# Scripting smart contracts for distributed ledger technology

Pablo Lamela Seijas[1], Simon Thompson[1], and Darryl McAdams[2]

[1] University of Kent, UK
{pl240,S.J.Thompson}@kent.ac.uk
[2] IOHK
darryl.mcadams@iohk.io

Distributed Ledger Technology (DLT) offers a way of maintaining a synchronised log in a non-centralised, distributed way; notably, this allows the implementation of cryptocurrencies and, more recently self-enforcing smart contracts. Bitcoin is the first widely-used implementation of a cryptocurrency but it has very limited scripting capabilities in practice. Ethereum allows smart contracts to contain arbitrary time-bounded turing-computable code that is executed and validated in a virtual machine. Nxt moves scriptability to clients and provides a delimited functionality through an API.

Because smart contracts can control money and potentially other assets, it is crucial that they behave as expected, not only in normal conditions, but also when attacked by malicious agents. In particular, contracts must be *reentrant* if they call unknown code, they must gracefully handle all kinds of *exeception*s, they must not expect agents to collaborate (in some cases by including *rewards and penalties* to deter attacks).

Designers of smart-contract languages and cryptocurrencies may mitigate the likelihood of errors being made by their users by carefully designing them to be intuitive, explicit, and by providing well-tested artefacts. Some examples of effort in this direction include: the use of *zero-knowledge proofs* for providing anonymity (see Zerocash); the use of *SNARKS* to hide private inputs (Hawk allows to design contracts by separating private and public parts); and allowing the use and enforcement of *higher-level specifications*, like the use of polymorphic types, combinators, finite-state machines (FSMs), or domain specific languages (DSLs). Additionally, there are many open challenges that are specific to DLT systems, like the design of ways for *amending the rules* (see Tezos), the *unpredictability* of the initial execution state derived from the decentralisation, the need for a safe *source of randomness*, the *cost of validating* the contracts (which could be mitigated through the use of verifiable computation), the amount of work required by *proof-of-work* (see *proof-of stake*), and the need to preserve the delicate *equilibrium of incentives* that keeps block-chains secure.

In the full paper[3], we provide references for all the work mentioned here, we survey these and other representative examples of the advanced use of cryptocurrencies and blockchains beyond their basic usage as a payment method, and we analyse existing scripting solutions, their strengths and weaknesses, and some existing solutions for known problems with them. Through our work, we have seen that, while there have been many diverse efforts in different directions, there are still many open questions, no universal solutions, and lots of room for future research and experimentation.

[3] Pablo Lamela Seijas, Simon Thompson, and Darryl McAdams. *Scripting smart contracts for distributed ledger technology*. 2016. URL: https://eprint.iacr.org/2016/1156.pdf