

Tezos: A Self-Amending Crypto-Ledger Position Paper

L.M Goodman

August 3, 2014

“Laissez faire les propriétaires.”

— Pierre-Joseph Proudhon

Abstract

The popularization of Bitcoin, a decentralized crypto-currency has inspired the production of several alternative, or “alt”, currencies. Ethereum, CryptoNote, and Zerocash all represent unique contributions to the crypto-currency space. Although most alt currencies harbor their own source of innovation, they have no means of adopting the innovations of other currencies which may succeed them. We aim to remedy the potential for atrophied evolution in the crypto-currency space by presenting Tezos, a generic and self-amending crypto-ledger.

Tezos can instantiate any blockchain based protocol. Its seed protocol specifies a procedure for stakeholders to approve amendments to the protocol, *including* amendments to the amendment procedure itself. Upgrades to Tezos are staged through a testing environment to allow stakeholders to recall potentially problematic amendments.

The philosophy of Tezos is inspired by Peter Suber’s Nomic[1], a game built around a fully introspective set of rules.

In this paper, we hope to elucidate the potential benefits of Tezos, our choice to implement as a proof-of-stake system, and our choice to write it in OCaml.

Contents

1 Motivation	2
1.1 The Protocol Fork Problem	3
1.1.1 Keeping Up With Innovation	3
1.1.2 Economics of Forks	4
1.2 Shortcomings of Proof-of-Work	5
1.2.1 Mining Power Concentration	5
1.2.2 Bad incentives	6
1.2.3 Cost	7
1.2.4 Control	8
1.3 Smart Contracts	8
1.4 Correctness	9
2 Abstract Blockchains	10
2.1 Three Protocols	10
2.1.1 Network Protocol	10
2.1.2 Transaction Protocol	11
2.1.3 Consensus Protocol	11
2.2 Network Shell	11
3 Proof-of-Stake	12
3.1 Is Proof-of-Stake Impossible?	12
3.2 Mitigations	13
3.2.1 Checkpoints	13
3.2.2 Statistical Detection	13
3.3 The Nothing-At-Stake Problem	14
3.4 Threat Models	14
4 Potential Developments	15
4.1 Privacy Preserving Transactions	15
4.1.1 Ring Signatures	15
4.1.2 Non Interactive Zero-knowledge Proofs of Knowledge	15
4.2 Amendment Rules	15
4.2.1 Constitutionalism	15
4.2.2 Futarchy	16
4.3 Solving Collective Action Problems	16
4.3.1 Raising Awareness	16
4.3.2 Funding Innovation	16

1 Motivation

In our development of Tezos, we aspire to address four problems we perceive with Bitcoin[2]:

- The “hard fork” problem, or the inability for Bitcoin to dynamically innovate due to coordination issues.
- Cost and centralization issues raised by Bitcoin’s proof-of-work system.
- The limited expressiveness of Bitcoin’s transaction language, which has pushed smart contracts onto other chains.
- Security concerns regarding the implementation of a crypto-currency.

1.1 The Protocol Fork Problem

1.1.1 Keeping Up With Innovation

In the wake of Bitcoin’s success, many developers and entrepreneurs have released alternative crypto-currencies (“altcoins”). While some of these altcoins did not diverge dramatically from Bitcoin’s original code¹, some presented interesting improvements. For example, Litecoin introduced a memory hard proof of work function² and a shorter block confirmation time. Similarly, Ethereum has designed stateful contracts and a Turing-complete transaction language[3]. More important contributions include privacy-preserving ring signatures (CryptoNote)[4] and untraceable transactions using SNARK (Zerocash)[5].

The rise of altcoins has inspired a vast competition in software innovation. Cheerleaders for this Hayekian growth, however, miss a fundamental point: for a cryptocurrency to be an effective form of money, it needs to be a stable store of value. Innovation within a ledger preserves value through protecting the network effect giving the currency its value.

To illustrate the problem of many competing altcoins, let us compare a crypto-currency and a smart phone. When purchasing a smart phone, the consumer is paying for certain features, such as the ability to play music, check email, message his friends, and conduct phone calls.

Every few weeks, a newer smartphone model is released on the market which often contains enhanced features. Though consumers who have the older model may be jealous of those with the latest model, the introduction of newer smartphones does not render older smartphones dysfunctional.

This dynamic would change, however, if the newest phones could not communicate with older models. If the many models and styles of smartphone could not be used together seamlessly, the value of each smartphone would be reduced to the number of people with the same model.

Crypto-currencies suffer from the same fate as smartphones which are incompatible with one another; they derive their value from a network effect, or the number of users who have given it value. To this end, any innovation that occurs outside of a crypto-currency will either fail to build enough network effect to be noticed, or it will succeed but undermine the value of the savings in the old currency. If smartphones were incompatible with older models, there would

¹wow, such unoriginal

²scrypt mining ASICs are now available

be either very little innovation or extremely disruptive innovation forcing older phones into obsolescence.

Side-chains are an attempt to allow innovations which will retain compatibility with Bitcoin by pegging the value of a new currency to Bitcoin and creating a two-way convertibility. Unfortunately, it's unclear whether they will be flexible enough to accommodate protocols substantially different from Bitcoin. The only alternative so far is to fork the protocol.

1.1.2 Economics of Forks

To understand the economics of forks, one must first understand that monetary value is primarily a social consensus. It is tempting to equate a cryptocurrency with its rules and its ledger, but currencies are actually focal points: they draw their value from the common knowledge that they are accepted as money. While this may seem circular, there is nothing paradoxical about it. From a game theoretic perspective, the perception of a token as a store of value is stable so long as it is widespread. Note that, as a ledger, Bitcoin is a series of 1s and 0s. The choice to treat the amounts encoded within unspent outputs as balances is a purely *social* consensus, not a property of the protocol itself.

Changes in the protocol are referred to as "forks"³. They are so called because, in principle, users have the option to keep using the old protocol. Thus, during a fork, the currency splits in two: an old version and a new version.

A successful fork does not merely require software engineering, but the coordination of a critical mass of users. This coordination is hard to achieve in practice. Indeed, after a fork, two ledgers exist and users are confronted with a dilemma. How should they value each branch of the fork?

This is a coordination game where the answer is to primarily value the branch other users are expected to primarily value. Of course, said users are likely to follow the same strategy and value the branch for the same reason. These games were analyzed by economist Thomas Schelling and focal points are sometimes referred to as "Schelling points" [6].

Unfortunately, there is no guarantee that this Schelling point will be the most desirable choice for the stakeholders, it will merely be the "default" choice. A "default" could be to follow the lead of a core development team or the decrees of a government regardless of their merit.

An attacker capable of changing social consensus controls the currency for all intents and purposes. The option to stick with the original protocol is widely irrelevant if the value of its tokens is annihilated by a consensus shift.⁴

Core development teams are a potentially dangerous source of centralization. Though users can fork any open source project, that ability offers no protection against an attacker with enough clout to alter the social consensus. Even assuming the likely benevolence of a core development team, it represents a weak

³not to be confused with blockchain forks which happen *within* a protocol

⁴The argument that there can never be more than 21 million bitcoin because "if a fork raised the cap, then it wouldn't be Bitcoin anymore" isn't very substantive, for Bitcoin is what the consensus says it is.

point on which an attacker could exercise leverage.

Tezos guards against the vulnerabilities wrought by the source of centralization through radically decentralized protocol forks. It uses its own cryptoledger to let stakeholders coordinate on forks. This allows coordination and enshrines the principle that forks are not valid unless they are endogenous, making it much harder to attack the protocol by moving the consensus.

Suppose for instance that a popular developer announces his intention to fork Tezos without making use of the protocol's internal procedure. "Why would he attempt to bypass this process?" might ask stakeholders. Most certainly, because he knew that he wouldn't be able to build consensus around his proposed fork *within* Tezos.

This signals to the stakeholders that their preferred consensus would be to reject this fork, and the Schelling point is thus to refuse it, no matter the clout of that developer.

1.2 Shortcomings of Proof-of-Work

The proof-of-work mechanism used by Bitcoin is a careful balance of incentives meant to prevent the double spending problem. While it has nice theoretical properties in the absence of miner collusion, it suffers in practice from severe shortcomings.

1.2.1 Mining Power Concentration

There are several problems with proof-of-work as a foundation for crypto-currencies. The most salient problem, which is all too relevant as of 2014, is the existence of centralized mining pools, which concentrate power in the hands of a few individuals.

The proof-of-work mechanism is decentralized, which means that users do not need to *explicitly* trust anyone to secure the currency. However, *implicitly*, Bitcoin has yielded a system where all users have to trust the benevolence of one or two pool operators to secure the currency.

A conspiracy of miners holding more than 50% of the hashing power is known as 51% attack[7]. It allows the attackers to prevent transactions from being made, to undo transactions, to steal recently minted coins and to double spend[8].

A centralized mint signing blocks would be just as secure, and far less wasteful, as a miner controlling 51% of the hashing power. If a centralized mint is unacceptable to Bitcoin users, they should not tolerate *de facto* centralization of mining power.

The concentration of mining power is no coincidence: large mining pools face less variance in their returns than their competitors and can thus afford to grow their operation more. In turn, this growth increases their market share and lowers their variance.

To make things worse, the large mining pool ghash.io has hinted at a business model where they would prioritize "premium" transactions submitted directly

to them. This means that large miners would earn proportionally more than smaller miners. Sadly, p2pool has had trouble attracting hashing power as most miners selfishly prefer the convenience of centralized mining-pools.

Many have argued that fears of market concentration are overblown. They are generalizing hastily from the real world economy. Real businesses compete in a rapidly changing landscape where Schumpeterian creative destruction exercises constant evolutionary pressure on incumbents. Real businesses need local knowledge, face organizational issues and principal agent problems. Bitcoin mining is a purely synthetic economic sector centered around hashing power, a purely fungible commodity. It would be mistaken to hastily generalize and think that such a sterile environment is endowed with the same organic robustness that characterizes a complex, fertile, economy.⁵

Furthermore, the economic argument generally holds that natural monopolies have few incentives to abuse their position. The same could be said about a Bitcoin miner — after all, why would a dominant miner destroy the value of their investments by compromising the currency? Unfortunately, this still creates a huge systemic risk as such miners can be compromised by a dishonest attacker. The cost of executing a double spending attack against the network is *no more* than the cost of subverting a few large mining pools.

There have been proposals intended to address this issue by tweaking the protocol so it would be impossible for pool organizers to trust their members not to cheat. However, these proposals only prevent pools from gathering mining force from anonymous participants with whom there is no possibility of retaliation. Pooling is still possible between non-anonymous people: organizers may operate all the mining hardware while participants hold shares, or organizers may track cheaters by requiring inclusion of an identifying nonce in the blocks they are supposed to hash. The result of such proposals would thus be to increase variance for anonymous mining operations and to push towards further concentration in the hands of mining cartels.

Proof-of-stake, as used by Tezos, does not suffer from this problem: inasmuch as it is possible to hold 51% of the mining power, this implies holding 51% of the currency, which is not only much more onerous than controlling 51% of hashing power but implies fundamentally better *incentives*.

1.2.2 Bad incentives

There is an even deeper problem with proof-of-work, one that is much harder to mitigate than the concentration of mining power: a misalignment of incentives between miners and stakeholders.

Indeed, in the long run, the total mining revenues will be the sum of the all transaction fees paid to the miners. Since miners compete to produce hashes,

⁵It is possible that a new technology will supplant ASICs who themselves replaced FPGA boards. However, the pace of this type of innovation is nowhere fast enough to prevent miners from forming dominating positions for long period of times; and such innovation would benefit but a new (or the same) small clique of people who initially possess the new technology or eventually amass the capital to repeat the same pattern.

the amount of money spent on mining will be slightly smaller than the revenues. In turn, the amount spent on transactions depends on the supply and demand for transactions. The supply of transactions on the blockchain is determined by the block size and is fixed.

Unfortunately, there is reason to expect that the demand for transactions will fall to very low levels. People are likely to make use of off-chain transaction mechanisms via trusted third parties, particularly for small amounts, in order to alleviate the need to wait for confirmations. Payment processors may only need to clear with each other infrequently.

This scenario is not only economically likely, it seems necessary given the relatively low transaction rate supported by Bitcoin. Since blockchain transaction will have to compete with off-chain transaction, the amount spent on transactions will approach its cost, which, given modern infrastructure, should be close to zero.

Attempting to impose minimum transaction fees may only exacerbate the problem and cause users to rely on off-chain transaction more. As the amount paid in transaction fees collapses, so will the miner's revenues, and so will the cost of executing a 51% attack. To put it in a nutshell, the security of a proof-of-work blockchain suffers from a commons problem[9]. Core developer Mike Hearn has suggested the use of special transactions to subsidize mining using a pledge type of fund raising[10]. A robust currency should not need to rely on charity to operate securely.

Proof-of-stake fixes these bad incentives by aligning the incentives of the miners and stakeholders: by very definition, the miners *are* the stakeholders, and are thus interested in keeping the transaction costs low. At the same time, because proof-of-stake mining is not based on destruction of resources, the transaction cost (whether direct fees or indirect inflation) are entirely captured by miners, who can cover their operating costs without having to compete through wealth destruction.

1.2.3 Cost

An alternative is to keep permanent mining rewards, as Dogecoin[11] has considered. Unfortunately, proof-of-work arbitrarily increases the costs to the users without increasing the profits of the miners, incurring a deadweight loss. Indeed, since miners compete to produce hashes, the amount of money they spend on mining will be slightly smaller than the revenues, and in the long run, the profits they make will be commensurate with the value of their transaction services, while the cost of mining is lost to everyone.

This is not simply a nominal effect: real economic goods (time in fabs, electricity, engineering efforts) are being removed from the economy for the sake of proof-of-work mining. As of June 2014, Bitcoin's annual inflation stands at a little over 10% and about \$2.16M dollars are being burned daily for the sake of maintaining a system that provides little to no security over a centralized system in the hands of ghash.io.

The very security of a proof-of-work scheme rests on this actual cost being

higher than what an attacker is willing to pay, which is bound to increase with the success of the currency.

Proof-of-stake eliminates this source of waste without lowering the cost of attacks — indeed, it automatically scales up the cost of an attack as the currency appreciates. Because the thing you must prove to mine is not destruction of existing resources but provision of existing resources, a proof-of-stake currency does not rely on destroying massive resources as it gains in popularity.

1.2.4 Control

Last but not least, the proof-of-work system puts the miners, not the stakeholders, in charge. Forks for instance require the consent of a majority of the miners. This poses a potential conflict of interest: a majority of miners could decide to hold the blockchain hostage until stakeholders consent to a protocol fork increasing the mining rewards; more generally, they will hold onto the hugely wasteful system that empowers them longer than is economically beneficial for users.

1.3 Smart Contracts

Though Bitcoin does allow for smart contracts, most of its opcodes have been historically disabled and the possibilities are limited. Ethereum introduced a smart contract system with some critical differences: their scripting language is Turing complete and they substitute stateful accounts to Bitcoin's unspent outputs.

While emphasis has been put on the Turing complete aspect of the language, the second property is by far the most interesting and powerful of the two. In Bitcoin, an output can be thought of as having only two states: spent and unspent. In Ethereum, accounts (protected by a key) hold a balance, a contract code and a data store. The state of an account's storage can be mutated by making a transaction towards this account. The transaction specifies an amount and the parameters passed to the contract code.

A downside of a Turing complete scripting language for the contracts is that the number of steps needed to execute a script is potentially unbounded, a property which is generally uncomputable.

To address this problem, Ethereum has devised a system by which the miner validating the transaction requires a fee proportional to the complexity and number of steps needed to execute the contract.

Yet, for the blockchain to be secure, *all* the active nodes need to validate the transaction. A malicious miner could include in his block a transaction that he crafted specially to run into an infinite loop and pay himself an exorbitant fee for validating this transaction. Other miners could waste a very long time validating this transaction. Worse, they could just slack and fail to validate it. In practice though, most of the interesting smart contracts can be implemented with very simple business logic and do not need to perform complex calculations.

Our solution is to cap the maximum number of steps that a program is allowed to run for in a single transaction. Since blocks have a size limit that caps the number of transactions per block, there is also a cap on the number of computation steps per block. This rate limitation foils CPU-usage denial-of-service attacks. Meanwhile, legitimate users can issue multiple transactions to compute more steps than allowed in a single transaction, though at a limited rate. Miners may decide to exclude too long of an execution if they feel the included fee is too small. Since the Tezos protocol is amendable, the cap can be increased in future revisions and new cryptographic primitives included in the scripting language as the need develops.

1.4 Correctness

Bitcoin underpins a \$8B valuation with a modest code base. As security researcher Dan Kaminsky explains, Bitcoin looks like a security nightmare on paper. A **C++** code base with a custom binary protocol powers nodes connected to the Internet while holding e-cash, sounds like a recipe for disaster. **C++** programs are often riddled with memory corruption bugs. When they are connecting to the Internet, this creates vulnerabilities exploitable by remote attackers. E-cash gives an immediate payoff to any attacker clever enough to discover and exploit such a vulnerability.

Fortunately, Bitcoin’s implementation has proven very resilient to attacks thus far, with some exceptions. In August 2010, a bug where the sum of two outputs overflowed to a negative number allowed attackers to create two outputs of 92233720368.54 coins from an input of 0.50 coins. More recently, massive vulnerabilities such as the heartbleed bug have been discovered in the OpenSSL libraries. These vulnerabilities have one thing in common, they happened because languages like **C** and **C++** do not perform any checks on the operations they perform. For the sake of efficiency, they may access random parts of the memory, add integers larger than natively supported, etc. While these vulnerabilities have spared Bitcoin, they do not bode well for the security of the system.

Other languages do not exhibit those problems. OCaml is a functional programming language developed by the INRIA since 1996 (and itself based on earlier efforts). Its speed is comparable to that of **C++** and it generally features among the fastest programming languages in benchmarks[12]. More importantly, OCaml is strongly typed and offers a powerful type inference system. Its expressive syntax and semantics, including powerful pattern matching and higher-order modules, make it easy to concisely and correctly describe the type of logic underpinning blockchain based protocols.

OCaml’s semantic is fairly rigorous and a very large subset has been formalized[13], which removes any ambiguity as to what is the intended behavior of amendments.

In addition, Coq, one of the most advanced proof checking software is able to extract OCaml code from proofs. As Tezos matures, it will be possible to automatically extract key parts of the protocol’s code from mathematical proofs

of correctness.

Examples of spectacular software failure abound. The heartbleed bug caused millions of dollars in damages. In 2013, a single bug at high-frequency trading firm Knight capital caused half a billion dollars worth of losses. In 1996, an arithmetic overflow bug caused the crash of Ariane 5, a rocket that had cost \$7B to develop; the cost of the rocket and the cargo was estimated at \$500M.

All of these bugs could have been prevented with the use of formal verification. Formal verification has progressed by leaps and bounds in recent years, it is time to use it in real systems.

2 Abstract Blockchains

Tezos attempts to represent a blockchain protocol in the most general way possible while attempting to remain as efficient as a native protocol. The goal of a blockchain is to represent a single state being concurrently edited. In order to avoid conflicts between concurrent edits, it represents the state as a ledger, that is as a series of transformations applied to an initial state. These transformations are the “blocks” of the blockchain, and — in the case of Bitcoin — the state is mostly the set of unspent outputs. Since the blocks are created asynchronously by many concurrent nodes, a block tree is formed. Each leaf of the tree represents a possible state and the end of a different blockchain. Bitcoin specifies that only one branch should be considered the valid branch: the one with the greatest total difficulty. Blocks, as their name suggests, actually bundle together multiple operations (known as transactions in the case of Bitcoin). These operations are sequentially applied to the state.

2.1 Three Protocols

It is important to distinguish three protocols in cryptoledgers: the network protocol, the transaction protocol, and the consensus protocol.

The role of the meta shell is to handle the network protocol in as agnostic a way as possible while delegating the transaction and consensus protocol to an abstracted implementation.

2.1.1 Network Protocol

The network protocol in Bitcoin is essentially the gossip network that allows the broadcasting of transactions, the downloading and publishing of blocks, the discovery of peers, etc. It is where most development occurs. For instance, bloom filters were introduced in 2012 through BIP0037 to speed up the simple payment verification for clients which do not download the whole blockchain.

Changes to the network protocol are relatively uncontroversial. There may be initial disagreements on the desirability of these changes, but all parties interests are fundamentally aligned overall.

These changes do not need to happen in concert either. One could devise a way to integrate Bitcoin transactions steganographically into pictures of cats

posted on the Internet. If enough people started publishing transactions this way, miners would start parsing cat pictures to find transactions to include in the blockchain.

While a healthy network requires compatibility, competing innovation in the network protocol generally strengthens a cryptocurrency.

2.1.2 Transaction Protocol

The transaction protocol describes what makes transactions valid. It is defined in Bitcoin, for instance, through a scripting language. First, coins are created by miners when they find a block. The miner then attaches a script to the coins that he mined.

Such a script is known as an “unspent output”. Transactions combine outputs by providing arguments for which their scripts evaluate to true. These arguments can be thought of keys and the scripts as padlocks.

In simple transactions, such scripts are merely signature-checking scripts but more complex scripts can be formed. These outputs are added up and allocated among a set of new outputs. If the amount of output spent is greater than the amount allocated, the difference can be claimed by the miner.

Changes to the transaction protocol are more controversial than changes to the network protocol. While a small group of people could unilaterally start using the cat-picture broadcast algorithm, changing the transaction protocol is trickier. Such changes typically do not affect the block validity and thus only require the cooperation of a majority of the miners. These are generally referred to as “soft-fork”.

Some relatively uncontroversial changes still stand a chance to be implemented there. For instance a fix to the transaction malleability issue would be a transaction protocol level change. The introduction of Zerocash, also a transaction protocol level change, risks being too controversial to be undertaken.

2.1.3 Consensus Protocol

The consensus protocol of Bitcoin describes the way consensus is built around the most difficult chain and the miner reward schedules. It allows miners to draw transactions from the coin base, it dictates how difficulty changes over time, it indicates which blocks are valid and which are part of the “official” chain.

This is by far the most central and most difficult to change protocol, often requiring a “hard-fork”, that is a fork invalidating old blocks. For instance, the proof of work system, as is the reliance on SHA256 as a proof-of-work system, etc.

2.2 Network Shell

Tezos separates these three protocols. The transaction protocol and the consensus protocol are implemented in an isolated module plugged into a generic

network shell responsible for maintaining the blockchain.

In order for the protocol to remain generic, we define the following interface. We want our blockchain to represent the current “state” of the economy, which we call in Tezos the **Context**. This could include the balances of the various accounts and other informations such as the current block number. Blocks are seen as operators that transform an old state into a new state.

In this respect, a protocol can be described by only two functions:

- **apply** which takes a Context and a block and returns either a valid Context or an invalid result (should the block be invalid)
- **score** which takes a Context and returns a score allowing us to compare various leafs of the blockchain to determine the canonical one. In Bitcoin, we would simply record the total difficulty or the chain inside the Context and return this value.

Strikingly, these two functions alone can implement *any* blockchain based crypto-ledger. In addition, we attach those functions to the context itself and expose the following two functions to the protocol:

- **set_test_protocol** which replaces the protocol used in the test-net with a new protocol (typically one that has been adopted through a stakeholder voter).
- **promote_test_protocol** which replaces the current protocol with the protocol currently being tested

These two procedures allow the protocol to validate its own replacement. While the seed protocol relies on a simple super-majority rule with a quorum, more complex rules can be adopted in the future. For instance, the stakeholders could vote to require certain properties to be respected by any future protocol. This could be achieved by integrating a proof checker within the protocol and requiring that every amendment include a proof of constitutionality.

3 Proof-of-Stake

Tezos can implement any type of blockchain algorithm: proof-of-work, proof-of-stake, or even centralized. Due to the shortcomings of the proof-of-work mechanism, the Tezos seed protocol implements a proof-of-stake system. There are considerable theoretical hurdles to designing a working proof-of-stake systems, we will explain our way of dealing with them.⁶

3.1 Is Proof-of-Stake Impossible?

There are very serious theoretical hurdles to any proof-of-stake system. The main argument against the very possibility of a proof-of-stake system is the

⁶A full, technical, description of our proof-of-stake system is given in the Tezos white paper.

following: a new user downloads a client and connects for the first time to the network. He receives a tree of blocks with two large branches starting from the genesis hash. Both branches display a thriving economic activity, but they represent two fundamentally different histories. One has clearly been crafted by an attacker, but which one is the real chain?

In the case of Bitcoin, the canonical blockchain is the one representing the largest amount of work. This does not mean that rewriting history is impossible, but it is costly to do so, especially as one's hashing power could be used towards mining blocks on the real blockchain. In a proof-of-stake system where blocks are signed by stakeholders, a former stakeholder (who has since cashed out) could use his old signatures to costlessly fork the blockchain — this is known as the nothing-at-stake problem.

3.2 Mitigations

While this theoretical objection seems ironclad, there are effective mitigations. An important insight is to consider that there are roughly two kinds of forks: very deep ones that rewrite a substantial fraction of the history and short ones that attempt to double spend. On the surface there is only a quantitative difference between the two but in practice the incentives, motivations, and mitigation strategies are different.

No system is unconditionally safe, not Bitcoin, not even public key cryptography. Systems are designed to be safe for a given *threat model*. How well that model captures reality is, *in fine*, an empirical question.

3.2.1 Checkpoints

Occasional checkpoints can be an effective way to prevent very long blockchain reorganizations. Checkpoints are a hack. As Ben Laurie points out, Bitcoin's use of checkpoints taints its status as a fully decentralized currency[14].

Yet, in practice, annual or even semi-annual checkpoints hardly seem problematic. Forming a consensus over a single hash value over a period of months is something that human institutions are perfectly capable of safely accomplishing. This hash can be published in major newspapers around the world, carved on the tables of freshmen students, spray painted under bridges, included in songs, impressed on fresh concrete, tattooed on pet ferrets... there are countless ways to record occasional checkpoints in a way that makes forgery impossible. In contrast, the problem of forming a consensus over a period of minutes is more safely solved by a decentralized protocol.

3.2.2 Statistical Detection

Transactions can reference blocks belonging to the canonical blockchain, thus implicitly signing the chain. An attacker attempting to forge a long reorganization can only produce transactions involving coins he controlled as off the last checkpoint. A long, legitimate, chain would typically show activity in a

larger fraction of the coins and can thus be distinguished, statistically, from the forgery.

This family of techniques (often called TAPOS, for “transactions as proof of stake”) does not work well for short forks where the sample is too small to perform a reliable statistical test. However, they can be combined with a technique dealing with short term forks to form a composite selection algorithm robust to both type of forks.

3.3 The Nothing-At-Stake Problem

An interesting approach to solving the nothing-at-stake problem was outlined by Vitalik Buterin in the algorithm Slasher[15]. However, Slasher still relies on a proof of work mechanism to mine blocks and assumes a bound on the length of feasible forks.

We retain the main idea which consists in punishing double signers. If signing rewards are delayed, they can be withheld if any attempt at double spending is detected. This is enough to prevent a selfish stakeholder from opportunistically attempting to sign a fork for the sake of collecting a reward should the fork succeed. However, once rewards have been paid, this incentive to behave honestly disappears; thus, we use a delay long enough for TAPOS to become statistically significant or for checkpointing to take place.

In order to incentivize stakeholders to behave honestly, we introduce a ticker system. A prospective miner must burn a certain amount of coins in order to exercise his mining right. This amount is automatically returned to him if he fails to mine, or after a long delay.

In order to allow stakeholders not to be permanently connected to the Internet and not to expose private keys, a different, signature key is used.

3.4 Threat Models

No system is unconditionally safe, not Bitcoin, not even public key cryptography. Systems are designed to be safe for a given *threat model*. How well that model captures reality is, *in fine*, an empirical question.

Bitcoin does offer an interesting guarantee: it attempts to tolerate amoral but selfish participants. As long as miners do not collude, it is not necessary to assume that any participant is honest, merely than they prefer making money to destroying the network. However, non collusion, a key condition, is too often forgotten, and the claim of Bitcoin’s “trustlessness” is zealously repeated without much thought.

With checkpointing (be it yearly), the same properties can be achieved by a proof-of-stake system.

Without checkpointing proof-of-stake systems cannot make this claim. Indeed, it would be theoretically possible for an attacker to purchase old keys from a large number of former stakeholders, with no consequence to them. In this case, a stronger assumption is needed about participants, namely that a majority of current or former stakeholders cannot be cheaply corrupted into

participating in an attack on the network. In this case, the role “stake” in the proof-of-stake is only to avoid adverse selection by malicious actors in the consensus group.

4 Potential Developments

In this section, we explore some ideas that we’re specifically interested in integrating to the Tezos protocol.

4.1 Privacy Preserving Transactions

One of the most pressing protocol updates will be the introduction of privacy preserving transactions. We know of two ways to achieve this: ring signatures and non-interactive zero-knowledge proofs of knowledge (NIZKPK).

4.1.1 Ring Signatures

CryptoNote has built a protocol using ring signatures to preserve privacy. Users are able to spend coins without revealing which of N addresses spent the coins. Double spenders are revealed and the transaction deemed invalid. This works similarly to the coin-join protocol *without* requiring the cooperation of the addresses involved in obfuscating the transaction.

One of the main advantage of ring signatures is that they are comparatively simpler to implement than NIZKPK and rely on more mature cryptographic primitives which have stood the test of time.

4.1.2 Non Interactive Zero-knowledge Proofs of Knowledge

Matthew Green et al. proposed the use of NIZKPK to achieve transaction untraceability in a blockchain based cryptocurrency. The latest proposition, Zerocash, maintains a set of coins with attached secrets in a Merkle tree. Committed coins are redeemed by providing a NIZKPK of the secret attached to a coin in the tree. It uses a relatively new primitive, SNARKs, to build very small proofs which can be efficiently checked.

This technique is attractive but suffers from drawbacks. The cryptographic primitives involved are fairly new and have not been scrutinized as heavily as the relatively simple elliptic curve cryptography involved in Bitcoin.

Secondly, the construction of these proofs relies on the CRS model. This effectively means that a trusted setup is required, though the use of secure multi-party computation can reduce the risk that such a setup be compromised.

4.2 Amendment Rules

4.2.1 Constitutionalism

While this is more advanced, it is possible to integrate a proof checker within the protocol so that only amendments carrying a formal proof that they re-

spect particular properties can be adopted. In effect this enforces a form of constitutionality.

4.2.2 Futarchy

Robin Hanson has proposed that we vote on values and bet on beliefs. He calls such a system “Futarchy” [16]. The main idea is that values are best captured by a majoritarian consensus while the choice of policies conducive to realizing those values is best left to a prediction market.

This system can quite literally be implemented in Tezos. Stakeholders would first vote on a trusted datafeed representing the satisfaction of a value. This might be, for example, the exchange rate of coins against a basket of international currencies. An internal prediction market would be formed to estimate the change in this indicator conditional on various code amendments being adopted. The market-making in those contracts can be subsidized by issuing coins to market makers in order to improve price discovery and liquidity. In the end, the amendment deemed most likely to improve the indicator would be automatically adopted.

4.3 Solving Collective Action Problems

The collective action problem arises when multiple parties would benefit from taking an action but none benefit from individually undertaking the action. This is also known as the free-rider problem. There are several actions that the holders of a cryptocurrency could undertake to raise its profile or defend it against legal challenges.

4.3.1 Raising Awareness

As of July 2014, the market capitalization of Bitcoin was around \$8B. By spending about 0.05% of the Bitcoin monetary mass every month, Bitcoin could make highly visible charitable donations of \$1M *every single week*. Would, as of 2014, an entire year of weekly charitable donations raise the value of Bitcoin by more than 0.6%? We think the answer is clearly, and resoundingly “yes”. Bitcoin stakeholders would be doing well while doing good.

However, Bitcoin stakeholders are unable to undertake such an operation because of the difficulty of forming large binding contracts. This type of collective action problem is solved in Tezos. A protocol amendment can set up a procedure by which stakeholders may vote every month on a few addresses where 0.05% of the monetary mass would be spent. The stakeholder’s consensus might be to avoid dilution by voting on an invalid address, but it could also be that the money would be better spent as a charitable gift.

4.3.2 Funding Innovation

Financing of innovation would also be facilitated by incorporating bounties directly within the protocol. A protocol could define unit tests and automatically

reward any code proposal that passes these tests.

Conversely, an innovator designing a new protocol could include a reward to himself within the protocol. While his protocol could be copied and the reward stripped, the stakeholder's consensus would likely be to reward the original creator. Stakeholders are playing a repeated game and it would be foolish to defect by refusing a reasonable reward.

Conclusion

We've presented issues with the existing cryptocurrencies and offered Tezos as a solution. While the irony of preventing the fragmentation of cryptocurrencies by releasing a new one does not escape us, Tezos truly aims to be the *last* cryptocurrency.

No matter what innovations other protocols produce, it will be possible for Tezos stakeholders to adopt these innovations. Furthermore, the ability to solve collective action problems and easily implement protocols in OCaml will make Tezos one of the most reactive cryptocurrency.

References

- [1] Peter Suber. Nomic: A game of self-amendment. <http://legacy.earlham.edu/~peters/writing/nomic.htm>, 1982.
- [2] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. <https://bitcoin.org/bitcoin.pdf>, 2008.
- [3] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. <https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-White-Paper>, 2014.
- [4] Nicolas van Saberhagen. Cryptonote v 2.0. <https://cryptonote.org/whitepaper.pdf>, 2013.
- [5] Matthew Green et al. Zerocash: Decentralized anonymous payments from bitcoin. <http://zerocash-project.org/media/pdf/zerocash-extended-20140518.pdf>, 2014.
- [6] Thomas Schelling. *The Strategy of conflict*. Cambridge: Harvard University Press, 1960.
- [7] Bitcoin Wiki. Weaknesses. https://en.bitcoin.it/wiki/Attacks#Attacker_has_a_lot_of_computing_power, 2014.
- [8] Gaving Andresen. Centralized mining. <http://bitcoinfoundation.org/centralized-mining/>, 2014.
- [9] Bitcoin Wiki. Tragedy of the commons. https://en.bitcoin.it/wiki/Tragedy_of_the_Commons, 2014.

- [10] Bitcoin Wiki. Dominant assurance contracts. https://en.bitcoin.it/wiki/Dominant_Assurance_Contracts, 2014.
- [11] Simon de la Rouviere. Not actually capped at 100 billion? <https://github.com/dogecoin/dogecoin/issues/23>, 2013.
- [12] Debian project. Computer language benchmarks game. <http://benchmarksgame.alioth.debian.org/u32/index.html>, 2014.
- [13] Scott Owens. A sound semantics for ocaml light. <http://www.cl.cam.ac.uk/~so294/ocaml/paper.pdf>, 2008.
- [14] Ben Laurie. Decentralised currencies are probably impossible, but let's at least make them efficient. <http://www.links.org/files/decentralised-currencies.pdf>, 2011.
- [15] Vitalik Buterin. Slasher: A punitive proof-of-stake algorithm. <https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm/>, 2014.
- [16] Robin Hanson. Shall we vote on values, but bet on beliefs? <http://mason.gmu.edu/~rhanson/futarchy2013.pdf>, 2013.