On Space-Scarce Economy In Blockchain Systems

No Author Given

No Institute Given

Abstract. In this paper we study space-scarce economy in massively replicated open blockchain systems. In these systems, such as Bitcoin, memory to hold a current state snapshot needed to validate transactions becomes the most scarce resource eventually. The issue is even more critical for blockchain systems used to store data (votes, certificates, logs etc.). Uncontrolled state size growth could lead to security issues, such as denial-of-service attacks. Only technical solutions, not economic, have been proposed to tackle this problem to the moment. In contrast, we propose to add a new component to a transaction fee scheme based on how much additional space will be needed for new objects created in result of transaction processing and for how long they will live in the state. We provide three possible options towards implementing the new fee component, namely *prepaid outputs*, *postpaid outputs* and *scheduled payments*. We provide an analysis of the model with respect to all the three options. We show that the state growth could be bounded by a fee factor, miners are getting additional stable rewards and lost coins are being taken back into circulation eventually.

1 Introduction

Bitcoin [1] was introduced in 2008 by S. Nakamoto as a purely peer-to-peer version of electronic cash with a ledger written into blockchain data structure securely replicated by each network node. Security of the scheme is relied on mining process. If majority of miners are honest, then Bitcoin meets its security goals as formal analysis [2] shows. For work done a miner is claiming a reward which consists of two parts. First, some constant number of bitcoins are created out of thin air according to a predefined and hard-coded token emission schedule. Second, a miner claims fees for all the transactions included into the block. A transaction fee is set by a user during transaction creation. Transaction fees are useful for an existing cryptocurrency economy for two reasons:

1. *Incentivization of miners.* A rational Bitcoin miner does not include all the valid transactions into blocks as, due to the increased chances of orphaning a block, the cost of adding transactions to a block could not be ignored [3,4]. As shown in [4], even in absence of block size limit, Bitcoin fee market is healthy and the miners surplus is maximized at a finite quantity of block space. Thus the miner is incentivized to produce a block of a limited size.

This means that only a subset of transactions which provides enough value to a miner will be included in a block. A paper [4] provides a procedure to calculate transaction fee based on block propagation time.

2. *Limit resources usage and prevent spam.* Besides of network utilization, transaction processing requires a miner to spend some computational resources. For most of the cryptocurrencies, a transactional language is limited (with Bitcoin Script [5] being one of the most limited), thus a number of CPU cycles needed to process a transaction is strictly bounded and corresponding computational costs are not directly considered. In contrast, in cryptocurrencies supporting smart contract languages, such as [6,7,8], transaction processing may require a lot of computations, and computational costs are included in transaction fee. This cost is specific to concrete transactional language and is out of scope of this paper.

A transaction in Bitcoin fully spends outputs from previous transactions, and also creates new outputs of user-defined values. A notable and the only exception is a coinbase transaction of a block which creates fixed amount of money out of thin air and also claims transaction fees without referring to any outputs (a fee for a non-coinbase transaction is sum of claimed outputs values minus sum of values for created outputs). A node is checking a transaction in Bitcoin by using a set of unspent outputs. In other cryptocurrencies a representation of a *state* needed to validate and process an arbitrary transaction could be different (for example, in Ethereum [9] such structure is called the *world state* and fixed by the protocol). To process a transaction quickly, the state (or most accessed part of it) should reside in random-access memory. Once it becomes too big to fit into RAM an attacker can perform denial-of-service attacks against cryptocurrency nodes. For example, during attacks on Ethereum in Autumn, 2016, an attacker added about 18 million accounts to the state (whose size was less than 1 million accounts before the attack) and then performed succesful denial-of-service attacks against the nodes[10]. Similarly, in 2013 a denial-of-service attack against serialized transactions residing in a secondary storage (HDD or SSD) was discovered in Bitcoin[11].

The main purpose of this paper is to consider a new mandatory component in a transaction fee scheme reflecting state growth. In all known cryptocurrencies of today, an element of the state once created lives possibly forever without paying anything for that. This leads to continuously increasing state (we point to Bitcoin unspent transaction outputs (UTXO) set size as an example [12]). Moreover, state may grow fast during spam attack, for example, 15 million outputs were quickly put into UTXO set during spam attacks against Bitcoin in July 2015 [13], and most of these outputs are not spent yet. The paper [14] is proposing a technical solution for non-mining nodes where only miners hold the full state (assuming that they can invest money in random-access memory of sufficiently big capacity), while other nodes are checking proofs of state transformations generated by miners, and size of a proof (in average and also in a worst case) is about $log(S)$ in regards with a state size S. Nevertheless, big state could lead to centralization of mining or SPV mining [15], and these concerns

should be addressed. Also, there is an increasing demand to use a blockchain as a data storage, and storing permanently objects in the state without a cleaning procedure is not a viable option.

We propose an economic solution to the problem of unreasonable state growth (such as spam attacks, or objects not being using anymore but still living in the blockchain). The solution is a new mandatory fee component. We state that a user should pay fee for both the additional space needed to store objects created by a transaction, and also for lifetime of new bytes. This model is usual for cloud storage services where users pay for gigabytes of data per month. We provide a possibility for miners to control their storage requirements by changing a fee factor. Later in this paper we will refer to this new fee component as to a *space-time fee*.

Proposed fee regime is promoting money circulation in the blockchain economy. The limited lifetime of a state element also leads to lost coins being taken back into circulation (supposedly by miners).

Summarizing, we study an economy where quick-access storage of a node in a massively-replicated system becomes the most scarce system resource eventually. Thus we call such an economy a *space-scarce economy*.

1.1 Assumptions

Here we provide assumptions our model is based on:

- all the fees for a block are going to a single miner like in Bitcoin. There are proposals to share the rewards for a block within a group of miners, for example in [16,17], and they are out of scope of the paper.
- a state is a set of unspent outputs. An output is not modifiable so can be only created and then spent at whole.
- an output is protected by a spending condition which is defined as a logical formula. Predicates in the formula can refer to properties of a blockchain (for example, its current height available via variable $Height$), spending transaction tx and the output out itself. We assume that it is possible to compare two scripts, and also it is possible to determine whether the spending transaction contains an output with a given property. For example, $tx.has\_output(script = out.script)$ evaluates to true if the spending transaction contains an output with the same script as the output has. Note that Bitcoin Script is too limited to support scripts comparison as well as using the spending transaction and the output to spend in a spending condition.
- for simplicity, we assume that a block is of a finite size but all the transactions a miner has at a moment of block generation can be packed into it, if otherwise is not stated explicitly.
- time is measured via *height* which is a number of blocks since an initial block (a genesis block) till a block of interest.
- all anyone-can-spend outputs are collected by miners immediately as they appear.

– we are considering minimal mandatory fees in the paper. All the nodes are checking that a fee paid by a transaction is not less than a minimum and rejecting the whole block if it contains a transaction violating fee rules. Thus a fee regime is considered as a part of consensus protocol in our work. A user can pay more than the minimum to have a higher priority for a transaction of interest.

1.2 Structure of the Paper

The paper is organized as follows. A design of our new fee component is provided in Section 2. The model then is analyzed in Section 3. In Section 4 we observe related work, and in Section 5 we shape a plan for further research.

2 The Model

In our model transaction fee f consists of 3 parts: *validation and processing cost* f_v, *propagation cost* f_p and *state space-time difference cost* f_s, and f_s is non-negative:

$$f = f_v + f_p + f_s, f \geq 0 \tag{1}$$

Validation and propagation costs are non-negative and assumed to be independent from the space-time fee. It is possible for f_s to be negative since a transaction may decrease size of the state. The space-time fee f_s depends on size of outputs to be marked as spent, remaining lifetimes for them, newly created outputs and their time to live. If a transaction is freeing some space-time $T_{cleared}$ and claiming space-time T_{filled}, then the resulting space-time fee can be defined as $f_s = K \cdot (T_{filled} - T_{cleared})$, where K is a space-time price. As time in a blockchain system is associated with blocks (precisely, height of a block), and space is measured in byte, K is to be a price of byte per block. We assume that miners can steadily change this price like they change "gas" limit per block in Ethereum. For example, a miner can be given a right to raise or lower K value by 0.1% per block. Or miners can vote during an epoch of $1,000 - 2,000$ blocks on whether to raise or lower K for the next epoch by 1%, or leave it intact.

We consider three options for the fee to be charged. Concretely, *prepaid outputs* are defined in Section 2.1, *postpaid outputs* are described in Section 2.2 and *scheduled payments* are proposed in Section 2.3.

2.1 Prepaid Outputs

First option we are going to analyze is prepaid outputs. In this model, an output of size B is prepaid for a duration of L_{max} blocks (L_{max} is to be set explicitly in a transaction creating the output). After L_{max} blocks after transaction inclusion into the blockchain, anyone (presumably, a miner generating first block after the expiration) is able to spend the output. Thus a spending script for any output

must be a combination of a regular user-defined spending script a condition on L_{max}:

$$(Height > out.height + L_{max}) \vee (regular\_script) \tag{2}$$

where *Height* is current blockchain height and *regular_ script* is user-defined spending script for the output. A mandatory minimal space-time fee to be paid by a transaction creating the output is following:

$$f_{s-prepaid} = K \cdot (\sum_i (B_i \cdot (L_{max\_i} - Height)) - \sum_j (B_j \cdot (L_{max\_j} - Height))) \tag{3}$$

2.2 Postpaid Outputs

In the second model, a user pays fee for consumed space-time when he spends an output. When an output is spent its lifetime is known so resulting transaction fee can be calculated as:

$$f_{s-postpaid} = \sum_i (K_i \cdot B_i \cdot (H_{spent\_i} - H_{creation\_i})) \tag{4}$$

where $H_{creation\_i}$ and H_{spent_i} are heights where i-th output was created and spent respectively. Note that each output has its own space-time price K_i which is the K value at the time of the output creation. We note that for output value V a height H_{max} exists when the whole output value will be consumed by the space-time fee:

$$H_{max} = H_{creation} + \frac{V}{K \cdot B} \tag{5}$$

After that anyone should be able to spend this output, and a spending script looks like

$$(Height > out.height + H_{max}) \vee (regular\_script) \tag{6}$$

2.3 Scheduled Payments

The last model we introduce is a combination of the prepaid and postpaid models. Like in the postpaid model, a user pays for an output when he spends it in a transaction. In addition, the output is also enforced to be moved after a known duration D_s since being touched last time. That is, if the user do not move the coins before this, then anyone (presumably, a miner) can create a transaction claiming this output, returning all the coins back to the owner except of a space-time fee $K \cdot B \cdot D_s$. A spending script for the output would be like:

$$
\begin{aligned}
&(regular\_script) \vee \\
&(Height > (out.height + D_s) \wedge (out.value \leq K \cdot B \cdot D_s \vee \\
&\quad tx.has\_output(value = out.value - K \cdot B \cdot D_s, script = out.script)))
\end{aligned} \tag{7}
$$

In this model D_s is another protocol parameter which may be fixed by a protocol design or changed via miners voting like K.

3 Analysis of the Model

In this section we provide analysis of the model. We show how the proposal is meeting its main goal which is the state size control. We get estimations for additional miners rewards as well as lost coins recirculation. Throughout this section we use following values got from the Bitcoin network at the moment of writing the paper (February, 2017):

- number of unspent transaction outputs: $N_{utxo} \approx 45,000,000$
- number of coins in circulation: $N_{coins} \approx 16,000,000$ BTC
- mean value of an output: $V_{mean} \approx 0.35$ BTC
- simple payment transaction fee: $f_p \approx 0.0002$ BTC
- mean output size: $B_{mean} \approx 36$ Bytes
- time interval between blocks: $T_{block} \approx 10$ minutes
- transaction fees per block: $M_{pg} \approx 1$ BTC
- constant block reward: $M_c = 12.5$ BTC
- average number of outputs spent in a block $N_{moved} \approx 5,500$

We use the numbers from the Bitcoin to make calculations for some typical cases as this is the most used cryptocurrency with the longest history known. However, a cryptocurrency with our proposal being deployed would have different values for its metrics. Thus the results from this section should be considered as initial estimations.

We study money flows in a space-scarce economy while dividing them into two types. Miners claim lost coins and also charge every output in the system state every block. Users are incentivized to move coins more frequently thus increasing mining rewards due to increased economic activity.

3.1 Lost coins recirculation

The need of lost coins recirculation has been discussed in the literature [18,19] in regards with combat deflation that will eventually occur in cryptocurrencies with fixed supply. Possible cryptocurrency supply is known by design but actual supply is not known due to lost keys [20]. Deflation is a problem for traditional monetary systems, however, it is still an open question whether it will be problematic for the Bitcoin economy [21,22].

Our model provides lost coins recirculation mechanism by design. In this section we study how recirculation depends from charging approach chosen as well as parameters.

Fee for keeping output of size B in the state for L blocks is $K \cdot B \cdot L$. In *Postpaid* and *Scheduled payments* models space-time fee is taken from output value itself so when keys for an output which value is V_{output} are lost the coins will be taken back to circulation after L_{max} blocks:

$$L_{max} = \frac{V_{output}}{K \cdot B} \tag{8}$$

Maximum lifetime of a concrete output depends on the miner-controlled K parameter and user-defined output size and value. For a typical case in Bitcoin:

$$L_{max} \approx \frac{1}{100 \cdot K} \tag{9}$$

From this equation it is possible to calculate K by specifying target expected lifetime for an output of an average size. For example, in case of targeted 50 years period of recirculation for an average output, value of K parameter would be about 10^{-9}. We will use this estimated K value further in the analysis. Significant difference between *Postpaid* and *Scheduled payments* is that in *Postpaid* lost output will return to economy all at once after potentially big L_{max} period, while in the *Scheduled payments* mode they will return to economy by portions during this period. This makes *Scheduled payments* preferable to *Postpaid* in regards with mining rewards stability. Anyway coins recirculation rate for both the modes in a long-term would be calculated as

$$R_{postpaid} = R_{scheduled} = \frac{N_{lost} \cdot V_{output}}{L_{max}} = N_{lost} \cdot K \cdot B \tag{10}$$

where N_{lost} is the number of outputs with lost keys. Assuming current Bitcoins statistics, $N_{lost} \approx 10^6$ and $K = 10^{-9}$ we can estimate that ≈ 0.036 bitcoins are to be released from abandoned outputs in every block.

In the *Prepaid* mode a user defines L_{max} by himself when he creates an output and number of released coins after L_{max} is equals to output value V_{output} rather than much smaller space-time fee f_s. Note that L_{max} do not directly depends on K and coins recirculation rate will only depend on V_{output}, user-defined L_{max} and number of lost outputs N_{lost}:

$$R_{prepaid} = \frac{N_{lost} \cdot V_{output}}{L_{max}} \tag{11}$$

Assuming that $K = 10^{-9}$ we can estimate difference between prepaid and postpaid models:

$$\frac{R_{prepaid}}{R_{postpaid}} = \frac{V_{output}}{K \cdot B \cdot L_{max}} \approx \frac{10^7}{L_{max}} \tag{12}$$

For example, if user-defined coin recirculation period $L_{max} = 10$ years, then $\frac{R_{prepaid}}{R_{postpaid}} \approx 19$

Concluding, our fee model by its design provides a way to return lost coins to circulation thus preventing potential deflation. *Postpaid* model has a disadvantage that lost coins could be returned to circulation in far future. In *Prepaid* model coin recirculation rate is presumably much higher than in other models and *Scheduled payments* mode is the most smooth way to recycle coins.

3.2 User-Driven Money flow

From users point of view, the longer you keep some output, the bigger space-time fee you pay. Thus the proposed fee regime provides an incentive to move coins more frequently stimulating economic activity. In all the fee models introduced in Section 2, a user pays $K \cdot B$ coins every block for keeping his output in the state.

Currently in Bitcoin N_{moved} outputs are moved in a block, and this number can be defined as:

$$N_{moved} = \frac{N_{utxo}}{L_{mu}} \tag{13}$$

where L_{mu} is mean lifetime of an output. We can then define L_{mu} as:

$$L_{mu} = \frac{N_{utxo}}{N_{moved}} \tag{14}$$

In Bitcoin, $L_{mu} \approx 8182$ blocks.

We expect users moving coins more frequently thus mean lifetime becomes $L_{mi} < L_{mu}$. L_{mi} depends on K and also on blockchain-specific usage scenarios. Increasement in money flow could be calcualted as:
$\Delta N_{moved} = \frac{N_{utxo}}{L_{mi}} - \frac{N_{utxo}}{L_{mu}} = N_{utxo} \cdot \frac{L_{mu} - L_{mi}}{L_{mu} \cdot L_{mi}}$

3.3 Miner rewards

Incentivizing mining is very important for viability of a cryptocurrency economy. In addition to usual propagation fee, the new fee regime provides two new sources of miner's income: a space-time fee and lost coins in the *Prepaid* model.

Thus in the *Prepaid* model:

$$M_3 = M_{propagation} + M_{space-time} + M_{lost} \tag{15}$$

In other models:

$$M_2 = M_{propagation} + M_{space-time} \tag{16}$$

We now calculate the difference with reward M_{pg} (propagation fees) miners have in Bitcoin. We assume that number of inputs per transaction is the same, then number of transactions as well propagation fees are to be increased in the same way as N_{moved}:

$$M_{propagation} = M_{pg} \cdot (1 + \frac{L_{mu} - L_{mi}}{L_{mu} \cdot L_{mi}}) \tag{17}$$

A space-time reward depends on state size and space-time price K only:

$$M_{space-time} = \sum_{i=1}^{N_{utxo}} K_i \cdot B_i \approx K \cdot N_{utxo} \cdot B \tag{18}$$

Recirculation reward in the *Prepaid* model:

$$M_{lost} = \frac{N_{lost} \cdot V_{output}}{L_{max}} \tag{19}$$

Thus the additional mining reward is:

$$\Delta M_2 = M_2 - M_{pg} = M_{pg} \cdot \frac{L_{mu} - L_{mi}}{L_{mu} \cdot L_{mi}} + K \cdot N_{utxo} \cdot B \tag{20}$$

$$\Delta M_3 = M_{pg} \cdot \frac{L_{mu} - L_{mi}}{L_{mu} \cdot L_{mi}} + K \cdot N_{utxo} \cdot B + \frac{N_{lost} \cdot V_{output}}{L_{max}} \tag{21}$$

We remark that space-time reward $K \cdot N_{utxo} \cdot B$ does not depend on network throughput and can be easily predicted. thus it can be considered as a constant block reward making mining more stable and thus securing the network. The value of it may be controlled by adjusting K parameter.

3.4 State size

In this section we analyze the main goal of the proposed fee model which is to control size of the state. We study two most important cases. First, we estimate how big the state should be to give miners rewards comparable to Bitcoin's constant block reward of today (12.5 bitcoins per block at the moment of writing this paper). Second, we study what is the maximum size of state could be considering an attacker having all the coins in the system initially and performing constant spam attack.

Constant Rewards. For a state of size S miner is getting $S \cdot K$ space-time fees every block. For them to be a replacement for the current constant reward in Bitcoin M_c following equality should hold:

$$S \cdot K = M_c \tag{22}$$

which leads to 13 GB state for $K = 10^{-9}\ \frac{BTC}{Byte \cdot Block}$.

Spam Attack. We consider an attacker who creates an output roughly of size of a block S_{block} for every block in a row. Initially the attacker has all the N_{coins} of the system and he does not buy new coins during the attack (miners do not sell the coins to him). We also assume that K is constant during the attack (however, in real world miners would probably raise K during the attack). As soon as each block space-time fee should be paid for all previously created outputs, maximum possible number of such outputs O_{max} is to be calculated from equation:

$$N_{coins} = \sum_{i}^{O_{max}} K \cdot S_{block} \cdot i \approx K \cdot S_{block} \cdot O_{max}^2/2 \tag{23}$$

assuming that number of unspent outputs is big enough. Resulting state size:

$$S = S_{block} \cdot O_{max} = \sqrt{\frac{2 \cdot N_{coins} \cdot S_{block}}{K}} \tag{24}$$

which is about 183 gigabytes for Bitcoin network parameters and $K = 10^{-9}$ $\frac{BTC}{Byte \cdot Block}$. Note that it is upper-bound of state size calculated from non-realistic assumptions that all available supply will be put into the attack, and that miners do not raise K during the attack.

Summarizing, the new fee regime allows to control state growth with K parameter and allows to estimate preferable state size as well as an upper bound. Reasonable K value for a Bitcoin-like system leads to state of 13 GB in case of replacing current constant block rewards in Bitcoin with space-time fees, and of 183 GB for the spam attack backed by all the system tokens. Thus the model is preventing spam attacks like [13] and provides guarantee that miners can have state fit in RAM.

4 Related Work

Ethereum [9] introduced a fee to cover computational costs for processing a transaction (via rewarding a miner), with a mandatory minimum. A price of a computational unit (which is an instruction of Ethereum Virtual Machine) is controlled by miners via adjusting "gas" price. We consider to adopt this approach toward charging users for consuming space-time units.

FreiCoin [23] is a cryptocurrency with demurrage. Demurrage-adjusted running balance (aggregated for all the outputs associated with a key) is to be recalculated when a transaction is touching any output which is a part of the balance. Unlike FreiCoin, miners are watching over the state constantly to clean it in our proposal, and coins being lost are getting back into circulation (in FreiCoin, outputs forgotten do live forever in the unspent outputs set).

In PascalCoin [24], if an account has not made any operations for 420,480 blocks then its balance is considered as forgotten and miner can claim it. Our approach to recover lost coins is more flexible and natural.

5 Further Work

We left some questions for further research:

- it is interesting to see how tokens in a cryptocurrency with a proposed fee component can be compared with a Gesell's demurrage currency [25].
- an interesting research vector is how to find appropriate mandatory minimal f_v, f_p, f_s values. Another question is how these components can be combined in a compound minimal transaction fee f. Aside of a simple sum, one possible option is to set $f = max(f_p, f_v, f_s)$.
- for currency blockchains, a subsidized period of free storage could be applied. One possible option is discussed in Appendix **??**.
- a possible interaction with the problem of blockchain instability without the constant block reward [26] in the setting of increased fee supply is another concern to study.
- miners can choose different strategies on changing K value, based on estimated demand curve and storage pricing. Finding a set of optimal strategies is the another interesting topic to study.

References

1. S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system.
2. J. Garay, A. Kiayias, N. Leonardos, The bitcoin backbone protocol: Analysis and applications, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2015, pp. 281–310.
3. G. Andresen, Back-of-the-envelope calculations for marginal cost of transactions. URL `https://gist.github.com/gavinandresen/5044482`
4. P. R. Rizun, A transaction fee market exists without a block size limit.
5. bitcoin Wiki, Bitcoin script. URL `https://en.bitcoin.it/wiki/Script`
6. P. L. Seijas, S. Thompson, D. McAdams, Scripting smart contracts for distributed ledger technology, in: International Conference on Financial Cryptography and Data Security, 2017.
7. L. Goodmani, Michelson: the language of smart contracts in tezos. URL `https://tezos.com/pages/tech.html`
8. Solidity. URL `https://solidity.readthedocs.io`
9. G. Wood, Ethereum: A secure decentralised generalised transaction ledger, Ethereum Project Yellow Paper. URL `https://ethereum.github.io/yellowpaper/paper.pdf`
10. The ethereum network is currently undergoing a dos attack. URL `https://blog.ethereum.org/2016/09/22/ethereum-network-currently-undergoing-dos-attack/`
11. M. Vasek, M. Thornton, T. Moore, Empirical analysis of denial-of-service attacks in the bitcoin ecosystem, in: International Conference on Financial Cryptography and Data Security, Springer, 2014, pp. 57–71.
12. Blockchain.info, Number of unspent transaction outputs. URL `https://blockchain.info/charts/utxo-count?timespan=all`

13. Bitcoin_Wiki, July 2015 flood attack.
URL https://en.bitcoin.it/wiki/July_2015_flood_attack
14. L. Reyzin, D. Meshkov, A. Chepurnoy, S. Ivanov, Improving authenticated dynamic dictionaries, with applications to cryptocurrencies, in: International Conference on Financial Cryptography and Data Security, 2017.
15. Spv mining.
URL https://bitcoin.org/en/alert/2015-07-04-spv-mining
16. I. Eyal, A. E. Gencer, E. G. Sirer, R. Van Renesse, Bitcoin-ng: A scalable blockchain protocol, in: 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16), USENIX Association, 2016, pp. 45–59.
17. E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, B. Ford, Enhancing bitcoin security and performance with strong consistency via collective signing, in: 25th USENIX Security Symposium (USENIX Security 16), USENIX Association, 2016, pp. 279–296.
18. H. Gjermundrød, I. Dionysiou, Recirculating lost coins in cryptocurrency systems, in: International Conference on Business Information Systems, Springer, 2014, pp. 229–240.
19. H. Gjermundrød, K. Chalkias, I. Dionysiou, Going beyond the coinbase transaction fee: Alternative reward schemes for miners in blockchain systems, in: Proceedings of the 20th Pan-Hellenic Conference on Informatics, ACM, 2016, p. 35.
20. D. Ron, A. Shamir, Quantitative analysis of the full bitcoin transaction graph, in: International Conference on Financial Cryptography and Data Security, Springer, 2013, pp. 6–24.
21. Bitcoin_Wiki, Deflationary spiral.
URL https://en.bitcoin.it/wiki/Deflationary_spiral
22. S. Barber, X. Boyen, E. Shi, E. Uzun, Bitter to better-how to make bitcoin a better currency, in: International Conference on Financial Cryptography and Data Security, Springer, 2012, pp. 399–414.
23. M. Friedenbach, J. Timón, Freimarkets: extending bitcoin protocol with user-specified bearer instruments, peer-to-peer exchange, off-chain accounting, auctions, derivatives and transitive transactions (2013).
24. Pascal coin.
URL https://github.com/PascalCoin/PascalCoin/raw/master/PascalCoin%20White%20Paper%20-%20EN.pdf
25. S. Gesell, P. Pye, The natural economic order, Owen, 1958.
26. M. Carlsten, H. Kalodner, S. M. Weinberg, A. Narayanan, On the instability of bitcoin without the block reward, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2016, pp. 154–167.