Designing Proof of Transaction Puzzles for Cryptocurrency

Taotao Li[1,2], Parhat Abla[1,2]and Mingsheng Wang[1], Qianwen Wei[1,2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
`litaotao,parhat,wangmingsheng,weiqianwen@iie.ac.cn`
[2] University of Chinese Academy Sciences, Beijing 100049, China

Abstract. One of the Bitcoin's innovations is the Proof of Work puzzle (aka scratch-off puzzle) as a consensus protocol for anonymous networks without pre-established PKI. Bitcoins based on the Proof of Work puzzle have been harshly blamed today for problems such as energy wasted and not easily scalable. In this paper, we construct a novel Proof of Transaction(PoT) puzzle, and prove that PoT puzzle satisfies the basic construction conditions of scratch-off puzzle. We also show construction of PoTcoin as application. PoTcoin has many advantage but not limited as strengthening the network topology, promoting currency circulation, anti-outsourcing computing and environment-friendly.

Keywords: consensus, proof of transaction, sequential aggregate signature, blockchain

1 Introduction

Since 2008, the Bitcoin[33] becomes the most popular cryptocurrency in the world. The most attractive part is a decentralized, distributed consensus mechanism (aka. Nakamoto consensus), that enables all participants in a peer-to-peer(P2P) network to reach consensus on a distributed public ledger (blockchain). In other words, the consensus is that thousands of independent nodes follow the simple rules that follows a spontaneously,asynchronous interactions.All Bitcoin's attributes, including currency, transactions, and security models that do not rely on central agencies and trust, are all derivatives of this mechanism.

Nakamoto consensus mechanism is as follows: In the Bitcoin network, a node (aka. miner) initiates a transaction and broadcasts it to the network. Other nodes receive the transaction and validate the transaction according to the verification algorithm. The transaction that satisfies the verification standard is temporarily saved to the node's transaction pool and broadcasts to other nodes again. Miners compete with others to solve a puzzle for obtaining the opportunity of adding confirmed transactions to the Bitcoin's public ledger of past transactions. Then, the solution will be broadcasted to other miners in the P2P network. If the solution is correct, miners will add the new block to the corresponding blockchain and continue to mine the new block. At the same time, the lucky miner(who firstly find the solution to the puzzle)will get corresponding reward and transaction fee. The mining process is a concurrently process, it may be the case that conflicting versions of blockchain, called fork. Due to the Bitcoin consensus mechanism [33], miners solve the problem of fork by mining on the longest chain. Instead, blocks on the shorter chain become the orphaned blocks, and all transactions in the orphaned blocks are returned to the transaction pool for recertification.

In the consensus mechanism, the most important part is the Scratch-off Puzzle(SoP). The SoP used by Bitcoin is based on moderately hard computational puzzle[3, 15], known as Proof of Work(PoW) puzzle. Essence of PoW puzzle is to solve the inequation, miners keeps doing hash computation to find a random number which satisfies the special inequation. Now, there are several challenges for PoW puzzle. Firstly, unequal mining probability. Some oligarch use their own mining resources or mining strategy, making their possession of scarcity of resources (ie, physical resources) account for most of the entire network of resources. Eventually the probability of common miners successfully mining block is getting smaller and smaller. Indeed, over the past few years, the computation power of large mine pool have exceeded the entire network by one-third over several times [39]. For example, GHash.io[1] computation power has more than half the amount of power in the entire network. Now, large mine pools (F2Pool, AntPool, BTCC and BW) are all located in China. Their total computation power reaches 60% of the whole network. Secondly, security is severely challenged. Bitcoin's security relies on the assumption that honest miner control most of the power in the network. In other words, if the attacker controls more than 50% of the power in the whole network, the network is not secure (for example, there exist double spending attack). Then, the above assumption has been seriously questioned. Eyal and Sirer[18] state that when a selfish mine pool has more than 25% of the whole network, and under the influence of selfish mining strategies[26, 37]and economies of scale[32], it attracts more miners to join the pool. Making the computation power of the mine pool has been more than half the power of the entire network or even more. Eventually, the Bitcoin system is no longer safe to decentralized system. Third, energy waste. In Bitcoin, in order to mine new blocks, miners run PoW algorithms, keep doing hash computation, thus causing a lot of unnecessary computational expense.

As mentioned above, a good SoP is crucial to the consensus mechanism and the extensibility of the entire blockchain. In this paper, we propose a new SoP - Proof of Transaction(PoT) puzzle. The idea of constructing the PoT puzzle is as follows: (1) Consideration of the transaction data in the blockchain. Obviously, a lot of data in the blockchain is the transaction. In addition to the currency transfer function, these transaction data has other attributes such as unforgeability, authenticity, traceability, etc. How can we use these data function and property once again perfect use? (2) Who will generate the next block? Needless to say, the next block is generated by Bitcoin miner in Bitcoin. In the PPcoin[24], the miner with large amount of coins generates the next block. In the Ethereum [9], the next block is generated by miners who have large computation power.

The computation power then easily be increased by external resources (such as state power), seriously affecting the security of the blockchain and increasing the instability of the blockchain. Based on the above two ideas, we used sequential aggregate signature scheme in the PoT puzzle. We use the transaction data in the blockchain so that those users who have initiated those transactions generate the next block. In other words, we give the mining right to those who really use the blockchain, so that these users really use the blockchain system as a "master". Unlike other cryptocurrencies, as computational power and coins increase, nodes will dominate the block generation. On the contrary, in the system we constructed, the number of transactions is not easy to increase in a short period of time. For the details of the sequential aggregate signature, see section 3.

We construct proof of transaction puzzle using sequential aggregate signature, and we show that proof of transaction puzzle satisfies the basic conditions for constructing Scratch-off puzzle[31]. Based on proof of transaction puzzle, we designed PoTcoin that have good performance, such as strengthening the network topology and facilitating the circulate of PoTcoin.

The contribution of this article is as follows:

- We successfully constructed a new Proof of Transcation puzzle using sequential aggregate signature and proved that the basic conditions for constructing Scratch-off puzzle are satisfied.
- Application for proof of transaction puzzle-PoTcoin. PoTcoin has the good performance of strengthening the network topology, promoting the circulate of PoTcoin, resistance to outsourcing computation and environment-friendly.

1.1 Related Work

As mentioned above, the Bitcoin PoW puzzle have been severely challenged in performance. In order to solve these problems, a large number of researchers have proposed many new Scratch-off puzzles. These puzzles can be classified in three main classes.

Scratch-off puzzle based on physical resources. Which is divided into three categories, the first category of puzzles are based on the computation of the hard puzzle, mining depends on the CPU computability. Bitcoin is using such computability to continuously do SHA-256 computations. Miller et al.[30] found that it takes approximately 2^{55} SHA-256 computations to mine a new block in the Bitcoin(which is equivalent to the amount of work required to crack a DES password), resulting in a waste of computation and natural resources. And solving this puzzle has no real value to society. Therefore, Scratch-off puzzles are considered based on useful computational puzzle (eg, protein folding problems[40]). In 2013, King [23] proposed Primecoin, using the huge computational power of the whole network to find prime numbers, the disadvantage is that the complete proof of security is ungiven. In 2014, Miller [30] proposed a Scratch-off puzzle based on Proofs-of-Retrievability (PoR) [22]. The main feature of this mechanism is the use of Bitcoin mining resources for distributed storage of archives, reducing the overall waste of Bitcoin. The main drawback is that it takes longer time to verify the results of the puzzle. The second category of scratch-off puzzle is based on storage puzzle. The capability of mining new block is dependent on miners storagability. Dziembowski [17] proposed the Proof-of-Space (PoS) and later the PoS was improved by [35, 36, 16]. The third category of scratch-off puzzle is based on the CAPTCHA problem[2]. This category of scratch-off puzzle can make miners more fair during mining a new block. Blocki's Proof-of-Human (PoH)[7] puzzle is based on CAPTCHA and indistinguishability obfuscation (IO)[19] and uses human-machine interaction to solve artificial intelligence problems (for example, reading distorted letters). The solution to the problem must rely on human participation, the probability of each miners successfully mining new blocks is uniform. However, the development achievements of IO [19] can not meet the demand of the PoH mechanism, therefore the PoH is unable to practical implement.

Scratch-off puzzle based on virtual resources. The main advantage of this type of Scratch-off puzzle is the reduction of consumption and transfer the physical resources needed for mining to virtual resources. For example, in the Nextcoin[13] based on the Proof-of-Stake (PoS) [25, 10], a user who owns the large amount of coins during the mining process, decide to produce the next block. At the same time, it also brought a significant flaw, centralization of the coin break decentralization of system. In other words, the more coins a miner has, the greater the probability of mining new blocks. Thus, a few number of miners who have most of the coins, more and more easy to mine new blocks and result in centralization of system.

Hybrid scratch-off puzzle based on physical resource and virtual resource. This type of scratch-off puzzle increases security and the cost of attacks. Duong[14] and Bentov[6] show that the advantage of this mechanism is that honest nodes still have the chance to use stakes to prevent the blockchain even if the malicious nodes have more than 50% computability. Typical cryptocurrencies are TwinsCoin[12].

Section 2 gives the basic knowledge and sequential aggregate signature are given in section 3, then we define the PoT puzzle and show its security. In section 5 we give application of the PoT puzzle, and its advantage and we conclude the whole article in last section.

2 Preliminaries

2.1 Assumption

PoT protocol is based on the Bitcoin protocol. In the PoT protocol, we assume that the resources (the number of transactions that have been initiated and recorded in the blockchain) owned by each PoT user are equal, and we also assume that the number of users in the PoT protocol is n, denoted as u_i ($i \in [1, n]$), where the number of online users is m($m \leq n$). Note that this is an "ideal assumption". In reality, each different user u_i has a different number of transactions. However, this ideal assumption is not loss of generality, because in reality user u_i is a combination of arbitrary users under ideal assumptions. We pointed out that in the protocol, the number of users who really participate in the operation of the protocol can not be determined. That is, we can not identify the number of users in this protocol that are participating in the operational protocol. In short, this is a static model under our assumption that the number of users is fixed while running the protocol.

2.2 Scratch-off puzzle

As mentioned in the introduction, the Bitcoin protocol is based on a computationally moderate puzzle that all miners compete with each other to solve it. However, it is common to call Bitcoin's puzzle as proof of work puzzle, and the basic requirements for building such a puzzle are somewhat different[11, 15, 21, 38]. Miller et al.[30, 31] shows some requirements that a Bitcoin puzzle(aka scratch-off puzzle) should satisfies.The following gives Miller[31] for the definition of scratch-off puzzle and a scratch-off puzzle must meet the three requirements.

In what follows, let λ denote a security parameter. A scratch-off puzzle is parameterized by parameters$(\underline{t}, \mu, d, t_0)$ where, informally speaking, $\underline{t}$ denotes the amount of work needed to attempt a single puzzle solution, μ refers to the maximum amount by which an adversary can speed up the process of finding solutions, d affects the average number of attempts to find a solution, and t_0 denotes the initialiazation overhead of the algorithm.

Definition 1. *A scratch-off puzzle is parameterized by parameters $(\underline{t}, \mu, d, t_0)$, and consists of the following algorithms (satisfying properties explained shortly):*

1) *$\mathcal{G}(1^\lambda)\rightarrow$ puz: generates a puzzle instance.*
2) *Work(puz,m,t) $\rightarrow$ ticket: The Work algorithm takes a puzzle instance puz, some payload m, and time parameter t. It makes t unit scratch attempts, using $t \cdot \underline{t} + t_0$ time steps in total. Here $\underline{t} = ploy(\lambda)$ is the unit scratch time, and t_0 can be thought of as the initialization and finalization cost of Work.*
3) *Verify(puz,m,ticket) $\rightarrow$ {0,1}: checks if a ticket is valid for a specific instance puz, and payload m. If ticket passes this check, we refer to it as a winning ticket for (puz,m).*

Intuitively, the honest Work algorithm makes t unit scratch attempts, and each attempt has probability 2^{-d} of finding a winning ticket, where d is called the puzzle's difficulty parameter. For simplicity, we will henceforth use the notation $\zeta(t, d) := 1 - (1 - 2^{-d})^t$ to refer to the probability of finding a winning ticket using t scratch attempts. For technical reasons that will become apparent later, we additionally define the shorthand $\zeta^+(t, d) = \zeta(t + 1, d)$.

A scratch-off puzzle must satisfy three requirements:

1) Correctness. For any (puz,m,t), if Work(puz,m,t) outputs ticket $\neq \perp$, then Verify(puz,m,ticket) = 1.

2) Feasibility and parallelizability. Solving a scratch-off puzzle is feasible, and can be parallelized. More formally, for any $\ell = poly(\lambda)$, for any $t_1, t_2, ..., t_\ell = ploy(\lambda)$, let $t := \sum_{i \in [\ell]} t_i$.

$$Pr \begin{bmatrix} puz \leftarrow \mathcal{G}(1^\lambda), \\ m \leftarrow \{0,1\}^\lambda, \\ \forall i \in [\ell] : ticket_i \leftarrow Work(puz, m, t_i), \\ \forall i \in [\ell] : Verify(puz; m; ticket_i) \end{bmatrix} \geq \zeta(t) - negl(\lambda)$$

Intuitively, each unit scratch attempt, taking time $\underline{t}$, has probability 2^{-d} of finding a winning ticket. Therefore, if ℓ potentially parallel processes each makes $t_1, t_2, ..., t_\ell$ attempts, the probability of finding one winning ticket overall is $\zeta(t) \pm negl(\lambda)$ where $t = \sum_{i \in [\ell]} t_i$.

3) μ-Incompressibility. Roughly speaking, the work for solving a puzzle must be incompressible in the sense that even the best adversary can speed up the finding of a puzzle solution by at most a factor of μ.More formally, a scratchoff puzzle is μ-incompressible (where $\mu \geq 1$) if for any probabilistic poly-nomial-time adversary $\mathcal{A}$ taking at most $t \cdot \underline{t}$ steps,

$$Pr \begin{bmatrix} puz \leftarrow (1^\lambda), \\ (m, ticket) \leftarrow \mathcal{A}(puz) : \\ Verify(puz, m, ticket) = 1 \end{bmatrix} \leq \zeta^+(\mu t) \pm negl(\lambda)$$

Note that $\zeta^+(t) = 1 - (1 - 2^{-d})^{t+1}$ is roughly the probability of outputting a winning ticket after t unit scratch attempts, though we additionally allow the adversary to make a final guess at the end (as in [38]), and hence the t+1 in the exponent instead of just t. Ideally, we would like the compressibility factor μ to be as close to 1 as possible. When μ =1, the honest Work algorithm is the optimal way to solve a puzzle.

3 Sequential Aggregate Signature Scheme

Aggregate signature[8](based on pairing) is a generalization of multi-signature in which several users sign on distinct messages. In aggregate signature, those signatures are generated by individuals and aggregate them. Note that the aggregating party may malicious. A sequential aggregate signature(SAS)[28] is very same as aggregate signature but the every signer will sign the message by some order. In a SAS scheme the signer may take the secret key and a message to be signed plus a SAS signature so far as input and output a SAS signature. Note that every signer will sign the message and aggregate then too. After all the SAS signature should be valid signature corresponding to all the signers public keys.

In this paper our purpose is to construct a secure proof of puzzle, So we didn't go any further. The following definition and the security experiment are very similar to [27] and the definition is as follows.

Definition 2. *An aggregate signature is consisted of three PPT algorithms (KeyGen,AggregateSign, AggregateVerify) such that:*

KeyGen(1^n) *input a security parameter 1^n and output public-secret key pair (pk,sk).*

AggregateSign($\sigma_k, ((m_1, pk_1), \cdots, (m_k, pk_k))$) *inputs an aggregate signature σ_k and k tuple of message-public key pairs $((m_1, pk_1), \cdots, (m_k, pk_k))$. Outputs an aggregate signature σ_{k+1} and k+1 tuple of message-public key pairs $((m_1, pk_1), \cdots, (m_{k+1}, pk_{k+1}))$*

AggregateVerify($\sigma_n, ((m_1, pk_1), \cdots, (m_n, pk_n))$) *Checks aggregate signature against the all messages and returns a boolean bit b. b=1 means σ_n match all the messages, b=0 implies σ_n is a invalid signature.*

A trivial sequential aggregate signature scheme can be constructed from ordinary signature scheme by putting all the signatures together. Namely, suppose $(Keygen, Sign, Verify)$ is a ordinary signature scheme , then we can obtain a sequential signature scheme by letting $AggregateSign(M_i, \mathrm{M}, sk_i, \sigma_i) = ((\mathbf{M}, m_i), (\sigma_i, \sigma))$, and $AggregateVerify$ on the fly, where $\sigma = Sign(m_i, sk_i)$.

the security of sequential aggregate signature schemes(SAS) is defined as the nonexistence of an adversary capable, within the restrict of a certain game, of existentially forging a sequential aggregate signature. Existential forgery here means that the adversary attempts to forge a sequential aggregate signature, on messages of his choice, by some set of users not all of whose private keys are known to the forger. We formalize this intuition as the sequential aggregate chosen-key security model. In this model, the adversary $\mathcal{A}$ is given a single public key. His goal is the existential forgery of a sequential aggregate signature. We give the adversary power to choose all public keys except the challenge public key. The adversary is also given access to a sequential aggregate signature oracle on the challenge key. His advantage, $Adv^{\mathcal{A}}_{AggSig}$, is defined to be his probability of success in the following experiment between a challenger and a PPT adversary $\mathcal{A}$:

Setup choose $(pk, sk) = Keygen(1^n)$, and give pk to $\mathcal{A}$ as a challenge.

Certification Query $\mathcal{A}$ provides key pairs (pk', sk') to certification center C for certifying his public key pk'. $C = (C, pk')$, if sk' is matching pk'.

Signature Query $\mathcal{A}$ can query a sequential aggregate signature under the challenge public key pk, on a message M of his own choice. furthermore, $\mathcal{A}$ provide an aggregate signature σ' so far on message vector $\mathbf{M}$ under public key $\mathbf{pk}$. Challenger checks that the validity of σ'; that $pk \notin \mathbf{pk}$; that $|\mathbf{pk}| < n$ (n is upper bound on the length of sequential signature); that $\mathbf{pk} \subset C$. If any of them fails the return $\perp$, otherwise respond with $\sigma = AggregateSign(sk, M, \sigma', \mathbf{M}, \mathbf{pk})$.

Output After polynomially many time querying the AggregateSign() algorithm, $\mathcal{A}$ outputs a forgery $(\sigma^*, \mathbf{M}, \mathbf{pk})$ and this forgery must be valid under AggregateVerify(); $pk \in \mathbf{pk}$ and $\mathbf{pk} \backslash \{pk\} \subset C$; $|\mathbf{pk}| \leq n$;

We will denote the advantage of adversary successes in the above game by $AggSignForge_{\mathcal{A}}^{SAS}$ and upper bound on the length of sequential aggregate signature by a positive integer n. ϵ and t positive reals , and q_C, q_S are polynomials in security parameter. The security of sequential aggregate signature scheme is given below:

Definition 3. *A sequential aggregate signature scheme is* $(t, q_C, q_S, n, \epsilon)-$*secure if there not exists a t-time adversary making* q_C *certification queries and making* q_S *queries to Signing algorithm and win the above game with advantage more than* ϵ*,that:*

$$Prob[AggSignForge_{\mathcal{A}}^{SAS}(n) = 1] \leq \epsilon.$$

The probability is taken over the randomness used in the experiment and adversary.

secure SAS schemes can be constructed permutations[28]. There is lattice based SAS scheme[4] which is secure in the random oracle model[5]. It can be constructed by any collection of preimage sampleable trapdoor functions like [20] or more efficient one[29]. When we say SAS scheme we mean by that a secure SAS scheme. We will use a simplified sequential aggregate signature scheme in our PoT puzzle. In the next section we will describe a new scratch-off puzzle using sequential aggregate signature scheme in detail and prove its security.

4 Proof of Transaction Puzzle

In this section, we define the syntax and security of the proof of transaction puzzle and use the sequential aggregate signature to illustrate the structure of the proof of transaction puzzle.

4.1 Definition

In the proof of work puzzle, all nodes in the entire network compete with each other to solve the puzzle in each a epoch. The node that first provides the correct answer indicates that it effectively solves the puzzle and obtains the block reward. The proof of work puzzle is composed of a set of algorithms: setup algorithm Setup(), puzzle instance generation algorithm G(), puzzle solution algorithm C() and verification algorithm V(). In the setup algorithm, it is mainly used to design public parameters. In the puzzle instance generation algorithm, it mainly uses the parameters and data of the setup algorithm to generate a puzzle instance. In the puzzle solution algorithm, the node keeps doing SHA-256 computation and tries to find the answer. It is worth noting that this is a non-deterministic algorithm. Because a user try an answer every time, the user do not know whether the answer will solve the puzzle. in the verification algorithm, the node verifies the answers to the puzzle received in the network. Note that this is a deterministic algorithm. Because each node that receives the puzzle and answers can verify the correctness of the answer through a verification computation. In order to reach a consensus, the proof of work puzzle must meet the basic conditions for constructing Scratch-off puzzle.

Our proof of transaction puzzle and proof of work puzzle are similar, but main differences are as follows: (1) Different mining resources. In the proof of work puzzle, the probability of miners mining new block is proportional to the computation power. The greater the computation power of miners, the greater the probability of miners mining new block. In proof of transaction puzzle, the probability of users mining a new block is proportional to the number of transactions they own. The greater transactions they have, the greater the probability of users mining new block; (2) The form of the puzzle is different. In the proof of work puzzle, miners solve an inequation problem. in the proof of transaction puzzle, users solve an equation problem; (3) The way to solve the puzzle are different. In the proof of work puzzle, miners constantly change the random numbers so that the hash value of the random number and the public parameters are less than the difficulty value. In proof of transaction puzzle , the user needs to find a chain of sequential signatures whose length is equal to the difficulty value. the syntax is as follow:

Definition 4 (Proof of Transaction Puzzle). *The proof of transaction puzzle consists of a set of algorithms (Setup, G, $u^{\mathcal{O}(\cdot)}$, V) as follows:*

Setup: *Setup is a system random setting algorithm that inputs the parameter 1^λ (λ is a security parameter) and outputs a system public parameter $PP \leftarrow Setup(1^\lambda)$, which includes a puzzle difficulty parameter is $\omega = ploy(\lambda)$.*

G: *G is a probabilistic puzzle generation algorithm, input the common parameter PP, and output the puzzle instance $puz = \sum_{i\in[1,\omega]} puz_i \leftarrow G(PP)$.*

$u^{\mathcal{O}(\cdot)}$**:** *$u^{\mathcal{O}(\cdot)}$ is a puzzle solution algorithm that outputs a answer $\sigma \leftarrow u^{\mathcal{O}(\cdot)}(PP, puz)$ of length ω where $\mathcal{O}(\cdot)$ is a signature oracle which inputs an online transaction [3], a special signature and message, outputs the signature of the owner of the transaction[4].*

V: *V is a deterministic puzzle verification algorithm that inputs public parameters PP and a pair of puzzle-answer (puz, σ) and outputs a bit $b := V(puz, \sigma, PP)$, which also includes signature verification. b = 1 means that σ is the valid signature of the puzzle puz, otherwise b = 0.*

We require Setup, G, $u^{\mathcal{O}(\cdot)}$ are a probabilistic polynomial time algorithm, V is a deterministic polynomial time algorithm.

Following notation of Miller et at.[31] we will let $\epsilon(k, \omega) = 1 - (1 - m^{-\omega})^k$, where m represents the number of online users in the network and m^{-1} represents probability of a user obtaining a valid signature after calling signature oracle once. In simple terms, $\epsilon(k, \omega)$ represents the probability of the user calling k times of signature oracle to get a valid answer to the puzzle.

[3] This transaction is in the longest block chain

[4] Note that this signature is for public parameters and the data which the user received from former user

Definition 5 (Honest User Solvability). *If a proof of transaction puzzle system (Setup, G, $u^{\mathcal{O}(\cdot)}$, V) is honest user solvable for each polynomial k = ploy (λ), and for any honest user $u^{\mathcal{O}(\cdot)}$ who controls k work unit, it holds that*

$$Prob\begin{bmatrix} PP \leftarrow Setup(1^\lambda); \\ puz \leftarrow G(PP); \\ \sigma \leftarrow u^{\mathcal{O}(\cdot)}(PP, puz); \\ V(PP, puz, \sigma) = 1; \end{bmatrix} \geq \epsilon(k, \omega) - negl(\lambda)$$

Definition 6 (Adversarial User Unsolvability). *If a proof of transaction puzzle system (Setup, G, $u^{\mathcal{O}(\cdot)}$, V) is adversarial user unsolvable for each polynomial k = ploy (λ), and for any adversary $\mathcal{A}$ who controls at most k work unit, it holds that*

$$Prob\begin{bmatrix} PP \leftarrow Setup(1^\lambda); \\ puz \leftarrow G(PP); \\ \sigma \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(PP, puz); \\ V(PP, puz, \sigma) = 1; \end{bmatrix} \leq \epsilon(k+1, \omega) + negl(\lambda)$$

Remark 1) in the proof of transaction system, the adversary can try to guess the signature value he needs without calling the signature oracle, just as in[38]. after calling the signature oracle k times, the adversary can try to guess the signature value. Therefore, the probability that $\epsilon(k+1, \omega) = 1 - (1 - m^{-\omega})^{k+1}$ is approximately equal to probability of calling k times the signature oracle and then obtain a valid answer. 2) Like Miller's incompressibility factor μ in[31], we have $\mu = 1$ here. This is because our proof of transaction puzzle system is optimal. As in Definition 3, honest users need at least k work units to solve the puzzle.

4.2 Structure

In this part, we will present the structure of proof of transaction puzzle on Bitcoin system. In Bitcoin system, the proof of work puzzle generates a puzzle instance $puz \leftarrow G(s)$ with the latest recent public parameter s. And miners try a random number x_i by constantly calling the random oracle(e.g.,the SHA256 hash function). A valid random number x_i makes the hash value of public parameter s and random number x_i less than the difficulty value. Namely, miners compute $y_i = RO(s, x_i)$. If $y_i < T_\omega$, the corresponding x_i is regarded as the answer to the proof of work puzzle. Given a random oracle $RO : \{0,1\}^\star \rightarrow \{0,1\}^n$, we will use the notation $T_\omega = 2^{n-\omega}$. Intutitively, this ensures that $RO(s, x_i) < T_\omega$ with probability is $2^{-\omega}$.

In our proof of transaction system, we first give a security parameter λ, where $\omega = ploy(\lambda)$ indicates the difficulty of the puzzle puz. The proof of transaction system generates a puzzle instance puz $\leftarrow G(s)$ with the latest public parameter s, where s represents the block header of the packed block constructed by the user. The packed block contains hash value of the block header of the previous block and transaction information. In order to solve the proof of transaction puzzle, the user needs to call the signing oracle $\mathcal{O}(\cdot)$, so that the length of a sequential aggregate signature chain is equal to the difficulty value. First, every user u_i computes $\sigma_i = Hash(s)$ with $Hash$ is a hash function, and $H_i = \sigma_i$ mod H with H is the height of the current blockchain, H_i is a specific block on the blockchain. The purpose is to randomly select a block from the blockchain; Then the user computes $Tx_i = \sigma_i mod \phi(H_i)(\phi$ is a function that computes the number of transactions contained in a block.) and broadcasts Tx_i to the network. Note that Tx_i is a specific transaction on the block H_i. The purpose of this step is to randomly selected a transaction from the block H_i. That is to say, the whole process is equivalent to each user randomly select a transaction Tx_i from the blockchain; Third, every user who received the transaction Tx_i verify that whether he is owner(who launched the transaction) of the transaction Tx_i. If the user u_{i+1}(also called a pair-key (sk_{i+1}, pk_{i+1})) is owner of the transaction Tx_i, the user u_{i+1} uses his private key sk_{i+1} which used to sign on the transaction Tx_i, to sign on σ_i and $M_i(M_i$ is i tuple of message-public key pairs $((m_1, pk_2), ..., (m_i, pk_{i+1})$ with m_1 is s and $m_i = \sigma_i||s)$, and get a aggregate signature $\sigma_{i+1} = u_{i+1}^{\mathcal{O}(\cdot)}(\sigma_i, M_i)$. Namely, the signature of the user u_{i+1} can be obtained by calling signature oracle $\mathcal{O}(\cdot)$; Further, the user computes a hash on σ_{i+1} to get σ'_{i+1}, and computes a block $H_{i+1} = \sigma'_{i+1}$ mod H; Then the user u_{i+1} computes $Tx_{i+1} = \sigma'_{i+1} mod \phi(H_{i+1})$ and broadcasts Tx_{i+1} to the network. Similarly, every user who received the transaction Tx_{i+1} verify that whether he is owner of the transaction Tx_{i+1}, if the user u_{i+2} (sk_{i+2}, pk_{i+2}) is owner of the transaction Tx_{i+2}, the user u_{i+2} uses his private key sk_{i+2} which used to sign on the transaction Tx_{i+1}, to sign on σ_{i+1} and M_{i+1}, and get a aggregate signature $\sigma_{i+2} = u_{i+2}^{\mathcal{O}(\cdot)}(\sigma_{i+1}, M_{i+1})$. And so on, a series of signature values $(\sigma_{i+3}, ..., \sigma_{k+1})$ will be obtained; Finally, users compute length τ of the series of signature values $(\sigma_{i+1}, ..., \sigma_{k+1})$. If $\tau = \omega$, it means that users successfully find the answer $Ticket := \{\sigma_{k+1}, [(m_1, pk_2), ..., (m_k, pk_{k+1})]\}$ to the proof of transaction puzzle.

Structure Our proof of transaction puzzle structure consists of the following three phases (Setup, Scratch-off, Verify).

Setup $s \leftarrow Setup(1^\lambda)$. λ is a security parameter, $\omega = ploy(\lambda)$ indicates the difficulty of puzzle puz. s is a parameter used to generate the puzzle $puz \leftarrow G(s)$, which indicates the block header of the packed block constructed by the user itself.

Scratch-off puzzle $puz := \sum_{i \in [1,\omega]} puz_i$ is solved during the Scratch-off phase. Every user can generate the parameter s, and compute hash value of s and export the next user. After that, the previous user needs to call the signature oracle $\mathcal{O}(\cdot)$ to export the next user.

```
1:  procedure PoT($s$)                                  ▷ s is block information
2:      $\sigma_1 \leftarrow Hash(s)$
3:      $H_1 = \sigma_1 \text{mod H}$
4:      $Tx_1 = \sigma_1 \bmod \phi(H_1)$
5:      $u_2 \leftarrow Tx_1$
6:      for $i = 1 \dots k$ do
7:          $\sigma_{i+1} \leftarrow u_{i+1}^{\mathcal{O}(\cdot)}(\sigma_i || M_i)$
8:          $\sigma'_{i+1} = Hash(\sigma_{i+1})$
9:          $H_{i+1} = \sigma'_{i+1} \text{mod H}$
10:         $Tx_{i+1} = \sigma'_{i+1} \text{mod} \phi(H_{i+1})$
11:         $u_{i+1} \leftarrow Tx_{i+1}$
12:     end for
13:     return $(\sigma_{k+1}, M_k)$          ▷ a sequential aggregate signature on $M_k$
14: end procedure
```

The answer Ticket is defined as follows:

$$Ticket := (\sigma_{k+1}, M_k)$$

Verify Compute $b := V(puz, Ticket, \omega)$. If b = 1, the Ticket is the correct answer to puz. That is, $\tau = \omega$. Otherwise, b = 0. The verification is necessary to repeat the Scratch-off process to verify that each Scratch-off is performed correctly.

To prove the security of the proof of transaction, we mimic Blocki's proof idea in[7]. If the Definition 3 is true in the sequential aggregate signature, we can easily verify that the proof of transaction puzzle is honest user solvable. Next, we give the security theorem of our proof of transaction puzzle.

Theorem 1. *If the sequential aggregate signature used in the PoT puzzle is secure and $Hash$ is random oracle, then PoT puzzle is adversarial user unsolvable.*

Proof. Suppose there exists an adversary $\mathcal{A}$ which controls k work-unit and the success probability in definition 6 is greater than $\epsilon(k+1) + negl(\lambda)$, then we can construct another adversary $\mathcal{A}'$ who can also (controls all secret keys except the challenged one) succeed in the SAS security game with noticeably.

Let q_H be the number of queries that the adversary $\mathcal{A}$ make to random oracle. Without lose of generality. we assume that the adversary $\mathcal{A}$ query an input on the random oracle only once.

we define an algorithm $\mathcal{D}$ that works as follows:

Algorithm $\mathcal{D}$

The algorithm input a $(pk, publicparam)$

1 Choose uniform $j \in \{1, ..., q_H\}$.
2 Run $\mathcal{A}$ in input $publicparam$,
3 When $\mathcal{A}$ makes ith random oracle query $Hash(\sigma_i)$, answer it as follows:
- If $i = j$, choose any feasible number r such that $pk = (r \bmod H) \bmod \phi(H_i)$, and return it to $\mathcal{A}$ as a response.
- If $i \neq j$, choose a random number r, and return r as a respond to the query.

4 When $\mathcal{A}$ makes SAS query on $(M_{i-1}, \mathbf{pk}, \sigma_{i-1})$ for pk_i, answers the query as follows:
- if $i = j$, query $(M_{i-1}, \mathbf{pk}, \sigma_{i-1})$ to SAS scheme and obtain σ_i.
- if $i \neq j$, produce a SAS signature σ_i by his own.
- return the SAS signature σ_i.

Now we define two games in the view of adversary $\mathcal{A}$.
Game0: This is the real game. In this game $\mathcal{A}$ will communicate with a challenger who works as same as in the Procedure $PoT(s)$. Namely, the challenger take a SAS signature σ_{i-1} so far as an input, then outputs a SAS signature σ_i corresponding to pk_i; taking random oracle query and responses with randomly. The adversary may query the random oracle and does two modulo computation to decide the next signer's public key. The adversary$\mathcal{A}$ will query the random oracle at most q_H times and query on signature oracle at most m times, then it outputs a valid SAS chain of length k+1.
Game1: In this game the adversary $\mathcal{A}$ will communicate with algorithm $\mathcal{D}$. All the random oracle and signature queries are responds by $\mathcal{D}$ as above. The adversary $\mathcal{A}$ may make q_H times random oracle query and m times SAS queries. After all it will outputs a SAS chain of length k+1.

Lemma 1. ***Game0*** *and* ***Game1*** *are computationally indistinguishable.*

Proof. Note that in **Game0**, the way to decide next signer in the SAS chain is random due to the randomness returned by random oracle and two modulo computation. In **Game1** , all the queries are similar to **Game0** but jth random oracle query. Because the challenge public key is generated randomly, the generation process for the next signer is the same. The SAS queries in these two games are completely indifferent. In the view of adversary $\mathcal{A}$, these two games are indistinguishable. □

Lemma 2. *If $\mathcal{A}$ wins in* ***Game1*** *with noticeably, then $\mathcal{A}'$ wins in experiment $AggSignforge^{SAS}_{\mathcal{A}'}$ with non negligibly.*

Proof. If the adversary $\mathcal{A}$ successfully generate a valid SAS chain of length k+1 by querying signature oracle at most m times, then it should forge one of his SAS chain. Without lose of generality, we assume that the adversary $\mathcal{A}$ only successfully generate a valid signature corresponding to a public key which is decided by querying on random oracle. We can compute the successful probability of $\mathcal{A}'$ as follows:

$$\begin{aligned} Prob[\mathcal{A}'] &= Prob[\mathcal{A}^{win} \wedge l = j] && (1)\\ &= Prob[\mathcal{A}^{win}] \quad Prob[l = i] && (2)\\ &= \frac{1}{q_H} Prob[\mathcal{A}^{win}] && (3) \end{aligned}$$

Note that l is index of public key that $\mathcal{A}$ attacks. Thus, if the adversary $\mathcal{A}$ succeed with noticeably, then the adversary $\mathcal{A}'$ will break the SAS scheme with non negligibly. □

The proof of theorem1 is straight based on the above two lemmas. □

5 Application—PoTcoin

In this section, we show how to use proof of transaction to construct a new cryptocurrency– PoTcoin. As mentioned earlier, our PoT protocol is very similar to the PoW protocol, except that PoT is used instead of PoW. In PoTcoin, we are not going to introduce PoT in detail, we mainly focus on the differences between them. In the following discussion, we use bitcoin (or PoTcoin) to represent coin units in the Bitcoin protocol (or PoT protocol).

5.1 Backgrand of Bitcoin

There are many interesting innovations and features in the Bitcoin protocol. However, in order to better introduce the PoTcoin, we have given the corresponding knowledge.

Blockchain. All transactions in Bitcoin are on blockchain. These transactions are stored with a special cryptographic structure, and we represent the blockchain as $C = b_0, ..., b_N$. C is valid if and only if $b_i(i \leq N)$ is valid. A single block $b_i = (Tx_i, Nonce_i, h_{i-1})$ is valid if and only if the following three conditions must be met: Firstly, all transactions Tx_i recorded in the block are valid. That is to say, each transaction is signed by the sender and all output amounts cannot exceed all input amounts; Secondly, the cryptographic hash value $h_{i-1} = SHA256(b_{i-1})$ must be the hash value of the previous block b_{i-1}; Third, the random number $Nonce_i$ contained in the block b_i satisfies $SHA256(b_i) < 2^{256-\omega}$, where ω represents the difficulty parameter ,which we will discuss in detail below. The first condition ensures that users cannot spend coin from other users. The second condition ensures that users can not forge a new blockchain $C' = b_0, ..., b_{i-1}, b'_i, b'_{i+1}$. The third condition ensures that it is moderated difficult to mine a new block on the blockchain.

Reward,epoch bitcoin is issued in a predetermined ratio in the Bitcoin protocol. At this writing, 12.5 bitcoins are distributed about every 10 minutes (a epoch). When a new time epoch begins, the node generates a puzzle puz by computing the latest block in the current blockchain. Then, nodes compete with each other to solve the puzzle puz for this epoch. The node that first submits a valid answer, will get the reward newly mined in the corresponding epoch.

5.2 PoTcoin

Similar to bitcoin, all PoTcoin transactions are recorded in the blockchain, denoted as $C = b_0, ..., b_N$, where each block $b_i = (Tx_i, Ticket_i, H_{i-1})$ contains three pieces of data. Namely, all transactions Tx_i in the block b_i, answer $Ticket_i$ of the proof of transaction puzzle and hash $H_{i-1} = SHA256(b_{i-1})$ of the previous block b_{i-1}. In the block b_i, all transactions Tx_i must be valid and the block b_i must contain the hash value H_{i-1}. In our PoT protocol, each user can find answers to PoT puzzle and verify the validity of the answer. In detail, given a PoT puzzle system (Setup, G, $u^{\mathcal{O}(\cdot)}$, V), each user can get the parameters s and ω by running the setup algorithm, a valid block b_i must contain a valid

$Ticket_i$. Let the verifier output 1 after running algorithm $V(puz, Ticket_i, \omega)$, $Ticket_i$ is the valid answer to the puzzle puz. Given an effective blockchain $C = b_0, ..., b_N$, the user can constructs a valid block by constructing a valid block $b_{N+1} = (Tx_{N+1}, Ticket_{N+1}, H_N)$. In order to find $Ticket_{N+1}$, users must stay online and sign certain data until a sequential aggregate signature chain of length ω is obtained. In the process of forming a sequential aggregate signature chain, if the next user who has been confirmed is not online or the signature generated by the next user is not available, users need to start again to form a new sequential aggregate signature chain. Otherwise, users find a valid sequential aggregate signature chain and successfully construct a valid block b_{N+1}.

The choice of parameters In Bitcoin, ω is a difficulty parameter. In general, it takes about 10 minutes for miners to generate a block [34]. That is, the user needs to compute 2^{ω} times for mining new block. In a new difficulty cycle, initially it takes 10 minutes or more to generate a block. However, as time goes on and changing of the computability, the time to generate a block is slowly less than or greater than 10 minutes ,until near the end, it may take less or more time to generate a block (such as 6 minutes or 14 minutes). We can clearly recognize that the dynamic difficulty is conducive to stability. If the difficulty is fixed, as more and more miners join the Bitcoin system, the time to generate a block will be reduced. Therefore, the difficulty value ω must be periodically adjustable. In Bitcoin, the difficulty value ω is adjusted every 2016 blocks for about two weeks and the difficult parameter $\omega = \omega_{old} - log(\frac{t_{elapsed}}{2016 \times 10min})$ [34].

In the PoT protocol, our difficulty parameter $\omega = ploy(\lambda)$ is very easy to adjust. We can adjust the difficulty parameter ω directly by adjusting the security parameter λ. When the time to generate a block is smaller or larger, we can adjust the security parameter λ so that the time of block generation is stable at a certain time.

Reward distribute In the PoT protocol, on the one hand, we rewarded users who mined block, on the other hand, we motivated users to stay online. The production of a new block requires a group of users to work together to complete. Each user is likely to be the first miner to form an effective chain of sequential aggregate signature, but it is not easy to become a second, third, etc. Therefore, we assign rewards and transaction fees to miners who participated to mine new block based on the order of miners in the sequential aggregate signature chain. For example, it takes k users to form a valid chain of sequential aggregate signature, and all rewards awarded to users are denoted as M. We divide k users into three equal groups of users. The first k/3 users of the sequential aggregate signature chain are share M/5 of the reward. The second k/3 users share 3M/10 of the reward; The last k/3 users share M/2 of the reward. Of course, this ratio is not necessarily fixed and may change as the actual operation of the PoT protocol.

6 Advantage of PoTcoin

PoTcoin has the following advantages over bitcoin and other cryptocurrencies:

1. Enhancing network topology, facilitating PoTcoin circulation. Firstly, users must stay online in order to get the reward; Secondly users should initiate more transactions to increase their share of total transactions; Finally, it will strengthen the network topology and facilitate the circulation of PoTcoin as the number of users and transactions in the network increases.

2. Environmentally friendly. We know that cryptocurrencies such as Bitcoin consume a large amount of useful computing resources such as energy or storage space during mining [35]. And in our PoTcoin mining process, the user stays mostly online and does some signature computations and verification, where resources are consumed as much as a normal computer consumes. So we think PoTcoin is environment-friendly.

3. Resistance to outsourcing computation. In Bitcoin, some "rational" miners outsource their mining resources to one or more large mining pool in order to expand their revenues and form Hosted mining, such as Alydian[17]. Hosted mining is very attractive, as it reduces the cost of miners mining due to economies of scale. In the PoT protocol, it is clear that if a user outsources his own mining resources (the ownership of the transaction – the private key) to several large mining pool, the user leaks his private key, and the large mining pool can take away the user's PoTcoin.

7 Conclusion

Currently, most cryptocurrencies are based on proof of work puzzle and proof of stake puzzle. However, these cryptocurrencies are faced with a very serious challenge. The Bitcoin based on the proof of work faces the problem of resource waste and security. The Peercoin based on the proof of stake faces centralization of the coin.

In this paper, inspired by the challenges faced by cryptocurrencies, we construct a novel proof of transcation puzzle for the first using sequential aggregation signature. We show that proof of transcation puzzle satisfies the basic conditions of constructing scratch-off puzzle. We also designed a new cryptocurrency – PoTcoin, based on the proof of transcation puzzle. Our PoTcoin has good performance, for example, strengthening the network topology, facilitating the circulation of PoTcoin, resistance to outsourcing and environment-friendly. We leave behind a public challenge that how many transactions the user owns when the users will dominate the generation of blocks in the network? Just as Eyal et al.[18] analyzed in Bitcoin, selfish miners will dominate the generation of blocks in the network through selfish mining strategy, when selfish miners own mining power more than 25% of the total network.

Bibliography

[1] http://arstechnica:com/security/2014/06/bitcoin/security/guarantee/shattered/by/anonymousminer/with/51/network/power/ , 2014.

[2] Luis Von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. Captcha: Using hard ai problems for security. *Lecture Notes in Computer Science*, 2656:294–311, 2003.

[3] Adam Back. Hashcash - a denial of service counter-measure. In *USENIX Technical Conference*, 2002.

[4] Rachid El Bansarkhani and Johannes A. Buchmann. Towards lattice based aggregate signatures. In *Progress in Cryptology - AFRICACRYPT 2014 - 7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings*, pages 336–355, 2014.

[5] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73, 1993.

[6] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof. 2014.

[7] Jeremiah Blocki and Hong Sheng Zhou. Designing proof of human-work puzzles for cryptocurrency and beyond. In *Proceedings, Part II, of the 14th International Conference on Theory of Cryptography - Volume 9986*, pages 517–546, 2016.

[8] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, pages 416–432, 2003.

[9] Vitalik Buterin. A next-generation smart contract and decentralized application platform. 2014.

[10] Vitalik Buterin. Proof of stake faq. https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ, 2016/ , 2016.

[11] Liqun Chen, Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi. Security notions and generic constructions for client puzzles. In *International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, pages 505–523, 2009.

[12] Alexander Chepurnoy, Tuyet Duong, Lei Fan, and Hong-Sheng Zhou. Twinscoin: A cryptocurrency via proof-of-work and proof-of-stake. In *In Cryptology ePrint Archive*, 2017.

[13] NXT Community. Nxt whitepaper. https://www.dropbox.com/s/cbuwrorf672c0yy/ NxtWhitepaper v122 rev4.pdf/ , 2016.

[14] Tuyet Duong, Lei Fan, and Hong-Sheng Zhou. 2-hop blockchain: Combining proof-of-work and proof-of-stake securely. In *In Cryptology ePrint Archive*, 2016.

[15] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *International Cryptology Conference on Advances in Cryptology*, pages 139–147, 1992.

[16] Stefan Dziembowski. Proofs of space and a greener bitcoin. 2013.

[17] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. 9216:585–605, 2015.

[18] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. 8437:436–454, 2013.

[19] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits (extended abstract). *Annual IEEE Symposium on Foundations of Computer Science*, 311(2):40–49, 2013.

[20] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 197–206, 2008.

[21] Bogdan Groza and Bogdan Warinschi. Cryptographic puzzles and dos resilience, revisited. 2014.

[22] Ari Juels and Burton S. Kaliski. Pors:proofs of retrievability for large files. In *ACM Conference on Computer and Communications Security*, pages 584–597, 2007.

[23] S. King. Primecoin: Cryptocurrency with prime number proof-of-work. 2013.

[24] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. 2012.

[25] Jae Kwon. Tendermint: Consensus without mining.

[26] Kevin Liao and Jonathan Katz. Incentivizing blockchain forks via whale transactions. 2016.

[27] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures, multisignatures, and verifiably encrypted signatures without random oracles. *J. Cryptology*, 26(2):340–373, 2013.

[28] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 74–90, 2004.

[29] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 700–718, 2012.

[30] A Miller, A Juels, E Shi, and B Parno. Permacoin: Repurposing bitcoin work for data preservation. In *IEEE Symposium on Security and Privacy*, pages 475–490, 2014.

[31] Andrew Miller, Ahmed Kosba, Jonathan Katz, and Elaine Shi. Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions. In *ACM Sigsac Conference on Computer and Communications Security*, pages 680–691, 2015.

[32] Frederick T Moore. Economies of scale: Some statistical evidence. *Quarterly Journal of Economics*, 73(2):232–245, 1959.

[33] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 2008.

[34] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016.

[35] Sunoo Park, Krzysztof Pietrzak, Joel Alwen, Georg Fuchsbauer, and Peter Gazi. Spacecoin : A cryptocurrency based on proofs of space. 2015.

[36] Ling Ren and Srinivas Devadas. Proof of space from stacked expanders. In *Theory of Cryptography Conference*, pages 262–285, 2016.

[37] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532, 2016.

[38] Douglas Stebila, Lakshmi Kuppusamy, Jothi Rangasamy, Colin Boyd, and Juan Gonzalez Nieto. Stronger difficulty notions for client puzzles and denial-of-service-resistant protocols. In *International Conference on Topics in Cryptology: Ct-Rsa*, pages 284–301, 2011.

[39] VitalikButerin. Bitcoin network shaken by blockchain fork. 2013.

[40] Peter G. Wolynes. Energy landscapes and solved protein-folding problems. *Philosophical Transactions*, 363(1827):453, 2005.