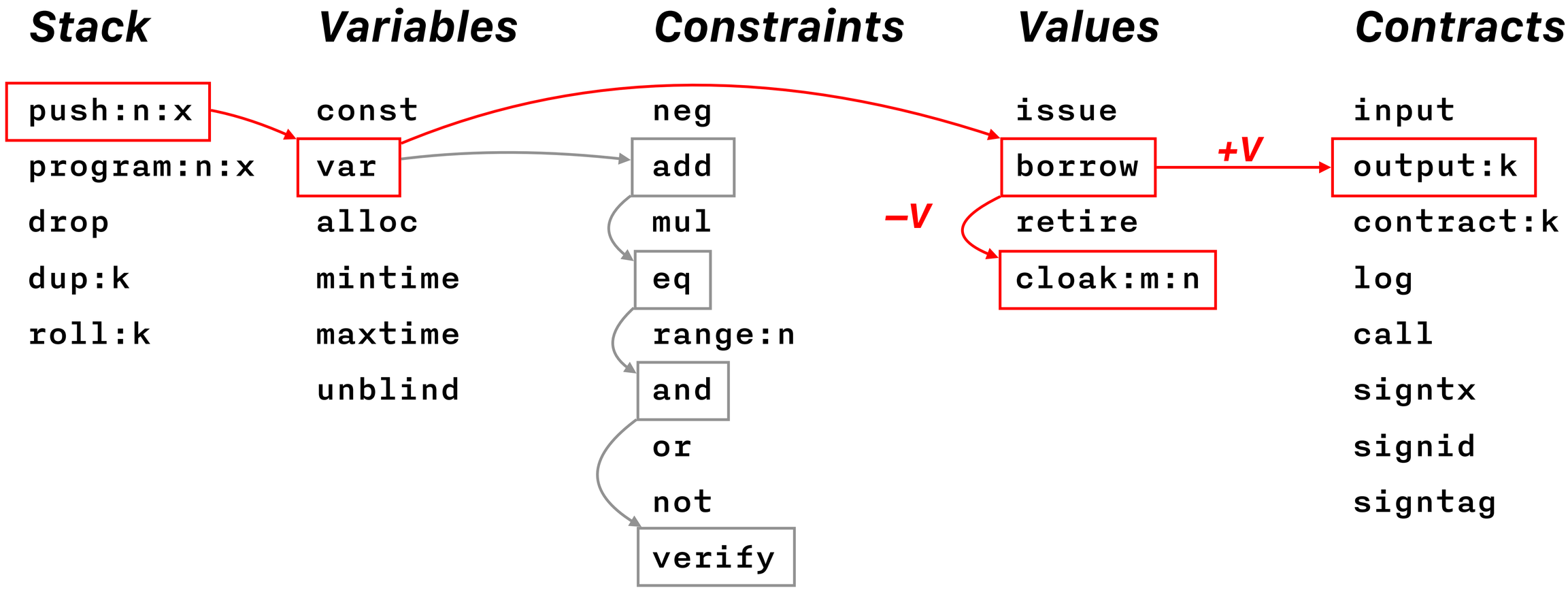
# EX: CUSTOM CONSTRAINTS

| *Stack* | *Variables* | *Constraints* | *Values* | *Contracts* |
|---|---|---|---|---|
| `push:n:x` | `const` | `neg` | `issue` | `input` |
| `program:n:x` | `var` | `add` | `borrow` | `output:k` |
| `drop` | `alloc` | `mul` | `retire` | `contract:k` |
| `dup:k` | `mintime` | `eq` | `cloak:m:n` | `log` |
| `roll:k` | `maxtime` | `range:n` | | `call` |
| | `unblind` | `and` | | `signtx` |
| | | `or` | | `signid` |
| | | `not` | | `signtag` |
| | | `verify` | | |



***Create variables from commitments, make expressions, form constraints and add them to the constraint system.***

# EX: CUSTOM CONSTRAINTS




*A variable defines a payment constraint with* `borrow` + `output`.

*Negative value is mixed with an actual payment in the* `cloak`.

# *LINEAR TYPES + CAPABILITIES*

*In ZkVM contracts imperatively express their requirements, entirely avoiding bugs like* ***confused deputy problem.***

# ZkVM TRADEOFFS

# *NOT TURING-COMPLETE*

*ZkVM optimized for **financial uses**, not arbitrary computations:*

- *issuing tokens and fundraising,*
- *multi-party vaults,*
- *derivative instruments,*
- *payment channels.*

# *O(n) BLOCKCHAIN*

- *Only zkSNARKs allow efficient compression (e.g. Coda).*
- *SPV clients use ≈50x less traffic than full blocks.*
- *Bootstrap from trusted source via Utreexo roots.*

# PRIVACY FEATURES

| ***Private*** | ***Not private*** |
| --- | --- |
| *Asset types* | *Programs* |
| *Asset quantities* | *Transaction graph* |
| *Data parameters* | |
| *In-transaction flow* | |

# 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*

*Taproot reveals only in dispute and only a specific branch.*

*Transaction graph*

*CoinJoin scales better.*

*Hiding UTXO links requires O(n) storage (nullifiers).*

# PERFORMANCE

# 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,*
- *≈1.5 Kb/proof, 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:*

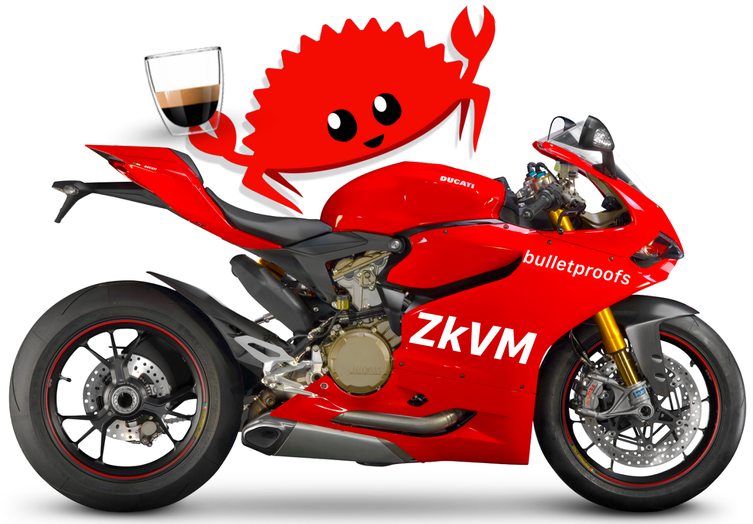
*github.com/dalek-cryptography/bulletproofs*

*ristretto.group*

*merlin.cool*

# THANK YOU

*Oleg Andreev*
**@oleganza**




*Project Slingshot is sponsored by Stellar Development Foundation.*