Elasticoin: Low-Volatility Cryptocurrency with Proofs of Sequential Work

Yuhao Dong
University of Waterloo
yd2dong@uwaterloo.ca

Raouf Boutaba
University of Waterloo
rboutaba@uwaterloo.ca

***Abstract*—Blockchain-based cryptocurrencies have gained increasing adoption in recent years, and many hope that they may usher in a new era of decentralized electronic money. Unfortunately, they perform the core functions of money quite poorly due to their extremely volatile market value. On the other hand, blockchain "stablecoins" aiming to reduce this volatility, usually through a peg to an external currency like the US dollar, tend to greatly sacrifice decentralization of the money supply that make cryptocurrencies so attractive in the first place.**

Elasticoin is a novel currency issuance algorithm which greatly reduces price volatility by using the cryptographic puzzle of *proofs of sequential work* to fix the cost of minting a coin to sequential computation time. This causes coin supply to be highly elastic, quickly responding to demand for new coins and greatly dampening price swings. We argue that Elasticoin's fixed-minting-cost approach to low volatility has significant advantages over using pegs or explicit measurement of demand to adjust supply.

I. Introduction

Decentralized, blockchain-based cryptocurrencies, pioneered by Bitcoin [1], are gaining increasing acceptance as alternative payment methods. Major websites and payment processors now accept payment in cryptocurrencies, interest in cryptocurrency investment powers an increasingly sophisticated trading industry, and the combined market capitalization of cryptocurrencies has reached more than 100 billion US dollars. Convenient, electronic money that is nevertheless completely decentralized and independent of centralized monetary authorities will likely continue to supply ample demand for blockchain cryptocurrencies.

Yet despite rising popularity, decentralized cryptocurrencies fail to reliably perform certain core functions of money as stores of value and units of account. Cryptocurrency payment processors typically convert immediately into traditional fiat money, few people store their savings in a cryptocurrency, and prices for services are rarely ever quoted in, say, bitcoins or litecoins. We believe that this is almost entirely due to the *volatile value* of cryptocurrencies — the value of a cryptocurrency unit can swing as much as 15% in a day [2], severely hampering usefulness as money — which is in turn caused by the completely demand-agnostic way new cryptocurrency units are issued (Ethereum, for example, simply "prints" 2 ETH per block [3]).

We propose Elasticoin, a low-volatility currency issuance algorithm easily implementable on various token systems, which fixes the cost of minting a coin to sequential processing time using the novel cryptographic puzzle of *proof of sequential work*. Elasticoin makes coin supply elastic through the invisible hand of the market: when demand for the coin pushes its price higher than the cost of minting, profit-seeking minters will then provide an effectively unlimited supply of new coins, and when demand is low, the supply of new coins will naturally dry up. We argue that this approach is superior to "stablecoins" that attempt to rigidly peg a cryptocurrency to an off-chain asset like the dollar, or flexible-supply schemes that globally adjust the rate of money creation based on some mechanism of measuring economic activity.

In this paper, we describe how the simple yet robust minting mechanism of Elasticoin works, centered on a non-interactive proof of sequential work we derive from existing work [4] using the well-known Fiat-Shamir heuristic [5]. We also compare Elasticoin's stability and security against other coin issuance models, including those of both "stablecoins" with a hard peg to an established currency and alternative ways of making coin supply elastic.

II. Design

Like other low-volatility cryptocurrencies, Elasticoin is intended to be implemented in a secondary token on an existing underlying blockchain, like an Ethereum ERC20 token, rather than issued as a primary cryptocurrency to incentivize a consensus mechanism. This allows us to sidestep cryptoeconomic concerns that force primary cryptocurrencies to have security-like rather than currency-like features, like the need for a stable issuance rate to stabilize miner incentives.

However, Elasticoin's mechanism for stabilizing currency value differs from all other low-volatility cryptocurrencies. Instead of directly manipulating the money supply or using a fixed issuance schedule, we target *fixed-cost minting*: letting the cost of creating a new unit of currency stay roughtly the same. This automatically gives us an extremely elastic supply of new currency, as minters will create new currency units to prevent the price from rising significantly above minting cost, while when prices are low, supply of new coins will dwindle to zero. Unlike pegging, we avoid the difficult problem of trustlessly measuring real-world economic data. Unlike proposals to reduce volatility by algorithmically manipulating money supply, we need no potentially gameable and difficult-to-tune algorithmic demand estimators.

It's important to know that unlike "stablecoin" mechanisms targetting a fixed exchange rate with an external currency, fixed-

978-1-7281-1328-9/19/$31.00 ©2019 IEEE

cost minting cannot give us a perfectly stable coin value, only an *approximate price ceiling*, as decreases in demand sharp enough to completely eliminate any demand for newly-minted currency will still cause drops in coin value. But even this will drastically reduce volatility. Firstly, fluctuations in the expectation of very low-probability demand shocks ("Bitcoin replaces the US dollar") cause great fluctuations in the present value of a fixed-supply cryptocurrency, but Elasticoin eliminates the effect of such hypothetical demand shocks on price, eliminating this source of present volatility. Furthermore, self-reinforcing bubbles, where demand drives rocketing prices, which then drives even more speculative demand, until a spectacular crash, will not happen, as the price ceiling prevents any irrational overpricing from happening in the first place and dashes any speculators' hopes of the price going "to the moon".

Of course, fixed-cost minting is far from trivial to implement — it's not obvious how to design a trustless minting mechanism that has a relatively fixed cost. Elasticoin solves this using a novel mechanism that fixes the cost of minting a new coin at *one day of sequential computation on the fastest processor*. In this section, we will first look at why we choose this metric as our fixed minting cost, then look at *non-interactive proofs of sequential work*, before discussing in detail how Elasticoin uses this cryptographic primitive to construct its minting mechanism.

A. Why sequential computation time?

"One day of sequential computation on the fastest processor" seems like a strange metric for defining a relatively fixed cost. There are, however, two important reasons why we chose it.

First of all, most "obvious" measures of cost cannot be easily measured trustlessly, as they refer to phenomena that the blockchain protocol cannot observe directly. For example, consumer price indices, human labor, currency exchange rates, etc are all metrics a real-life central bank may use to stabilize a currency, but decentralized cryptocurrencies simply have no satisfactory way of "measuring" these facts. Generally, a trusted, centralized oracle must be used, significantly weakening the case for decentralized cryptocurrencies. More decentralized mechanisms based on crowdsourcing such data have been proposed, but game-theoretical problems such as very cheap bribing attacks [6] pose very serious challenges that may or may not ever be solved. On the other hand, time and computation are easy to objectively, trustlessly measure, especially since all popular public blockchains assume some sort of weak clock synchronicity in their security model. In fact, the first blockchain cryptocurrency, Bitcoin, issues new coins with exactly these two measures (proof-of-work computation, with time used for difficulty adjustment).

TABLE I: Price of renting servers across time

Provider	2006	2010	2018
OVH (cheapest)	€69/mth	€69.99/mth	€59.99/mth
LeaseWeb (cheapest)	€59/mth	€24/mth	€29.99/mth
Amazon AWS (2 vCPUs)	$0.10/hr	$0.085/hr	$0.096/hr

Secondly, sequential computation time does have a fairly stable value, both intuitively and empirically. Intuitively, the subjective value of "tying up a CPU core for a day" seems to be fairly stable — technological advancements increase the amount of work a processor core can do per unit of time, but at the same time software advancements increase the amount of work we want processors to do. Empirically, the cost of renting a dedicated server for a month, or a cloud computing instance for an hour, has stayed pretty much the same since for at least the last decade, despite great changes in both hardware and software. Table I shows actual prices for three large hosting companies (OVH, LeaseWeb, AWS) in 2006, 2010, and 2018 (retrieved through the Internet Archive). It is clear that market prices for *up-to-date* computation time do not change drastically.

B. Non-interactive proofs of sequential work

We now describe non-interactive proof of sequential work (NiPoSW), a computational puzzle which Elasticoin relies on. NiPoSW is based on "Simple Proof of Sequential Work" (SPoSW) [4], an *interactive* protocol, itself a refinement of the original concept by Mahmoody et al. [7] of a publicly verifiable proof of work. In an interactive proof of sequential work, a prover gets an arbitrary "statement" χ, a time parameter N, and access to a hash function H. After querying H at least N times sequentially, the prover can then respond to challenges from a verifier and convince that verifier that it actually has completed the required work.

SPoSW's construction is based upon building a hash DAG, but the specifics are not really important for the purposes of understanding NiPoSW. In SPoSW, given a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, the following steps are defined:

- **Statement**: $\mathcal{V}$ samples a random binary string χ, sending it to $\mathcal{P}$
- **Compute**: $\mathcal{P}$ computes (by doing N sequential computation) a proof $(\phi, \phi_{\mathcal{P}}) := \mathsf{PoSW}(\chi, N)$. ϕ is sent to $\mathcal{V}$, but $\phi_{\mathcal{P}}$ is sent to $\mathcal{P}$.
- **Challenge**: $\mathcal{V}$ samples a random challenge γ consisting of many individual challenges $\gamma_1, \ldots, \gamma_t$.
- **Open**: $\mathcal{P}$ computes $\tau := \mathsf{open}(\chi, N, \phi_{\mathcal{P}}, \gamma)$ and sends it to $\mathcal{V}$.
- **Verify**: $\mathcal{V}$ computes and outputs $\mathsf{verify}(\chi, N, \phi, \gamma, \tau) \in \{\text{accept}, \text{reject}\}$.

For our purposes, we need a *non-interactive* proof of sequential work: a prover must be able to show that she has completed N work, starting from a random statement χ, with a static proof ϕ that can be checked offline without any interaction.

We use the well-known Fiat-Shamir heuristic [5] to transform SPoSW to its non-interactive version, NiPoSW, which exposes only two steps:

- **Solve**: $\mathcal{V}$ computes $\phi = \mathsf{Solve}(\chi, n)$, where χ is a random "statement", which *cannot be influenced by* $\mathcal{V}$, n is a difficulty parameter where $N = 2^n$ work must be done, and ϕ is the resulting proof of sequential work. $\mathcal{V}$ then broadcasts ϕ.
- **Verify**: Any $\mathcal{P}$ can then run $\mathsf{Verify}(\phi, \chi, n) \in \{\text{accept}, \text{reject}\}$ to check the validity of the proof.

More specifically, we use χ, combined with the hash function H, as a source of randomness to eliminate interaction. To eliminate interaction in Solve, we combine compute, challenge, and verify:

$$\phi = \mathsf{Solve}(\chi, n) = (\phi', \tau)$$

where

$$\tau = \mathsf{open}(\chi, 2^n, \phi'_{\mathcal{P}}, \Gamma(\chi))$$

where $\Gamma(\chi) = \{H(\chi||1), \ldots, H(\chi||t)\}$ and

$$(\phi', \phi'_{\mathcal{P}}) = \mathsf{PoSW}(\chi, 2^n)$$

Eliminating interaction from verification is straightforward:

$$\mathsf{Verify}(\phi = (\phi', \tau), \chi, n) = \mathsf{verify}(\chi, N, \phi', \Gamma(\chi), \tau)$$

C. Minting algorithm

Now that we have a primitive for publicly proving completion of sequential work, we can now construct Elasticoin's minting algorithm. Unlike most cryptocurrency issuance protocols, Elasticoin minting is a two-step process, where minters first *register* puzzles on the blockchain to later *solve* with a separate transaction.

To register a puzzle, a minter broadcasts a specially formatted transaction containing $\mathcal{A}$, the identity of the minter, generally a public key or other form of cryptocurrency "address". Once the transaction is confirmed on the blockchain, the minter notes t, the block height at which the transaction is embedded, and calculate

$$\chi_{\mathcal{A}} = \mathcal{A}||R_t$$

, where R_t is a public value unpredictable until the creation of block t, such as a block header hash.

The minter then begins work on a puzzle with seed $\chi_{\mathcal{A}}$ and a difficulty parameter N of her own choice. After solving the puzzle, she broadcasts another special transaction containing the solution to the puzzle,

$$\phi = \mathsf{Solve}(\chi_{\mathcal{A}}, N)$$

which entitles her to claim a predictable reward $r(N)$.

Through this process, the minter publicly proves that during the time after the puzzle is registered and before it's solved, she's done $\Omega(2^N)$ sequential operations. She cannot cheat by precomputing a solution, as $\chi_{\mathcal{A}}$ cannot be calculated in advance.

We now look at how rewards are determined; Elasticoin's reward function is what powers the "magic" that drives its minting cost to track sequential computation time.

We keep track of a variable $v_{\max}$, representing "the speed of the fastest minter ever observed on the blockchain, measured in operations per day". Every time a solution transaction is posted to the blockchain, at block height t_{solve}, solving a puzzle posted at height t_{register}, with difficulty parameter N, we update $v_{\max}$:

$$v_{\max} \leftarrow \max(v_{\max}, v)$$

where

$$v = \frac{2^N}{C(t_{\text{solve}} - t_{\text{register}})}$$

where C is the number of block intervals per day (for Bitcoin, $C = 144$)

Rewards are then calculated as

$$r(N) = \frac{2^N}{v_{\max}} \cdot \frac{v}{v_{\max}}$$

III. Evaluation and related work

In this section we qualitatively evaluate the properties of Elasticoin, first discussing its incentives and security, and then comparing it with existing work on stabilizing the price of a cryptocurrency. We argue that Elasticoin, overall, can power a low-volatility cryptocurrency with more robustness and decentralization than any existing proposal.

A. Incentives and security

Given that $v_{\max}$ is a reasonable estimate of the amount of sequential work the fastest processor generally available can do, it is obvious that for the fastest processor, minting one full coin requires a whole day of sequential work, achieving our goal. Why, however, will $v_{\max}$ be a good estimate? And will processors slower than the very fastest get reasonable rewards?

Importantly, $v_{\max}$ never decreases; we assume that technology never goes backwards, and recent processors are at least as fast as old ones. Therefore, we can dismiss attacks attempting to reduce $v_{\max}$. On the other hand, faking a high value of $v_{\max}$ is impossible without actual advances in computational speed, as puzzles cannot be precomputed. Finally, attempting to prevent $v_{\max}$ from changing is also practically impossible. Even if almost all the minters form a cartel and solve puzzles purposefully slowly, *one* honest minter is enough to keep $v_{\max}$ correct.

Processors slower than $v_{\max}$ take longer to solve puzzles of the same difficulty, yet are doubly penalized by the $v/v_{\max}$ term. This intentionally disincentivizes minting using suboptimal hardware, leading to emergent standardization on the most efficient way to mint at any given time, reducing market uncertainty about the cost of minting and thus the price. Furthermore, the term all but eliminates the possibility of using slow computation with volatile but usually cheap prices, such as spare computational cycles on a system used for other purposes, further reducing volatility.

We see that our minting mechanism does in fact cause minting costs to quickly converge to that of one day of computation, on the fastest NiPoSW solver, per Elasticoin. Two potential problems remain:

- **Fastest NiPoSW solver may not be a general-purpose processor**: Special-purpose hardware, such as an ASIC design, that solves NiPoSW faster than top-of-the-line generic CPUs may emerge. This would break our design goal of tying minting cost to time on a fast generic CPU. Fortunately, it's not difficult to parameterize NiPoSW with a hash function designed to be bottleneck on *memory bandwidth*, such as Argon2 [8]. Sequential memory bandwidth is already something CPU manufacturers heavily optimize for, and it is very unlikely that cost-effective ASICs can have better performance. There is also no "prize" for

minters who manage to bump $v_{\max}$ upwards, so it's unlikely that significant resources would be devoted to devising specialized NiPoSW solvers.

- **Time measurements may be inaccurate or insecure**: Elasticoin uses block height as a clock; if instead blocks appear at erratic schedules or can be delayed at will by adversaries, the entire mechanism fails. For example, a malicious network-based adversary might systematically delay blocks, increasing the time between new blocks, leading to overestimation of $v_{\max}$. More problematically, since $v_{\max}$ can never go down, one fraudulently high value for $v_{\max}$ will stick around forever.
  However, almost all blockchain protocols to some extent assume that the *long-run average* of block intervals stays roughly the same — deviations cannot last forever without breaking the blockchain itself. Blockchain-specific Elasticoin implementations can easily set a minimum number of block intervals that must elapse before solutions to puzzles are accepted, ensuring that time measurements are accurate and difficult to manipulate.

B. Pegged stablecoins

Most existing attempts at reducing cryptocurrency volatility are **stablecoins** that go "all the way" — they peg a cryptocurrency with a real-world currency, typically the US dollar, and thus achieve perfect price stability as measured in that currency. If stablecoins can really achieve robust stability without sacrificing any of the desirable properties of a cryptocurrency, they would truly be the holy grail of low volatility. Unfortunately, as we will now discuss, stablecoins of various shapes and forms all have worrying fundamental problems, in stark contrast to the robust minting mechanism of Elasticoin.

The simplest and most common type of stablecoin uses a **centralized currency board**. Tether, the stablecoin with by far the most usage, belongs in this category. We have a centralized bank, taking deposits in, say, US dollars, and issuing coins that are simply blockchain-traded IOUs that can be redeemed at the bank for a dollar each. This sort of arrangement is common in fiat currencies robustly pegged to another one; one Hong Kong dollar, for example, is in reality an IOU for 0.128 US dollars, backed by the Hong Kong Monetary Authority. With a trustworthy and reliable bank, no sort of economic shock could ever permanently break the peg.

The biggest problem, of course, is counterparty risk: if the currency board is compromised, dishonest, or fails, the currency will instantly collapse. It also trusts currency boards with a vast amount of cash — it would be very hard to resist the temptation to invest or speculate with the backings, as Tether is suspected of doing [9], further increasing counterparty risk.

Of course, **decentralized stablecoins** aiming to eliminate counterparty risk do exist. One example uses *over-collateralized cryptoasset debt*. For example, Dai [10] is a stablecoin pegged to the US dollar, backed by an entirely decentralized smart contract running on the Ethereum blockchain, using *on-chain* reserves of assets like Bitcoins and Ethers rather than dollars. The currency is over-collateralized: each coin is issued only against a debt much more than its face value in cryptoassets. One coin may be issued only when a user deposits $2 worth of ETH and can be redeemed to get the ETH back. To maintain stability, whenever the price falls below the peg, collateral would be used to buy up coins. The problem is that if the price of the collateral crashes fast enough, the currency will quickly lose its backing and crash. Other approaches to smart-contract based stablecoins exist, such as seigniorage shares [11], Basis [12] and many others [13], but similar "black swan" scenarios are generally acknowledged to exist for all of them, hinting that since the assets the coin is pegged to cannot be owned on-chain, a decentralized stablecoin robust to economic shocks might be impossible.

More importantly, even "decentralized" stablecoins must rely on a reliable, uncorruptible feed of exchange rates with an off-chain asset: Dai, for example, has a pool of oracles posting the ETH/USD exchange rate. This is very difficult to decentralize, and proposals to use some sort of voting based on the game-theoretic concept of Schelling points [14] to replace this oracle tend to fall prey to easy economic attacks [6].

C. Non-pegged low-volatility coins

Previous proposals for lower-volatility cryptocurrencies that are not pegged to an off-chain asset do exist, although they are rare and none seem to have been fully described or implemented. Suggestions of systems that measure the "economic activity" of a blockchain to drive an algorithm controlling the global money supply have been floated [15], but the most fleshed-out proposal might be in [16]. There, a model is gradually refined that *endogenously* estimates the price of Bitcoin from only on-chain parameters such as mining difficulty, transaction volume, etc. This approximate price feed can then be fed into a usual stablecoin mechanism like seigniorage shares to derive an "approximate stablecoin" that is truly decentralized, no longer depending on any trusted price information or attempting to keep a strict peg. Other approaches more dissimilar to stablecoin algorithms exist [17], yet they need similarly complex economic models.

These systems attempt to achieve the same result as Elasticoin (elastic coin supply), but using a complex algorithm with many tunable, and possibly overfitted, parameters to control how much money to print. On the other hand, leveraging proofs of sequential work allows Elasticoin to let an elastic coin supply *emerge* from rational decisions by individual minters, a much more general and robust method.

IV. Conclusion

In this paper, we described Elasticoin, a novel, decentralized, and robust way of stabilizing the price of a cryptocurrency without using any direct money supply manipulation in the style of central banks. This is possible due to the use of proofs of sequential work, cryptographic puzzles proving that a certain amount of sequential computation was completed, extended to a non-interactive setting suitable for cryptocurrency minting. We compare Elasticoin to existing proposals for both pegged stablecoins and non-pegged low-volatility cryptocurrencies, concluding that Elasticoin offers a combination of robustness, decentralization, and elegance not found in competing systems.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] "Cryptocurrency market capitalization," CoinMarketCap. [Online]. Available: https://coinmarketcap.com/

[3] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger. ethereum project yellow paper 151 (2014)," 2014.

[4] B. Cohen and K. Pietrzak, "Simple proofs of sequential work," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 451–467.

[5] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology—CRYPTO'86*. Springer, 1986, pp. 186–194.

[6] V. Buterin, "The p plus epsilon attack," *Ethereum Blog*, 2015. [Online]. Available: https://blog.ethereum.org/2015/01/28/p-epsilon-attack/

[7] M. Mahmoody, T. Moran, and S. Vadhan, "Publicly verifiable proofs of sequential work," in *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*. ACM, 2013, pp. 373–388.

[8] A. Biryukov, D. Dinu, and D. Khovratovich, "Argon2: new generation of memory-hard functions for password hashing and other applications," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016, pp. 292–302.

[9] J. M. Griffin and A. Shams, "Is bitcoin really un-tethered?" 2018. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3195066

[10] "The dai stablecoin system," Maker Team, 2017. [Online]. Available: https://makerdao.com/whitepaper/Dai-Whitepaper-Dec17-en.pdf

[11] R. Sams, "A note on cryptocurrency stabilization: Seignorage shares," 2015. [Online]. Available: https://github.com/rmsams/stablecoins

[12] N. Al-Naji, J. Chen, and L. Diao, "Basis: A price-stable cryptocurrency with an algorithmic central bank," 2018. [Online]. Available: https://blog.bitmex.com/wp-content/uploads/2018/06/basis_whitepaper_en.pdf

[13] M. Iwamura, Y. Kitamura, T. Matsumoto, and K. Saito, "Can we stabilize the price of a cryptocurrency?: Understanding the design of bitcoin and its potential to compete with central bank money," *Understanding the Design of Bitcoin and Its Potential to Compete with Central Bank Money (October 25, 2014)*, 2014.

[14] V. Buterin, "Schellingcoin: A minimal-trust universal data feed," *Ethereum Blog*, 2014. [Online]. Available: https://blog.ethereum.org/2014/03/28/schellingcoin-a-minimal-trust-universal-data-feed/

[15] "Hard problems of cryptocurrency," Ethereum Wiki. [Online]. Available: https://github.com/ethereum/wiki/wiki/Problems

[16] V. Buterin, "The search for a stable cryptocurrency," *Ethereum Blog*, 2014. [Online]. Available: https://blog.ethereum.org/2014/11/11/search-stable-cryptocurrency/

[17] K. Saito and M. Iwamura, "How to make a digital currency on a blockchain stable," *arXiv preprint arXiv:1801.06771*, 2018.