On Space-Scarce Economy In Blockchain Systems

Dmitry Meshkov and Alexander Chepurnoy

IOHK Research

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 a blockchain data structure maintained by the whole network. Security of the scheme is relied on a mining process. If a majority of miners is honest then Bitcoin meets its security goals as formal analysis [2] shows. For the 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.

A user pay fees to miners to have his transactions included into the blockchain. Fees are useful for an existing cryptocurrency economy for two reasons:

1. Incentivizing miner's activity. 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 providing 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, i.e. miners network resources utilization.

2. Limit resources usage and spam. Besides of network utilization, transaction processing requires a miner to spend some computational resources. For most of cryptocurrencies, transactional language is limited [5], so a number of CPU cycles needed to process a transaction is 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 to transaction fees. This cost is specific to concrete transactional language and is out of scope of this paper.

A transaction in Bitcoin spends outputs from previous transactions and creates new ones. 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 an ordinary 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 an arbitrary transaction could be different (for example, in Ethereum [9] such a structure is called the *world state* and is a part of the protocol). For fast validation, 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. Similarly, in 2013 a denial-of-service attack against output storage residing in a secondary storage (HDD or SSD) was discovered in Bitcoin.

The main purpose of this paper is to consider a new mandatory component for a transaction fee scheme reflecting state growth. In all known cryptocurrencies of today, a state element once created lives possibly forever without paying anything for that. This leads to continuously increasing state (we point to the Bitcoin UTXO set size as an example [10]). Moreover, state may grow fast during spam attack, for example, 15M additional outputs where introduced to Bitcoins UTXO during spam attacks in July 2015 [11] and most of these outputs are not spent yet. The paper [12] proposes a technical solution for non-mining nodes where only miners hold the full state (assuming they can invest money in sufficiently big RAM storages) while other full-nodes are checking proofs of state transformations generated by miners, and size of a proof (in average and also in 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 [13], 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 living forever). The solution is a new mandatory fee component. We state that users should pay fee for both the state increasement and time new bytes will live in massively replicated storage of blockchain nodes for. This model is usual for cloud storage services where users pay for gigabyte of data per month. Such an approach prevents putting any data to the storage forever. Also miners can control their storage requirements by changing a fee factor. Later in this paper we will refer to the 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 (by miners).

Summarizing, we study an economy where memory of a massively-replicated system node becomes the most scarce resource eventually. Thus we call such an economy a *space-scarce economy*. The model of it considered in this paper is called space-scarce economy, or *SSE*, model.

Alex notes : note that we're talking about minimal fees

1.1 Assumptions

– We assume that 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, share fees with miners of some number of blocks before or after current one [14,15]). One difference is that in Bitcoin a miner can include his transactions into a block for free.
– We assume that a state consists of unspent outputs. An output is not modifiable so could be created and spent only. An output is protected by a script which is defined as a logical formula which can refer to properties of a blockchain (for example, its current height), spending transaction and an outputs itself. Alex notes : example
– For simplicity, we assume that a block is of a finite size but all the transactions a miner has at a given moment of time could be packed into it, if otherwise is not stated explicitly.
– Time is measured via notion of *height* which is a number of blocks since an initial block (a genesis block) till a particular one.
– All anyone-can-spend outputs are collected by miners, presumably immediately as they appear.

1.2 Structure of the Paper

The paper is organized as follows. A design of our model is described in Section 2. The model is analyzed in Section 3. Alex notes : finish

2 The Model

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

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

Validation and propagation costs are non-negative and assumed to be independent from the space-time fee. We assume that space-time fee impact to total fee f may range from negligible to dominant. It is possible for f_s to be negative since a transaction may both increase or decrease the state. It depends on size of outputs to be marked as spent, 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 calculated as follows:

$$f_s = K \cdot (T_{filled} - T_{cleared}) \tag{2}$$

Alex notes : we need to say somehow here about prepaid and postpaid outputs defined below

where K is a space-time price. We assume that miners can change this price like they change "gas" cost in Ethereum (based on hardware prices and token exchange rate) and thus control cost of maintaining the blockchain.

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. Each output of a size B in a state is prepaid for a duration L_{max} set explicitly in a transaction creating the output. After L_{max} anyone (presumably a miner generating first block after that) is able to spend the overdue output. Thus a spending script for any output must be a combination of regular spending script and loan period L_{max}:

$$(Height > out.height + L_{max}) \vee (regular\_script) \tag{3}$$

where *Height* is current blockchain height and *regular_ script* is user-defined spending script for the output Dima notes : What is out.height?. This form of a script is a part of consensus and a mandatory space-time fee is to be paid by a transaction creating the output:

$$f_{s-prepaid} = K \cdot (\sum_i (B_i \cdot (L_{max\_i} - Height)) - \sum_j (B_j \cdot (L_{max\_j} - Height))) \tag{4}$$

Alex notes : Height is subjective here, write about that

2.2 Postpaid Outputs

In the second model a user should pay 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:

$$f_{s-postpaid} = \sum_i (K_i \cdot B_i \cdot (H_{spent\_i} - H_{creation\_i})) \tag{5}$$

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 taken by the space-time fee:

$$H_{max} = H_{creation} + \frac{V}{K \cdot B} \tag{6}$$

After that anyone should be able to spend this output, sending the whole value to transaction fee and a spending script looks like

$$(Height > out.height + H_{max}) \vee (regular\_script) \tag{7}$$

Dima notes : TODO: space-time fee may be enforced by a script)

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 before a known duration D_s since its creation. 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:

$$\begin{aligned}&(regular\_script)\vee\\&(Height > (out.height + D_s)\wedge\\&\quad has\_output(value = out.value - K \cdot B \cdot D_s, script = out.script))\end{aligned} \tag{8}$$

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 goal to control state size. We get estimations for additional miners rewards as well as lost coins recirculation. Throughout this section we use following values got from current state of the Bitcoin network:

- 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 longest history. However, a cryptocurrency with SSE being deployed would have different values for the metrics. Thus results from this section got with the Bitcoin statistical data should be considered as initial.

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 literature [16,17] in regards with combat deflation that will inevitably occur in cryptocurrencies with fixed supply. Coins declared supply is known by cryptocurrency design, but available supply is not known due to lost coins locked in outputs that will never be spent [18]. On the other hand, deflation is a real problem in the traditional monetary system, however, it is still an open question whether it will be problematic for the Bitcoin economy [19,20].

Our model provides lost coins recirculation mechanism by design while its properties vary from model implementation and parameters. In this section we study how lost coins recirculation depends from the both.

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:

$$L_{max} = \frac{V_{output}}{K \cdot B} \tag{9}$$

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:

$$L_{max} \approx \frac{1}{100 \cdot K} \tag{10}$$

From this equation it is possible to calcualte K by specifying target maximum output lifetime. For example, in case of 50 years period of full recirculation for an output value K parameter would be 10^{-9}. Significant difference between *Postpaid* and *Scheduled payments* is that in *Postpaid* all coins 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* because lost coins from big enough outputs in *Postpaid* will not return to the circulation in foreseeable future. Anyway coins recirculation rate for both the modes in a long-term would be calculated as

$$R_{postpaid} = R_{scheduled} = \frac{N_{lost} \cdot V_{output}}{L_{max}} = N_{lost} \cdot K \cdot B \tag{11}$$

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 bitcoin 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} rater than much smaller space-time fee f_s. Note here, 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}:

$$R_{prepaid} = \frac{N_{lost} \cdot V_{output}}{L_{max}} \tag{12}$$

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

$$\frac{R_{prepaid}}{R_{postpaid}} = \frac{V_{output}}{K \cdot B \cdot L_{max}} \approx \frac{10^7}{L_{max}} \tag{13}$$

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

Concluding, *SSE* model by 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 a 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, 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:

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

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

$$L_{mu} = \frac{N_{utxo}}{N_{moved}} \tag{15}$$

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

With *SSE* fee component enabled users move coins more frequently, thus mean lifetime becomes $L_{mi} < L_{mu}$. It is hard to estimate L_{mi} as it depends on K and also on blockchain-specific usage scenarios. Increasement in money flow could be calcualted as:

$\varDelta N_{moved} = \frac{N_{utxo}}{L_{mi}} - \frac{N_{utxo}}{L_{mu}} = N_{utxo} \cdot \frac{L_{mu} - L_{mi}}{L_{mu} \cdot L_{mi}}$

3.3 Miner rewards

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

Thus in the *Prepaid* model:

$$M_3 = M_{propagation} + M_{space-time} + M_{lost} \tag{16}$$

In other models:

$$M_2 = M_{propagation} + M_{space-time} \tag{17}$$

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}:

$$M_{propagation} = M_{pg} \cdot \left(1 + \frac{L_{mu} - L_{mi}}{L_{mu} \cdot L_{mi}}\right) \tag{18}$$

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

$$M_{space-time} = \sum_{i=1}^{N_{utxo}} K_i \cdot B_i \approx K \cdot N_{utxo} \cdot B \tag{19}$$

Recirculation reward in the *Prepaid* model:

$$M_{lost} = \frac{N_{lost} \cdot V_{output}}{L_{max}} \tag{20}$$

Thus the additional mining reward is:

$$\varDelta M_2 = M_2 - M_{pg} = M_{pg} \cdot \frac{L_{mu} - L_{mi}}{L_{mu} \cdot L_{mi}} + K \cdot N_{utxo} \cdot B \tag{21}$$

$$\Delta M_3 = M_{pg} \cdot \frac{L_{mu} - L_{mi}}{L_{mu} \cdot L_{mi}} + K \cdot N_{utxo} \cdot B + \frac{N_{lost} \cdot V_{output}}{L_{max}} \tag{22}$$

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

3.4 State size

In this section we analyze the main goal of *SSE* model which is to control state size. We study two most important cases. First, we estimate how big the state should be to give miners rewards comparable to Bitcoin's constant reward of today. Second, we study what is the maximum size of state could be considering a user having coins in the system and performing constant spam attack. Dima notes : last sentense is not clean

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:

$$S \cdot K = M_c \tag{23}$$

which leads to 13 Gb state for $K = 10^{-9}$ $(BTC/(Byte \cdot Block))$.

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 miner would likely raise K during the attack). As soon as each block space-time fee $K \cdot B$ should be paid for all previously created outputs, maximum possible number of such outputs O_{max} is to be calculated from equation:

$$N_{coins} = \sum_{i}^{O_{max}} K \cdot S_{block} \cdot i \approx K \cdot S_{block} \cdot O_{max}^2 / 2 \tag{24}$$

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

$$S = S_{block} \cdot O_{max} = \sqrt{\frac{2 \cdot N_{coins} \cdot S_{block}}{K}} \tag{25}$$

which is about 183 gigabytes for Bitcoin statistics and $K = 10^{-9}$ $(BTC/(Byte \cdot Block))$. Taking into account that *State* should be kept in memory, 183GB is quite big state size, however, miners can keep it in RAM. Note also that it is upper-bound of *State* size calculated from non-realistic assumptions that all available supply will be paid for space-time fee, and K is constant.

Thus *SSE* model allows to control state growth with K parameter and allows to estimate expected state size as well as an upper bound. Reasonable K value leads to 13 Gb of expected state size and 183 GB for a spam attack backed by all the tokens in the system. The model is preventing spam attacks like [11] and provide a guarantee that miners can have state fit in RAM.

4 Related Work

Ethereum[9] introduced a fee to cover computational costs of nodes 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[21] 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 [22], 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 natural. In the *prepaid* mode a user sets expiration time on his side (paying accordingly), in *postpaid* and *scheduled payments* modes a forgotten output is to be turned into space-time fees eventually.

5 Further Work

We left some questions for further research:

- it is interesting to see how tokens in a cryptocurrency with a *SSE* fee component can be compared with a Gesell's demurrage currency[23].
- an interesting research vector is how to find appropriate mandatory minimal f_v, f_p, f_s values. Another question is how these component could be combined in a transaction fee f. One possible option is to set $f = max(f_p, f_v, f_s)$.
- concrete dependency of space-time fee from output size and lifetime is another topic of discussion. One possible option is discussed in Appendix A.
- a possible interaction with the problem of blockchain instability without the constant block reward [24] in the setting of increased fee supply is another concern 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 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 151.
10. Blockchain.info, Number of unspent transaction outputs. URL `https://blockchain.info/charts/utxo-count?timespan=all`
11. Bitcoin_Wiki, July 2015 flood attack. URL `https://en.bitcoin.it/wiki/July_2015_flood_attack`
12. 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.
13. Spv mining. URL `https://bitcoin.org/en/alert/2015-07-04-spv-mining`
14. 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.
15. 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.
16. H. Gjermundrød, I. Dionysiou, Recirculating lost coins in cryptocurrency systems, in: International Conference on Business Information Systems, Springer, 2014, pp. 229–240.
17. 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.
18. 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.
19. Bitcoin_Wiki, Deflationary spiral. URL `https://en.bitcoin.it/wiki/Deflationary_spiral`
20. 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.
21. 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).

22. Pascal coin.
URL https://github.com/PascalCoin/PascalCoin/raw/master/PascalCoin%20White%20Paper%20-%20EN.pdf
23. S. Gesell, P. Pye, The natural economic order, Owen, 1958.
24. 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.

A On Subsidizing Space-time

It is usual to pay space-time fee for a cloud storage and thus *SSE* model is natural for a blockchain acting as a database. On the other hand, in a cryptocurrency setting it may be natural to allow users to keep their coins for free for some period of time S_s/B. For example $S_s \approx 1.8$ $(Mb \cdot Block)$ will allow to keep output for free for one year assuming 10 minutes block interval. While it is quite a big period for regular output, it is negligible for blockchain usage as a storage, e.g. will only allow to keep 12 Kb output for one day in *State*.

For sure, this free space time will increase *State* size upper bound. To maximize attack damage, the adversary will maximize number of created outputs, getting S_s space-time for each for free, filling all blocks during subsidized period S_s/B. In case of Bitcoin parameters one year of subsidized period for smallest outputs will make it possible to keep 51 Gb of outputs for free, increasing *State* size upper bound for 27%.

Thus, additional subsidized space-time parameter allows to use the same blockchain for both money and data applications. Also this additional parameter provide a new way to configure cryptocurrency, for example to increase K value painlessly for regular users, but increasing lost coin recirculation rate.

B Examples

Alex notes : moneychain and datachain

C Unified Transactions

Alex notes : We can throw away specially coinbase transactions getting unified transaction format, by requiring fees to be paid in form of anyone-can-spend output, and by creating constant rewards in predefined and hard-coded pre-genesis state, where a reward for a certain height is an output in the pre-genesis UTXO spendable only at this height.

D Questions and Answers

Q: *Aren't there important use-cases for "storing data forever"? Like when the ledger is used for storing property deeds or university degrees or other public,*

"official" documents?
A: While this is very appealing for a user to store something forever paying just once (or not paying anything at all), a resulting economy where miners need to store increasingly growing amount of data getting bounded rewards may be unstable or just not viable. As a solution, we propose a system where a state size could be bounded, then, as new data is to be introduced each block, expiration is strictly needed. However, for some real-life scenarios we can imagine hybrid systems, where some special actors (governments or universities) can store something forever (paying once, or not paying at all), while other participants are paying as we are proposing thus subsidizing public good.

Q: *If the miner can fully control K, how do you prevent a miner from setting K VERY high? Should not there be some sort of guarantee about the minimum value of L_{max} BEFORE a transaction is made?*
A: We assume miners will behave rationally, so they do not want to kill the business by setting K high. We propose to have a rule for changing K slowly, as block gas limit in Ethereum (say, a miner can increase or decrease K by 0.1% per block) so L_max could be estimated before the transaction being made with a good precision.