shorish research

# Enabling Trust on the Blockchain

Dr. Jamsheed Shorish*

October 20, 2017

## 1 Overview

At first glance the idea of implementing a trust process for a contract that has been coded within a blockchain may seem superfluous, as the idea of a *smart contract* is to eliminate the need for contract validation, verification of terms, necessity of audit, or other trust enabling tools.[1] Design challenges are actively discussed in smart contract development,[2,3] but there is a widespread feeling in the development community that once a smart contract has been 'correctly' designed, further intervention is unnecessary.

However, in practice there is a wide scope for traditional validation, verification and auditing functions whenever tangible assets (such as fiat money, real estate, financial assets and their derivatives etc.) are affected by blockchain-encoded outcome paths, irrespective of whether or not the contract has been 'correctly' designed. One reason for this is straightforward and involves what one might categorize as a human tendency to feel loss more strongly than gain:[4]

> *A party will seek redress when an adverse outcome they subjectively ascribe a low (or no) probability of occurring actually happens, and the outcome has a large negative effect on the stock of that party's tangible assets.*

*Shorish Research white paper 10/17a. *Shorish Research provides strategic consulting services built upon academic and real-world application expertise in the area of Computational Business.*

[1]See e.g. Blockgeeks' article "Smart Contracts; The Blockchain Technology That Will Replace Lawyers", accessed 19 October 2017.

[2]See e.g. Ethereum's Solidity documentation for a discussion of smart contracts built upon the Ethereum blockchain platform.

[3]Vitalik Buterin, "Blockchain and Smart Contract Mechanism Design Challenges", First Workshop of Trusted Smart Contracts, April 2017 (slides).

[4]An example is prospect theory (Kahneman and Tversky, "Prospect Theory: An Analysis of Decision Under Risk", *Econometrica* vol. 47 no. 2, pp. 263-291).

A classic example is an insurance contract, where the type of loss insured and the probability of loss together determine the probability of acceptance of a claim, after a particular loss has occurred. If a low probability event such as a hurricane causes catastrophic loss, the causative factors contributing to such loss (e.g. wind, flood) may, when the contract is interpreted during a claim, be excluded from the type of loss specified in the contract.[5] Nevertheless, the contracting party suffering the loss will still attempt to seek redress, because the scale of the catastrophe (e.g. the complete destruction of a domicile) carries a high quality-of-life penalty.

Another example concerns "flash crashes", defined as asset market movements which are triggered from algorithmic corrections leading to precipitous changes in prices over very short time intervals (an example is the May 2010 US stock market crash-and-recovery). Although in principle investors are aware of "the rules of the game" when investing and recognize the potential for steep losses (and potentially large gains) as a result of asset price changes, these changes are not efficiently priced into asset valuations when high-frequency trading is responsible for *ex ante* low probability, but high volatility, price movements. As a result, there is scope for seeking redress in such situations (using e.g. a class action lawsuit mechanism[6]).

This is also why outcomes which are by definition assumed to be unpredictable by everyone often motivate parties to include indemnification against such outcomes (e.g. 'Act of God' contract clauses), which both parties state are outside of the controllable circumstances of the contract but nevertheless impact the distribution of tangible assets. Without such unanimity, however, contracts which depend upon (and presumably price) low probability events that significantly change the distribution of tangible assets for one party are *subjectively* more likely to be construed as 'unfair' *ex post* of such events, and hence are more vulnerable to adjudication.

There is also a second reason why trust tools such as validation, verification and auditing functions will be necessary in the near future to ensure that blockchain-encoded contracts can be written:

> *One or more parties will seek redress if there is a sufficiently large wedge driven between the intent of a contract and its actual outcome, and that outcome significantly impacts a party's tangible assets.*

An important example of this within the blockchain ecosystem is the destruction of 'The DAO', a Decentralized Autonomous Organization (hence the name) launched at

[5]Chris French, "Hurricanes, Fraud, and Insurance: The Supreme Court Weighs in on, But Does Not Wade Into, the Concurrent Causation Conundrum in State Farm Fire and Casualty Company v. Rigsby", *165 U. Pa. L. Rev. Online 99*, 2017.

[6]Tara E. Levens, "Too Fast, Too Frequent? High-Frequency Trading and Securities Class Actions", *The University of Chicago Law Review* vol. 82, no. 3, pp. 1511-1557, 2015

the end of April 2016 as a smart contract on the Ethereum blockchain infrastructure.[7] The original intent of The DAO was to serve as an investment vehicle along the lines of Kickstarter. After raising c. USD 150 million in the first month of its inception, in June 2016 an anonymous user was able to execute part of The DAO's smart contract code to expropriate cybercurrency (Ethereum's 'ether') worth approximately USD 50 million, which was then held in a second smart contract that the user owned.[8]

The event precipitated a major schism within the Ethereum community, when the developers of Ethereum decided to change the smart contract of the The DAO to be able to return all of the invested funds to the original investors. Although this was presented by the developers and much of the community as 'the right thing to do', as it was 'clear' that the intent of The DAO as an investment vehicle had been divorced from its execution (viz. the movement of USD 50 million to the account of a single user, without an associated project), a significant part of the community disagreed. They stated, rather, that since the anonymous user had used The DAO's smart contract programming code *as it was written*, in the end the expropriation was in fact part of the set of outcomes that The DAO as a smart contract had provided–irrespective of whether the original writers of The DAO were aware of it. In other words, according to this view the smart contract itself had not been compromised, and its programming code did what it had been programmed to do, regardless of intent. Proponents of this perspective hold that normative judgements, such as whether or not a contract's execution or outcome was 'wrong', should remain in the province of the legal system.[9]

When the smart contract of The DAO was changed to return investor stakes (via a 'hard fork' of the Ethereum blockchain), Ethereum split into two blockchains: Ethereum with the new contract and recaptured investor funds ("Ethereum"), and the original Ethereum blockchain protocol, which contained the old contract and its own cryptocurrency ("Ethereum Classic"). As of this writing both blockchains are in existence and their cryptocurrencies are trading with value–note that in Ethereum Classic, original investors in The DAO did not fully recover their funds and the expropriated funds remain available to the anonymous user, albeit in a new currency.

This Overview serves to illustrate that the development of contracts on the blockchain does not obviate the need for trust tools–in fact, it may be argued that the contrary holds true, as this novel decentralized environment speaks to the need for a careful representation of trust and validity on the one hand (as demonstrated above), while preserving the efficiency gains that blockchain is expected to provide in the future on the other. This balance between trust and efficiency implies that the main challenge will be to develop

[7] See e.g. Richard Waters, "Automated company raises equivalent of $120m in digital currency", *Financial Times*, May 17, 2016.

[8] Phil Daian, "Analysis of the DAO exploit", *HackingDistributed* (blog), June 18, 2016.

[9] Arvicco, "Code is Law and the Quest for Justice", *Ethereum Classic Blog* (blog), September 9, 2016.

and implement trust tools that extend traditional trust services to the blockchain ecosystem in as 'non-disruptive' a fashion as possible. In the remainder of this white paper we outline some of suggested requirements for validation, verification and auditing tools to be developed for, on and off the blockchain.

# 2 Validation: defining a contract on the blockchain

A smart contract defined on the blockchain first needs to be in a standardized format, so that disputes cannot focus on the form of the contract–this is a minimal requirement for a contract to be implementable. Contracts on the blockchain can be written in a variety of manners, each of which 'compiles' (i.e. is made machine-readable, to be executed as a running piece of code) to provide the same functionality.

Assuming for the moment that such an unambiguous formatting of a contract onto the blockchain is possible,[10] then *validating* a contract is as "simple" as ensuring the compiled code runs as intended. This means, in particular, that:

- the code should not contain **syntax** errors, i.e. errors that will prevent compilation;
- the code should not contain **runtime** errors, i.e. errors that will occur during execution;
- the code should not contain **side effects**, i.e. legitimate execution which does not contain syntax or runtime errors, but which nonetheless leads to outcomes that are not intended or desired;
- the code should pass all **tests**, where a test is a preconfigured scenario that includes the desired outcome–the code is run against the test, and if the code outcome matches the desired outcome, the code passes the test.

Naturally it is important to describe what is meant by e.g. 'side effects', and the test suite should encompass (ideally) all use cases, or all expected use cases up to a certain probability, while (again, ideally) removing all undesirable side effects from the code. Specifying the proper test suite requires a high degree of experience and is often more art than science. It can be difficult for complex code, but automated software exists to assist in testing and can be leveraged to help prevent the kind of unintended behavior demonstrated by The DAO's smart contract.

Validating a contract is perhaps the most logical application of trust tools to a blockchain contract–for example, validation allows the creation of insurance contracts on

[10] Disclosure: Shorish Research is actively developing such a *lexicon* for 'translating' in-real-life contracts to smart contracts.

the blockchain, where an insurance smart contract can be used to transparently indicate when claims are filed, approved and paid, improving the efficiency of the claims process while reducing the scope of fraud. In this sense the blockchain acts as a decentralized repository for a contract's details, fiduciary responsibilities, and transactions structure which, once registered on the blockchain, is immutable.

# 3 Verification: executing a contract on the blockchain

Actually executing a smart contract leads to *verification* of the outcome–a validated contract will execute under the conditions it has been defined to wait for, where such conditions are both user-generated (e.g. a claims submission for insurance) and externally generated (e.g. a derivative that exercises upon a terminal date, or upon an underlying reaching a strike price). In the latter case there must be an infrastructure in place to credibly report applicable market conditions in order to execute a smart contract that relies upon them–such infrastructure is called an 'oracle' in the blockchain literature, and may be provided by software or by dedicated (tamper-proof) hardware.[11] Part of verifying the execution of a smart contract, then, relies upon 1) the existence of such an oracle, and 2) the *consensus* among users that the oracle is truthfully revealing external information, so that data cannot be manipulated to the benefit of a contracting party. Defining, implementing and maintaining oracles thus occupies the intersection of decentralized and trusted third-party verification of external conditions, and it appears to be an open question whether smart contracts that rely upon oracles will always contain some form of centralized (i.e. third party authority) dependence as a result.

One may also harness the blockchain itself for verification of the execution of a contract. This uses the fact that the blockchain is a ledger that records both the user-generated and externally generated conditions and the transactions that result, and there is no scope for tampering with the condition-to-transaction causality because such a recording is immutable. While this cannot control for the existence of fraud 'off-chain', i.e. before the blockchain has been accessed, it does minimize the scope of such fraud because of the validation step performed earlier–recall that in that step all contract participants and contract stipulations were vetted and included in the blockchain.

Finally, if off-chain fraud is suspected, the immutable nature of the recording of user and external conditions serves to provide the factual basis for further investigation–the blockchain records *the actual sequence of events* and thus eliminates the risk of a fraudulent party obfuscating the link between conditions and transactions (by e.g. a 'cover-up' of the path from, say, false data fed to the smart contract, and the resulting transaction).

[11] See e.g. Eric Larchevêque, "Hardware Pythias: bridging the Real World to the Blockchain", *Ledger* (blog post), August 31, 2016.

In each of these verification steps–the oracle subsystem, on-chain verification using the immutability of the blockchain as a ledger, and using the blockchain to verify claimed off-chain external conditions–trust tools must be developed and implemented in a way that all sides of a contract agree are transparent and perform as intended.

# 4 Auditing: following trust along the blockchain

*Auditing* a contract usually involves mapping out the chain of events from the contract inception to its outcome (or set of outcomes), ensuring that every step along the way conforms to the accepted best practices and legal requirements for the type of contract under consideration, while ensuring that double-spending along the (potentially long) sequence of transactions is not performed. Auditing is not just a method of detecting fraud or 'cheating' on the contract–rather, it allows both parties to ensure *ex post* that the contract was implemented and executed according to the intent and expectation of each party, and serves as an important signal of trust for repeated interaction between the contracting parties.

A smart contract on the blockchain simplifies the auditing process by virtue of

- the clear and transparent chain of transactions that a contract creates, which are themselves part of the blockchain and are thus immutable (as described earlier). This forms the basis for consensus between contract parties that *actual events have taken place*. This in turn reduces the chance of dispute and hence lowers the cost of dispute resolution as a result of (or during) an auditing process.

- the widespread use of automation to execute contract outcomes, when coupled with user generated and externally generated input (via an oracle). Verification of the automated procedure itself is a form of 'pre-auditing' and may be used in the audit step to confirm that the contract is performing as intended. Again, this also ensures that one or more parties cannot claim that the terms of a contract have somehow been changed (which changes the contract outcome), because the terms have been encoded in the blockchain and are immutable.

- the design of the blockchain itself, that prevents double-spending from occurring. Although a smart contract may contain code that permits an unintended transfer of resources (as happened with the The DAO, cf. the Overview), the blockchain's record of transactions remains viable precisely because it prevents double-spending in essentially every situation.[12]

[12]The '51% attack', in which a user or cartel of users control more than half of the computing power to validate blocks in a blockchain, is one way to subvert the prevention of double-spending. But the

# 5 Concluding remarks

There is very little time to wait before trust tools become necessary for blockchain contracts. It is true that the current government regulatory structure has not delineated the full procedural workflow for managing blockchain contracts or the resulting transactions (questions such as "Are transactions on the blockchain taxable?" or "Is an Initial Coin Offering a securities launch?"[13] remain open). This may incline one to adopt a 'wait and see' attitude, relying upon government agency to act as a leader and pathfinder toward addressing regulatory hurdles. To do so, however, relinquishes the initiative that traditional trust and relationship accountancy and consulting organizations currently possess, in the application of their own expertise in this new arena.

As discussed in this white paper, challenges that must be faced include:

1. creating the proper lexicon to use to translate contracts into code;
2. implementing that code in a clear, incentive-compatible fashion; and
3. developing the required monitoring and auditing tools.

These latter tools must all be able to leverage software and hardware oracles as providers of objective truth for conditional outcomes of smart contracts, while ensuring that the blockchain continues to provide the efficiencies in intermediation, redress and payments that appear, initially, to be self-evident. We believe that when these challenges are met, it will be possible to profitably assist and enable an enterprise, so that they are able to take advantage of the 'low-hanging fruit' provided by efficiency gains without sacrificing the ability to create, follow and maintain a trust relationship on the blockchain.

reward mechanism for computing 'true' blocks, in addition to the difficulty in both achieving such a high degree of computing power and using that power to compute historical blocks quickly enough to create an advantageous blockchain continuation, generally militates against the success probability of this attack vector in Proof-of-Work blockchains such as Bitcoin. See e.g. the *Bitcoin Developer Guide*, accessed 15 October 2017.

[13]Recent decisions by the US Securities Exchange Commission (SEC) and the US Commodities Futures Trading Commission (CFTC) indicate that regulatory bodies are interpreting cryptocurrencies as securities and/or commodities. See The Securities and Exchange Commission's "Report of Investigation Pursuant to Section 21(a) of the Securities Exchange Act of 1934: The DAO", accessed 19 October 2017, and LabCFTC, "A CFTC Primer on Virtual Currencies", accessed 19 October 2017.