The term “smart contract” can be traced back to a short 1994 paper by cryptocurrency pioneer (and Satoshi candidate) Nick Szabo, in which he defines it simply as a “computerized transaction protocol that executes the terms of a contract.” [1] Szabo was keenly aware of the potential these litigious computer programs had to change society (or at least digital transacting). It is only now, though, with the advent and subsequent explosion of purely digital currency, that the sweeping ramifications of the technology are coming calinto the fore. Before we can consider these broader societal implications, however, we must consider the substantial challenges smart contracts create for the distributed platforms on which they run.

Although Bitcoin is not tightly associated with the concept in the public zeitgeist, smart contracts it has supported them from its inception. [2] The scripting environment, however, is deliberately limited in scope. Notably, loops and gotos are omitted from the minimalistic, Forth-like language, rendering it Turning *in*complete. Moreover, scripts themselves are stateless (though their inputs come from the blockchain, which is stateful by its nature). [3] These qualities simplify verification of correctness and execution at the expense of flexibility.

Ethereum’s contract scripting environment, by contrast, is complex and highly expressive. Where Bitcoin emphasizes stability, Ethereum emphasizes adaptability. An Ethereum smart contract can be made to enforce practically any condition, however complex, so long as its inputs and outputs may be reliably digitized. Indeed, the Ethereum Virtual Machine, or EVM, is Turing complete, meaning anyone sitting down to write an Ethereum smart contract has the full power of a general-purpose computer at their disposal. This, as mentioned, comes at the expense of stability (as any CryptoKitties enthusiast will tell you) and also, potentially, security.

Our system, Cryptokernel, settles on an approach somewhere between the relative extremes of Ethereum and Bitcoin. CryptoKernel’s fundamental architecture is much more similar to Bitcoin’s than it is to Ethereum’s, in that it is UTXO-, rather than account-, based. Additionally, Cryptokernel scripts, as Bitcoin scripts, may evaluate to one of only two possible values: true or false. Like Ethereum’s scripts, however, they may also query the blockchain directly, and they have access to most features of Lua, which is a powerful (Turing complete) language.

In this paper, rather than trying to establish precisely which approach is best, we will instead assess each strategy’s comparative strengths and weaknesses, with consideration to factors such as robustness, computational intensity, flexibility, and ease of use. Included will be a brief review of the existing literature. We hope that cryptocurrency academics, professionals, and enthusiasts alike will find this research useful, and that they will consider its findings as they further develop the blockchain ecosystem.

[1]
http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html

[2] https://bitcoin.stackexchange.com/questions/29754/history-behind-the-scripting-language-in-bitcoin

[3] https://bitcoin.org/en/developer-guide