An Authentication Language for Blockchain Based on Σ-Protocols Enhanced by Boolean Predicates

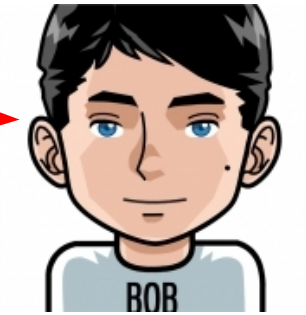
Alexander Chepurnoy

Ergo Platform and IOHK Research

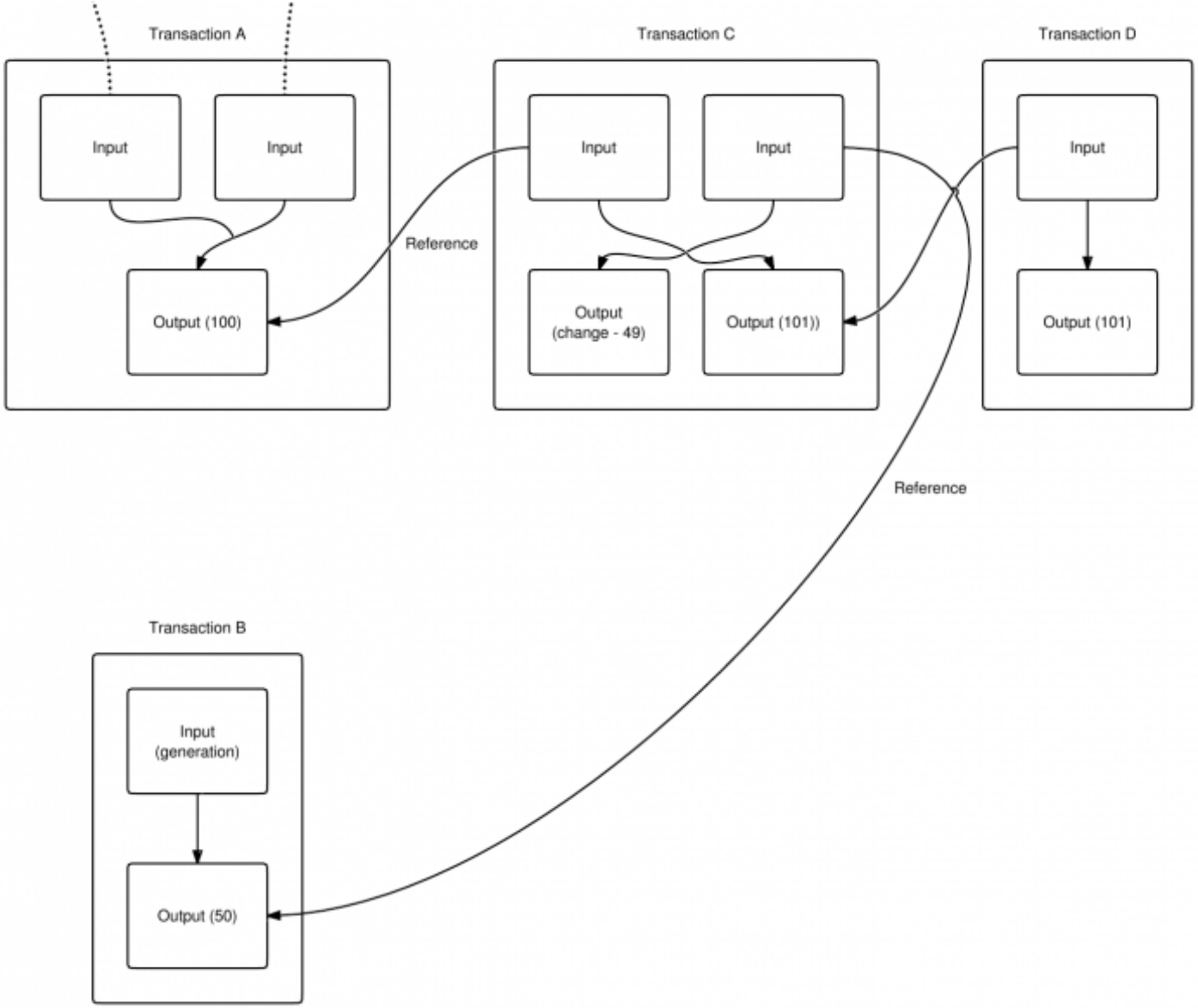
Cash money: no protection!

Cryptocurrencies: Protection!

pubkey_A



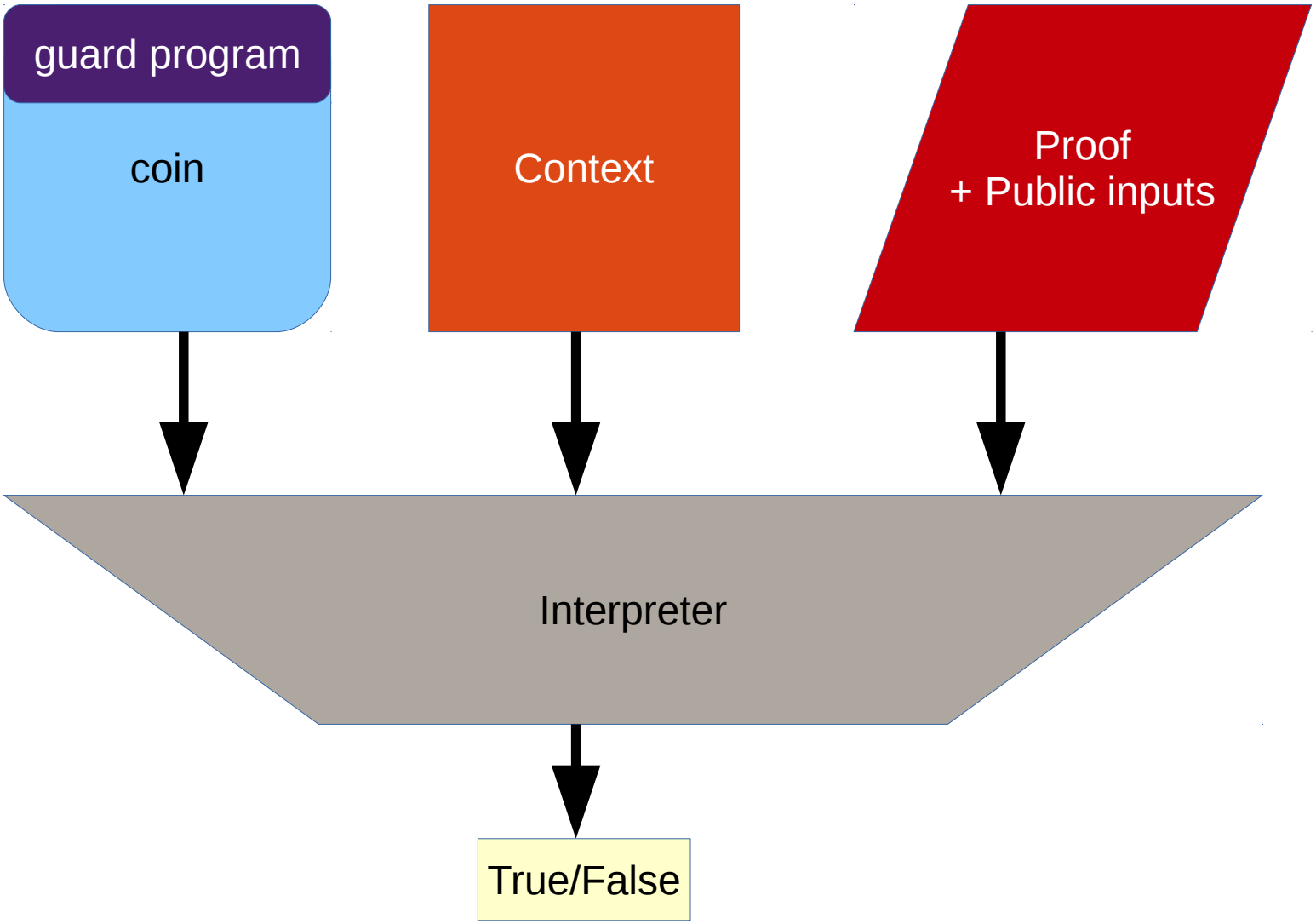
Bitcoin Transaction



Bitcoin Script

- Coin is an object created by an user
- Protected by a script
- Interpreter is to be run by anyone (against non-interactive proof to be included into the blockchain)

Authentication Language



Bitcoin Script

- A program is in stack-based language
- Proof is in the same language too
- Context is some meta-info on spending transaction & blockchain
- Cryptographic statements: OP_CHECKSIG, OP_CHECKMULTISIG, OP_HASH256 (and few others)

Bitcoin Script

Output script (proposition):
OP_DUP
OP_HASH160 <pubKeyHash>
OP_EQUALVERIFY
OP_CHECKSIG

Input script (proof):
<sig>
<pubkey>

Output: true if no failure during execution and top stack element is non-zero, false otherwise

Bitcoin Script Problems

- High-level cryptographic statements are fixed (to add ring or threshold signatures, hard-fork is needed)
- Limited support for cryptographic protocols
- The language was designed with no systematic approach, very limited, but many instructions were found problematic (DoS attacks / security issues) and switched off
- No tooling

Motivation

- More **powerful** language than the Bitcoin Script
- Still **simple**
- **Efficiency**: only parts of the blockchain which can be queried efficiently should be queried
- **Safety guarantee**: verification time for any script must be no more than a predefined constant C
- Tooling-friendly

Related Work

- Smart signatures languages (Simplicity, DEX, **Σ-State**)
- Safer “Turing-complete” languages (Plutus, Michelson, IELE etc)

Σ–State (Ergo Platform)

- Native cryptographic protocols
- Efficient execution (lean state, prover pays if something above a trivial level is needed)
- Simple model
- Provably bounded

Schnorr Protocol (a Σ–Protocol)

- For publicly known value h (in group G with known generator g), the prover is proving that there exists w, such as $g^w = h$ (in ZK)

- Prover commits to random r by sending $a=g^r$
- Verifier samples a random challenge $e=\{0,1\}^t$
- Prover computes $z = r+ew$

- Verifier accepts iff $g^z = a*h^e$

Generalized Schnorr Proofs

- subclass of Σ-protocols(ZKPoK protocols) for dlog statements
- **composable** (OR, AND, k-out-of-n)
- possible to turn into signatures by using Fiat-Shamir transformation
- **efficient** (Schnorr signature scheme in a simplest case)

Example 1:

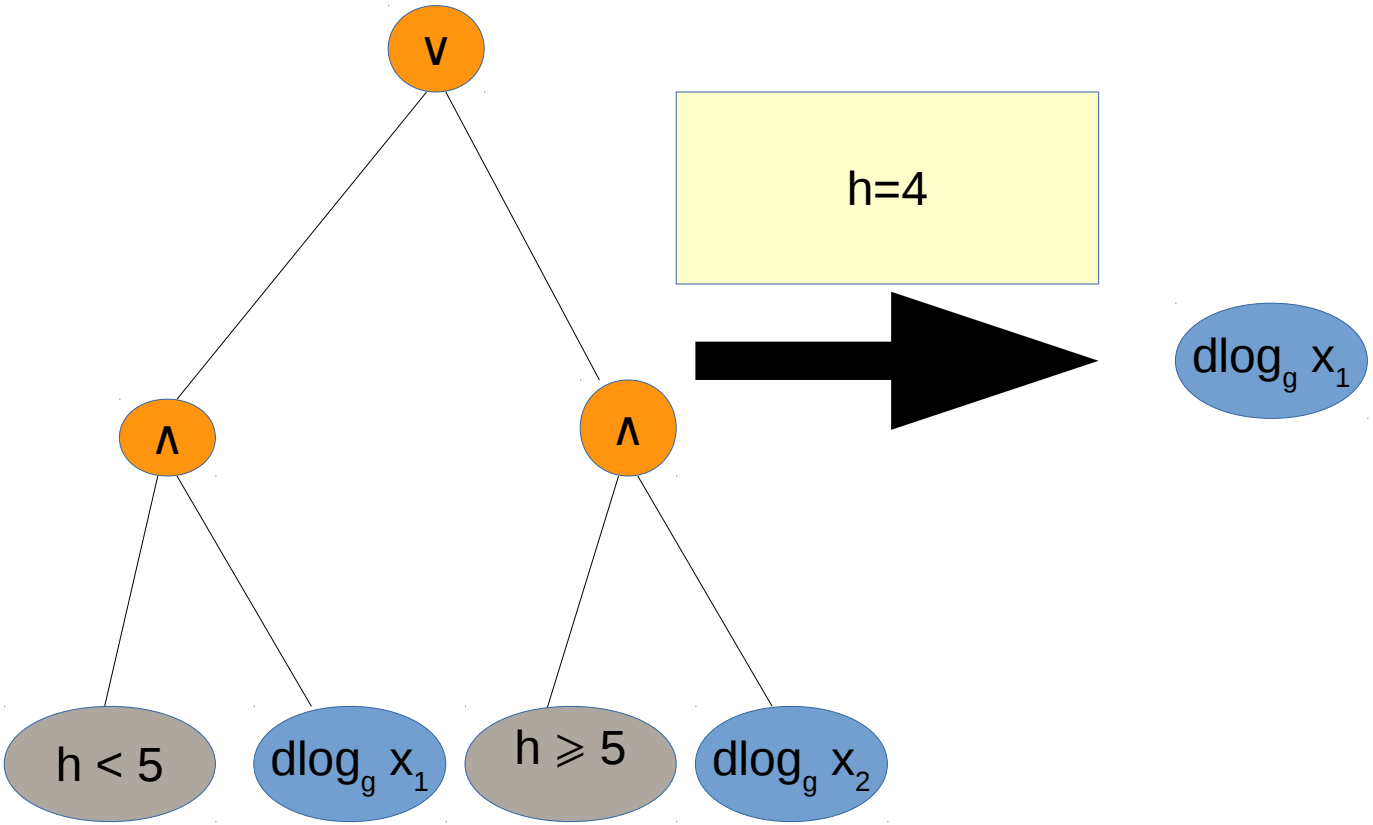
- **height > 100 \/ dlog_g x**

  “if height of a block to which a spending transaction is included > 100, output is spendable by anyone, before that output is spendable to a party presenting (zero-knowledge) proof a knowledge such *w* that $g^w=x$”

- **(height > 100 /\ dlog_g x1) \/ (dlog_g x1 /\ dlog x2)**

  “output could be spent anytime by proof of knowledge of both *x1* and *x2*, after block#100 it could be also spent by proof of *x1* knowledge”

Example 2:

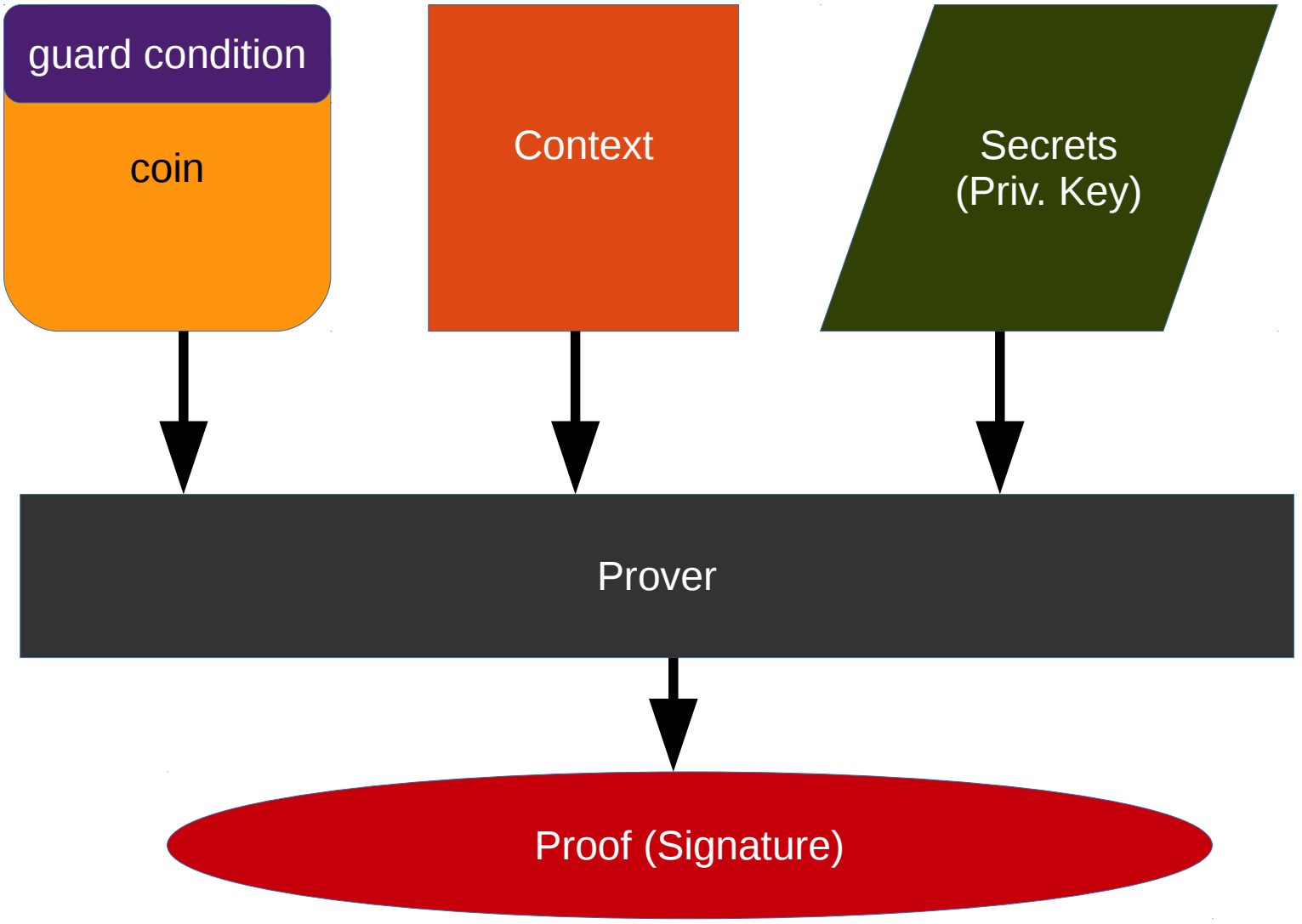


(h < 5 /\ dlog_g x1) \/ (h >= 5 /\ dlog x2)

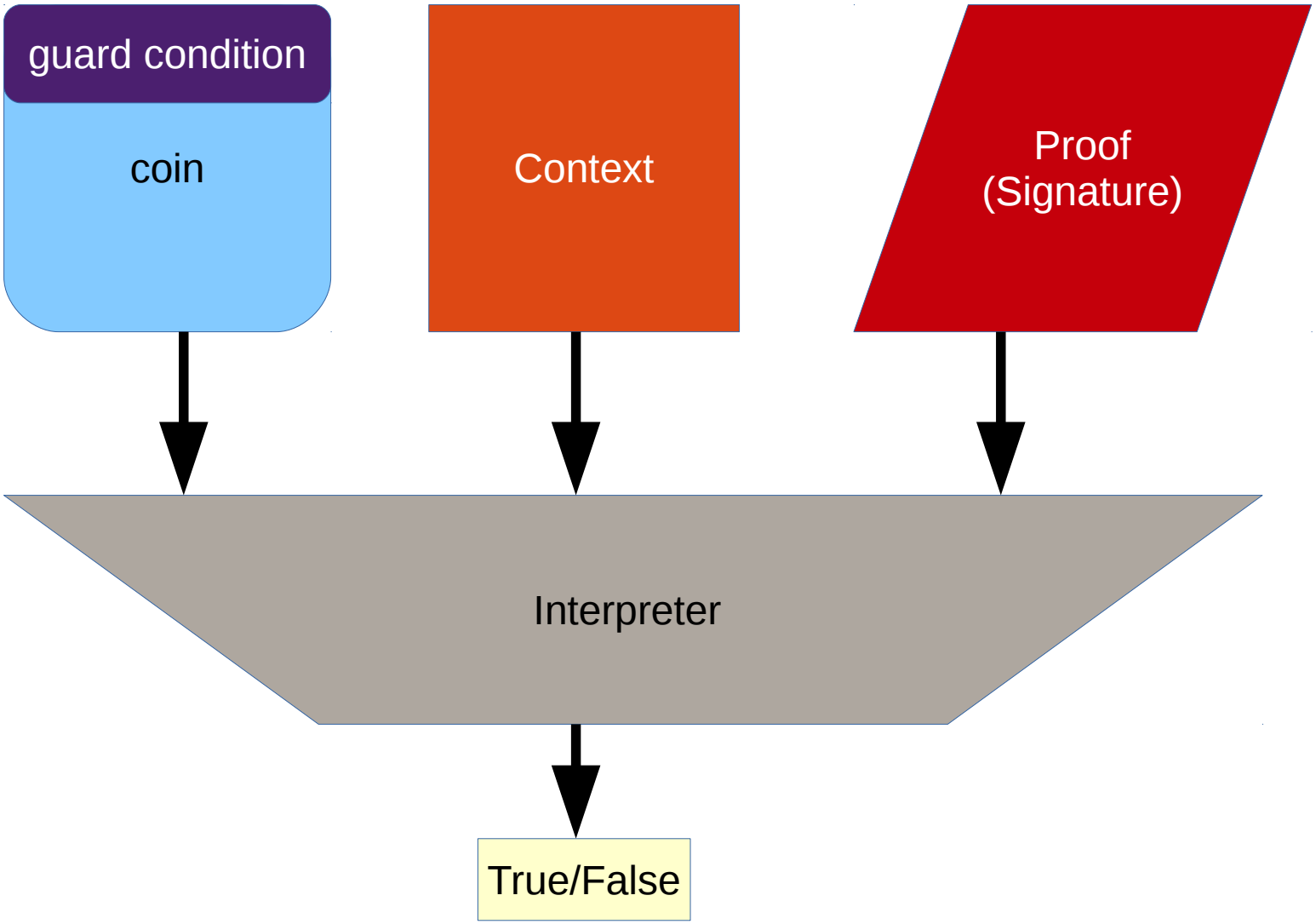
Σ–State

- combines cryptographic statements with predicates over context with /\ , \/ , k-out-of-n connectives
- 2-phase interpretation on both prover/verifies sides: first the composite logic formula to be reduced to one contains only connectives and cryptographic statements by evaluating state predicates, then prover generates a proof for the cryptographic formula, verifier checks it
- In non-interactive setting (e.g. blockchain) the proof is to be turned into a signature by using Fiat-Shamir

Generating a signature



Verifying:


guard condition
coin
Context
Proof
(Signature)
Interpreter
True/False

Bitcoin on Steroids: An Easy Way

Bitcoin: state is just about spending transaction bytes, block height & timestamp (for CLTV) + prover arguments

Enhancing the context: height, coins the spending transaction spends and creates, UTXO snapshot root hash, prover arguments

An output has additional data registers.

Crowdfunding Example

- **outputs**

- (height >= 100 /\ dlog x1) \/
  (height < 100 /\ exists(**outputs**,
  amount >= 100000 /\ proposition = dlog x2)

  “coin could be spent by a crowdfunding transaction if it pays at least 100000 tokens to x2 public key holder raising funds before block #100, or getting back to x1 after that”

And More Applications:

- Demurrage currency
- Oracles
- Better mixers
- Turing-complete systems (Rule 110)

And More:

- Our expressions are typeable, so not well-formed expressions (like “h>”) are rejected early
- As context is fully deterministic, it is possible to get upper-bound for verification cost, and reject too heavy expressions early

Cryptography

- Ring and threshold signatures for free, but may be not efficient(verification time and signature size are linear to size of the ring)
- We can include special efficient Σ-protocols (e.g. Groth-Kohlweiss ring signatures from EuroCrypt’2015)
- To have ability to update cryptographical statements set, languages like ZKPDL (Meiklejohn et al. USENIX’2010) could be used

Open Questions

- **No any standards** for generalized Schnorr proofs
- Could the authentication language be useful outside the blockchain?

Links

- A paper is coming
- Interpreter is online

  https://github.com/ScorexFoundation/sigmastate-interpreter

Questions?