Σ-STATE AUTHENTICATION LANGUAGE, AN ALTERNATIVE TO BITCOIN SCRIPT

Alexander Chepurnoy, IOHK Research

Motivation

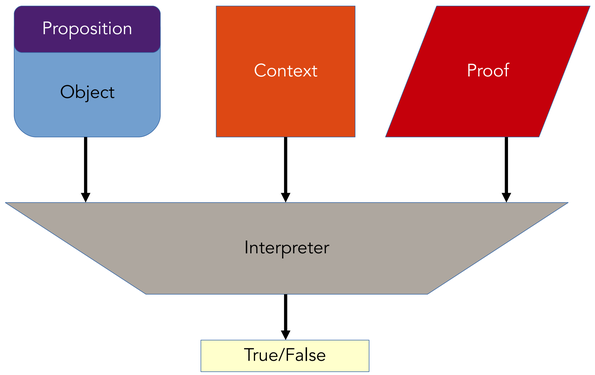
Bitcoin Script is a stack-based language. Interpretation of a program protecting a coin (an output) is about gluing it with a spending program and execute it to get a boolean outcome, There are proposals to use such a language outside digital currencies (e.g. Smarter Signatures[1]).

While Bitcoin Script allows for some contracts to be programmed, its abilities are limited while many instructions were removed after denial-of-service or security issues discovered. To add new cryptographic primitives, e.g. ring signatures, a hard-fork is required.

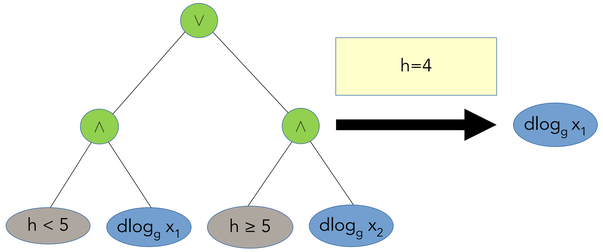
Thus the motivation we have is to develop an alternative language to protect coins in a cryptocurrency (and possibly not only), which is more flexible, extensible and powerful than one in Bitcoin (but still not "quasi Turing-complete" as Ethereum, as a lot of problems come with such power).

Contact Information:
Alexander Chepurnoy (alex.chepurnoy@iohk.io)

Σ-STATE LANGUAGES



- proposition is a logic formula consists of cryptographic statements, context predicates, $\wedge$, $\vee$, k-out-of-n connectives.
- cryptographic statements are provable via Σ–protocols.
- 2-phase interpretation on both prover/verifies sides: first the composite formula to be reduced to one contains only connectives and cryptographic statements, then prover generating a proof for the cryptographic formula, verifier checks it



EXAMPLES

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

Enhancing the context: output, height, spending transaction outputs values & scripts + prover arguments

CROWDFUNDING

a backer pays a project if spending transaction contains its input with not less than 100000 tokens, with timeout at height 100

- *(height ≥ 100 ∧ dlog_g backerPK) ∨ (height < 100 ∧ tx.has_output (amount ≥ 100000, proposition = dlog_g projectPK)*

DEMURRAGE CURRENCY

- a miner is enforcing every output to pay 2 tokens each 100 blocks, even if it is not touched, via this special script
- *regular_script ∨ (height ≥ (out.height + 100) ∧ tx.has_output(value ≥ (out.value − 2), script = out.script))*

Both scripts are not possible in Bitcoin!

FURTHER RESEARCH

Extensibility

Cryptographic statements could be improved with a language like ZKPDL[2], how to enrich context?

Cost Analysis

How to limit avoid tree explosion and charge fairly for verification?

Features

For example, we can allow an arbitrary output to be presented, along with a Merkle proof, if UTXO set is authenticated. Or we can add some memory to outputs.

Protocols

Show more useful protocols, such as efficient ring signatures (by using e.g. Groth-Kohlweiss scheme).

Please feel free to join the research!

REFERENCES

1. Allen, Cristopher, et al, "Smarter Signatures: Options to Ensure Fiduciary Protection on the Web" https://www.w3.org/2016/04/blockchain-workshop/interest/allen-todd-shea.html
2. Meiklejohn, Sarah, et al. "ZKPDL: A Language-Based System for Efficient Zero-Knowledge Proofs and Electronic Cash." USENIX Security Symposium. Vol. 10. 2010.