A Nonoutsourceable Puzzle under GHOST Rule

Gongxian Zeng, Siu Ming Yiu, Jun Zhang
Department of Computer Science
The University of Hong Kong, HK
Email: {gxzeng, smyiu, jzhang3}@cs.hku.hk

Hiroki Kuzuno
SECOM Co., Ltd., Japan
Email: khiro@khiro.org

Man Ho Au
Department of Computing
The Hong Kong Polytechnic University, HK
Email: csallen@comp.polyu.edu.hk

***Abstract*—Blockchain technology has attracted a lot of attention in recent years. Applications of blockchain are not only restricted to cybercurrencies, but have also been extended to other areas such as finance, e-health, music, and other business. One of the key components of blockchain is the design for miners who are responsible for adding new transactions (blocks) by solving a puzzle and receive some rewards in return. As a result, miners tend to join centralized mining pools to outsource their computing resources in order to gain more steady rewards, which may affect the security and fairness of the system. This motivates the researchers to propose nonoutsourceable puzzles. However, existing nonoutsourceable puzzles do not work well under the high-rate transaction processing protocol (GHOST). In this paper, we propose the first nonoutsourceable puzzle that can satisfy all security requirements of GHOST. Our experimental results show that our puzzle is practical.**

I. Introduction

Bitcoin and blockchain technology were proposed by Satoshi Nakamoto [1]. The design of blockchain technology attracts a lot of attention as it does not require a trusted third party nor a centralized authority while the integrity and security of the transactions can be guaranteed. The applications of blockchain have been extended from Bitcoin to other cybercurrencies (e.g. Litecoin [2], Zerocoin [3], Ethereum [4], Permacoin [5]) and also to other areas such as finance, e-health, music, and many business sectors using smart contracts [6], [7] and decentralized personal data management system [8]. In blockchain, using cybercurrency as an example, the list of transactions, i.e., the ledger, is maintained in the network in an open manner. Everybody in the system can get hold of this list. One of the key features in blockchain is the "miners" who are responsible for adding a new block that contains new transactions to the list by solving a puzzle and receive some rewards in return (e.g. bitcoins). Everybody can be a miner. In the original design of blockchain, solving the puzzle is a proof of work (i.e., the miner is required to spend a certain amount of computational resources in order to solve the puzzle successfully) and it is a probabilistic process. An implicit assumption is that miners will compete with one another, creating a fair environment so that no single miner can take control of the list. And a single correct blockchain will be maintained based on a simple majority rule as follows. Since miners work independently, an invalid (or a fake) transaction will not be agreed by the majority of users, thus only the *valid* blockchain that is agreed by the majority will be maintained. This assumption is valid if the miners are individuals, independent, and unrelated entities in the network. However, there is a trend that miners tend to join centralized mining pools, so that they can add up their computing resources to solve the puzzle together in order to gain more stable rewards. A centralized mining pool has a *pool operator*. This operator assigns mining work to other pool members and controls the whole pool. One can look at the issue from this perspective: the operator outsources puzzles to multiple parties to find the solutions and earn the reward while members in the pool get paid in return as well. If the pool has the dominant amount of computational resources, e.g. more than half of the computing power of the whole network, then the blockchain could be controlled by the pool operator since he can decide which transactions are included in the blocks and which version of the blockchains to support with high probability. This brings a lot of problems to the security of the blockchain design. For instance, in 2013, Vitalik Buterin [9] reported that there exist two blockchains in the network for six hours until some major mining pools, who supported the same chain, were shut down. The existence of mining pools is clearly a threat to the security of Bitcoin or the blockchain technology.

Decentralized mining pools. To tackle this problem, one existing direction is to derive schemes to have *decentralized* mining pools to remove the need for a pool operator. In these pools, members are usually treated equally and paid according to how much they have contributed to the solution of the puzzle. They tend to provide better incentives to miners to give up joining centralized mining pools, which are being controlled by pool operators. P2Pool [10] is the first decentralized mining pool. However, P2Pool could not resolve the efficiency and scalability problem, i.e., miners may need to invest more computing resources (e.g. local computation and bandwidth) when more and more contributions are submitted by peer miners. This makes it not attractive to the miners compared to centralized mining pools in which they may get a more stable reward with a more predictable investment. As a result, P2Pool is not popular and only account for a very small percentage of the mining power of Bitcoin. Recently, another decentralized mining pool, SMARTPOOL [11], that leverages the power of *smart contracts*, was proposed. It may be a better alternative that can compete with centralized mining pools. On the other hand, it is no guarantee that decentralized mining pools can stop miners joining centralized mining pools. Which pool a miner joins depends on quite a number of factors such as the amount of incentives/rewards a miner can obtain,

the investment (computing resources) the miner has to put in, and the variance of the rewards.

Nonoutsourceable puzzles. A more direct solution to tackle this "outsourcing problem" is to change the puzzle construction to make it *nonoutsourceable* (here, nonoutsourceability does not mean that the mining work cannot be outsourced, but if outsourced, member miners in the pool may be able to steal the rewards from the operator). The idea of nonoutsourceable puzzle was first used in Permacoin, another cryptocurrency, invented by Miller *el al.* [5]. The formal treatment, including the detailed constructions, was presented in [12]. Unlike decentralized mining pools, nonoutsourceable puzzles could stop pool operator outsource the mining task. This is a breakthrough in solving this problem. On the other hand, there are still some shortcomings in these puzzles. The proposed puzzles are based on Merkle tree, so the puzzle size is large. It can be hundred times larger than the original Bitcoin puzzle [1], resulting in less space for transactions if we want to bound the maximum block size to optimize the construction [13]. Another issue is the efficiency issue. It takes about 14 seconds to generate a proof for verification. Compared with the 10-minute block interval in Bitcoin, this is acceptable. But for other cryptocurrencies such as Litecoin [2] and Dogecoin [14], of which the block intervals are 2.5 minutes and 1 minutes respectively, the generation time may be a bit slow. If we consider Fastcoin [15] which creates a block every 12 seconds, the generation time is definitely not acceptable.

Along this direction, Ittay Eyal and Emin Gün Sirer [16] also presented a method called "Two-Phase Proof of Work (2P-PoW)" informally in a blog. In the original blockchain design in Bitcoin, miners only need to find a solution satisfying one inequality (or one phase). In 2P-PoW, the solution needs to satisfy two criteria (two phases). Their idea is as follows. Different from Miller's scheme, the private key related to the reward address will be integrated into the solution of a 2P-PoW based puzzle. If the puzzle is outsourced, the private key needs to be shared to pool members. Any member can then spend the reward. Moreover, their construction is related to the original Bitcoin puzzle, i.e., preserving the existing mining infrastructure and facilitate those miners who have invested on the hashing puzzle to stay in the system. Since it is no longer based on Merkle tree, the puzzle size is reduced substantially. However, it is still possible to outsource the first phase of the puzzle.

All these schemes focus on the puzzle. On the other hand, there have been new protocols and improvements proposed for blockchain. In the next section, we discuss these new development and show that the above schemes are not compatible with these new protocols.

Scalability and GHOST. Up to now, Bitcoin network can process about $270,000$ transactions per day, about 3 or 4 TPS (transactions per second). In contrast, it is reported that Visa's payment system can handle about 2000 TPS on average. Low scalability constitutes a serious problem for practical use, since it will cause high transaction fee. Solving this scalability has been an important topic for blockchain research, for examples, off-chain micropayment channel [17], [18], leader election and transaction serialization [19].

Another approach to solve the scalability problem is to increase the block creation rate. As part of the side effect, it brings in forks problem, that is new blocks to be created by different users at almost the same time and the blockchain would be divided into two chains. As a result, it would be easier to perform double spending attack, which refers to redirecting previously processed payments so that someone can use the same money twice. In 2015, [20] presented an alternative to the longest-chain rule which is called the Greedy Heaviest-Observed Sub-Tree (GHOST). GHOST has been adopted and a variant has been implemented as a part of Ethereum project. And the GHOST works well with another chain structure, a directed acyclic graph of blocks [21]. These two protocols can effectively tackle the fork problem when blocks are created at a higher rate so that more transactions can be processed. According to the new rule, the valid block in the chain is the one that has the most forks which means that most people support and work on this node. The observation behind the GHOST rule is that the blocks that are off the longest chain can still contribute to its weight to choose the long chain. Thus, **the reward for the accepted block would be shared by those off-chain blocks**.

However, as claimed in [12], when integrated into GHOST, the construction has the following problem: if a miner has a solution, he can derive multiple different solutions from it easily so that the node he supports could have more forks than others and gain unfair advantage. More details about this problem are given in Section III.

Contributions. Both nonoutsourceability and demand for higher transaction processing rate are equally important for blockchain development. In this paper, we propose a new nonoutsourceable puzzle that can be compatible with the GHOST protocol. Technically, we combine the ideas of Millier's Merkle Tree based scheme and 2P-PoW based scheme so that our puzzle posses the property of nonoutsourceability and has small puzzle size. We implemented our proposed puzzle and the experiments show that our construction is practical.

II. Preliminaries

A. Blockchain and GHOST

Blockchain, originally designed for Bitcoin, uses a public ledger to record all transactions. The ledger is a chain of sequenced blocks. Each block contains the transactions that have occurred since last block. To append a new block to the chain, users in the Bitcoin network need to solve a difficult puzzle (see the next subsection for details of the puzzle). The process of creating a new block is called mining and the users involved in this process are called miners. If all transactions included in this new block are valid, then other users would accept this block by appending new blocks after it. To motivate users in the Bitcoin network to become miners, the first miner that creates a new block would be rewarded (say Bitcoins). According to the mining speed, the difficulty of puzzle would be adjusted so that a block is mined about every 10 minutes

in Bitcoin network. If two new blocks are created by different miners at nearly the same time, then the blockchain would be divided into two chains and have forks. Some users would accept one of two chains and other users would accept the other one. It would not stop until one of chains is longer than the other one. In other words, the consensus rule in the original blockchain design is that the longest chain is considered to be the correct one [1].

To improve the ability of processing transactions, one method is to create blocks at a higher rate, say shortening the block interval to 1 minute. If that is the case, it is more likely to generate forks and many computational resources would be spent in the longest chain competition not in extending the chain. And it would be easier to perform double spending attack. Thus, the original longest-chain rule is not suitable when blocks are created at a high rate. In 2015, an alternative to the longest-chain rule, which is called the Greedy Heaviest-Observed Sub-Tree (GHOST), is presented in [20]. They regard a blockchain with forks as a tree. Under the GHOST rule, the node that has the most forks among those nodes with the same height would be regarded as a valid block in the chain because most people accept this block in order to extend it. Though only one block among all forks with the same height would be in the longest chain later, the other forks off this chain also contribute to choosing the longest chain. Thus these blocks would also be rewarded.

B. A scratch-off puzzle

Recall that to win the reward, a miner needs to solve a difficult puzzle. The following parameters are important for solving the puzzle.

- puz: the hash value of the last block.
- m: contains the miner's public key and a new set of transactions to be committed to the blockchain. To reduce the size of m, we can do some hash computations first. For example, we adopt Merkle hash tree and m is the value in the root.
- r: a nonce that satisfies $H(puz \parallel m \parallel r) \leq 2^{\lambda - d}$ where λ is the security parameter, d is the difficulty parameter and H is a secure hash function.
- $ticket$: the full solution. In Bitcoin, it includes r only.

The difficulty parameter is adjusted according to mining speed so that each epoch lasts for about 10 minutes. The mining process is something like a scratch-off lottery so that the Bitcoin-like puzzle is called a scratch-off puzzle and the formal definitions are given below (we follow the definitions from [12]):

Definition 1: A scratch-off puzzle is parametrized by parameters (t', μ, d, t_0), and consists of the following algorithms:

- $\mathcal{G}(\lambda) \to puz$: generate a puzzle instance.
- $Work(puz, m, t) \to ticket$: The $Work$ algorithm takes a puzzle instance puz, the payload m, and time parameter t as input. It makes t unit scratch attempts, using $t \cdot t' + t_0$ time steps in total. Here $t' = poly(\lambda)$ is the unit scratch time, and t_0 can be thought as the initialization and finalization cost of $Work$.
- $Verify(puz, m, ticket) \to \{0, 1\}$: The algorithm checks if the $ticket$ is a valid for a specific instance puz and the payload m. If the $ticket$ passed this check, we refer it as a "good" ticket for (puz, m).

Intuitively, the honest $Work$ algorithm makes t unit scratch attempts, and each attempt has probability 2^{-d} of finding a "good" ticket, where d is the puzzle's difficulty parameter. We let $\zeta(t, d) := 1 - (1 - 2^{-d})^t$ denote the probability of finding a "good" ticket after t scratch attempts.

A scratch-off puzzle must satisfy the following requirements:

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

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

$$Pr\left[\begin{array}{l} puz \leftarrow \mathcal{G}(\lambda), \\ m \leftarrow \{0,1\}^\lambda, \\ \forall i \in [l] : ticket_i \leftarrow Work(puz, m, t_i) : \\ \exists i \in [l] : Verify(puz, m, ticket_i) = 1 \end{array}\right] \geq \zeta(t, d) - negl(\lambda).$$

Incompressibility. Anyone can speed up the finding of a puzzle solution by at most a factor of $\mu (\mu \geq 1)$. Formally, a scratch-off puzzle is μ-incompressible if for any probabilistic polynomial-time adversary $\mathcal{A}$ taking at most t scratch attempts,

$$Pr\left[\begin{array}{l} puz \leftarrow \mathcal{G}(\lambda), \\ (m, ticket) \leftarrow \mathcal{A}(puz) : \\ Verify(puz, m, ticket_i) = 1 \end{array}\right] \leq \zeta^+(\mu t, d) \pm negl(\lambda).$$

Note that $\zeta^+(k, d)$ denotes $\zeta(k + 1, d)$ which is roughly the probability of outputting a "good" ticket after k failed unit scratch attempts because we allow the adversary to make a final guess. Roughly speaking, μ-incompressibility is the limitation of algorithm optimization to solve the puzzle.

It is trivial to prove that the puzzle used in Bitcoin is a scratch-off puzzle.

C. Mining pool

The more computing resources the miners hold , the faster they can find the solutions. Thus, people hire others to help them to solve the puzzle, forming a powerful mining pool. The hired persons are called workers/members and the organizer is called the pool operator. The outsourcing protocol for a scratch-off puzzle is defined as follows [12].

Definition 2: Let (t_e, t_w, t_o)-protocol denote a two-party outsourcing protocol between pool operators and workers for one scratch-off puzzle $(\mathcal{G}, Work, Verify)$, where t_e is the effective running time for solving the puzzle, t_w is the workers' most running time and t_o is the pool operator's most running time, such that:

- The pool operator's main input is puz, and the workers' input is $\perp$;
- $t_e < t_w + t_o$ and $t_o \ll t_e$;
- Operator outputs $ticket$ at the end, where $ticket$ is either a "good" ticket for the puzzle puz and the payload m or $ticket = \perp$.

Shortly speaking, if a scratch-off puzzle is outsourceable, then the most computing work is done effectively by the workers.

III. Security requirements under the GHOST rule

We first define nontransferability and nonoutsourceability under the GHOST rule. And then analyze whether those proposed nonoutsourceable puzzles are feasible in the GHOST rule.

A. Secure requirements under GHOST rule

GHOST offers a promising approach for increasing the rate of blocks and thereby reducing the reward variance for solo miners without compromising security. In fact, it creates a more complicated blockchain, or something like a blockgraph and allows for even the stale blocks to earn a reward. Since the GHOST changes the rules to choose the longest-chain, security issues would be changed in some ways. Though our main contribution is nonoutsourceability, nontransferability is a more basic property and also very important, so we will discuss it first.

A miner A finds a $ticket$, creates his block and sends it to the network. If A's block can help B to create B's block, then B may receive the reward once B has a better network connection and can send his block to others more quickly. Thus, the legitimate rights and interests for the first miner to create a new block can not be protected. In other words, a good puzzle cannot allow someone to transfer A's block to B's block easily. Generally, different miners have different payloads ms, so in most cases, nontransferability is that for different payloads m and m^*, knowing the $ticket$ for m does not help to find the solution for m^*. But when we adopt GHOST rule to choose the longest-chain, the condition is more restricted.

Consider the following cases:

- Supposing that miner A has found a solution for some m, if he can easily generate many other different solutions for the same m, then the block that the miner A supports would have more forks so that the miner would get rewards with higher probability.
- Supposing that miner A has found a solution for some m, if he can easily generate solutions for the other different m^*, then the miner would apply for many payment addresses so that the miner can have many different m^*. As a result, the block that the miner A supports would have more forks so that the miner would get rewards with higher probability.

The cases discussed above are all unfair means to achieve rewards and it can be expected that they would increase blockchain size rapidly. Thus, a fair and reasonable puzzle under the GHOST rule is that if someone finds a $ticket$ for the payload m, it is still difficult for him and other miners to find another solutions for m and also difficult to find $ticket^*$ for another payload m^*. Formal definition is as follows:

Definition 3: A scratch-off puzzle is nontransferable if it holds the following probability for any miner A who takes most t scratch attempts:

$$Pr\left[\begin{array}{l} puz \leftarrow \mathcal{G}(\lambda), \\ m \leftarrow \{0,1\}^{\lambda}, \\ ticket \leftarrow Work(puz, m, t), \\ Verify(puz, m, ticket) = 1, \\ \forall m^{'}: \\ ticket^{'} \leftarrow A(ticket, puz, m^{'}, t), \\ If\ m^{'} = m, ticket^{'} \neq ticket, \\ And\ Verify(puz, m^{'},\ ticket^{'}) = 1 \end{array}\right] \leq \zeta^{+}(t,d) - negl(\lambda).$$

Next, we are going to consider nonoutsourceability under GHOST rule. Nonoutsourceability does not mean that the outsourcing protocol (t_e, t_w, t_o)-protocol cannot be run between the pool operators and workers, but is based on the following observation:

Observation. If the pool operator wants to outsource his mining work to other workers, the operator would take the risk of losing money.

In other words, the mining work can help the miners create their own blocks or miners can spend the reward.

Definition 4: A scratch-off puzzle is nonoutsourceable if it holds the following property for every (t_e, t_w, t_o)-protocol in a mining pool:

1)For the pool operator $\mathcal{O}$, it exists an adversary $\mathcal{A}$ such that

$$Pr\left[\begin{array}{l} puz \leftarrow \mathcal{G}(\lambda), \\ m \leftarrow \mathcal{O}(\lambda), \\ ticket \leftarrow Work(puz, m, t), \\ Verify(puz, m, ticket) = 1, \\ m^* \leftarrow \mathcal{A}(\lambda): \\ ticket^* \leftarrow \mathcal{A}(ticket, puz, m^*, t), \\ Verify(puz, m^*, ticket^*) = 1 \end{array}\right] \geq \Phi(\lambda),$$

where $\Phi(\lambda) \in (0,1)$ is a constant, and when pool operator interacts with $\mathcal{A}$, from the view of pool operator, it is computationally indistinguishable from when the pool operator interacts with an honest worker.

2)Or for the pool operator $\mathcal{O}$, it exists an adversary $\mathcal{A}$ such that

$$Pr\begin{bmatrix} puz \leftarrow \mathcal{G}(\lambda), \\ m \leftarrow \mathcal{O}(\lambda), \\ ticket \leftarrow Work(puz, m, t), \\ Verify(puz, m, ticket) = 1 : \\ \mathcal{O} \xrightarrow{Tx} \mathcal{A} \end{bmatrix} \geq \Phi(\lambda),$$

where $\Phi(\lambda) \in (0,1)$ is a constant and Tx is a valid transaction that $\mathcal{O}$ transfers the reward to $\mathcal{A}$ and it is successfully created by $\mathcal{A}$.
In fact, in the transaction Tx, if the reward is transferred to $\mathcal{A}$, $\mathcal{A}$ may be detected, so for security, it can be any transaction spending the $\mathcal{O}$'s reward and the most important is that Tx is created by $\mathcal{A}$.

B. Analysis of current puzzles

MT-based scheme. Based on Merkle Tree, Miller *et al.* constructed a nonoutsourceable scratch-off puzzle and referred it as a weakly nonoutsourceable puzzle since the thieves can be caught. To fix this problem, they also proposed a generic transformation from any weakly nonoutsourceable scratch-off puzzle to a strongly nonoutsourceable puzzle so that the thieves can spend the money securely. However if we adopt the GHOST rule, these constructions are not feasible. Next, we are going to analyze why the two schemes are not suitable in GHOST.

(1)Weak scheme. It is complicated to analyze this weak scheme. We first introduce this weakly nonoutsourceable Merkle Tree based puzzle:

1)Preparation step: to solve a puzzle, a miner first constructs a Merkle Tree with random number in the leaves. The interval node is the hash result of its children nodes and the root is denoted by $digest$. The whole tree is kept in secret but $digest$ is public.

2)Scratch attempt: the miner samples a nonce r, computes $h = H(puz \parallel r \parallel digest)$ and uses h to select q leaves of the Merkle Tree (i.e., using h as a seed to a pseudorandom number generator) and their corresponding Merkle proofs, denoted by σ. If $H(puz \parallel r \parallel \sigma) \leq 2^{\lambda - d}$, then find the solution, or repeat the previous computations.

3)Sign: the miner computes $h^{'} = H(puz \parallel m \parallel digest)$ and uses $h^{'}$ to select a set of $4q^{'}$ leaves (i.e., using $h^{'}$ as a seed to a pseudorandom number generator). Finally, choose $q^{'}$ of them and their proofs (denoted by $\sigma^{'}$) as signature of m and return $ticket = \{digest, r, \sigma, \sigma^{'}\}$.

4)Verify: compute h and $h^{'}$ by ticket and other parameters, such as m. Then check whether σ and $\sigma^{'}$ contain leaves select by h and $h^{'}$. Finally, check whether r satisfy $H(puz \parallel r \parallel \sigma) \leq 2^{\lambda - d}$.

The reason why this puzzle is nonoutsourceable is that, in an outsourcing protocol, workers need query Merkle proofs with pool operator for every scratch attempt so that he can know a lot of nodes of this Merkle Tree, even he can sign his own block with high probability. However, once the miner finds the correct nonce r, then he can sign any payload m with his secret Merkle Tree. Thus, it does not satisfy nontransferability. But in this way, many public parameters in those block are the same, such as $digest$, r. Thus, one detection mechanism that can be proposed is to check these public parameters. If we find two blocks with the same public parameters, we say that they are invalid and created by malicious attackers so the block owners can not get any reward. Although transferability problem can be fixed in this way, outsourcing becomes possible. The pool operator would say that if someone steals the reward, then the pool operator would reveal the secret parameters and all workers create blocks with the same public parameters so that the thief could not get the reward either.

Finally, it is most likely that the whole Bitcoin-like network based on such puzzle would be in a stable state. Only one mining pool exists and others are solo miners. The reason that only one pool can exist is trivial. For simplicity, supposing that there are two ming pools: A and B, and supposing that A has more computing resources than B without loss of generality, they may choose different blocks to support so that one of them, most likely B, would fail to get reward. Then there are two different strategies for B:

- B can ask some workers to join the competing mining pool A to steal the money. According to the previous analysis, all the workers in the mining pool A would create the blocks with same public parameters once a theft happens. Thus, the pool A can not attain any rewards. At the same time, A would also ask some workers to join B too. Similar results would also happen to B. So no mining pools can exist in this way. On this condition, in order to achieve a stable reward, it is more likely that the two mining pools A and B would have a compromise and merge into one.
- asks one worker to join A to get the information about the block A would support so that B also supports that same block. Then B can also get some reward. In fact, in this way, the two mining pools A and B are equal to one mining pool.

Therefore, there exists an effective outsourcing protocol. Although the thefts can happen, thieves can not benefit from such actions.

(2)Strong scheme. As mentioned in the introduction, to prevent thieves from being caught, Miller's strong scheme generates a complicated and random proof instead of some parameters, such as the random Merkle tree, $digest$, r for verification. However, such an algorithm of generating proofs is not deterministic, so Miller's strong scheme can produce many different proofs for the same puzzle. In other words, if a malicious miner finds a correct solution, he can easily derive multiple solutions from the same underlying solution. According to the GHOST rule, the block which has the most forks would be accepted in the longest chain. In order to achieve rewards with higher probability, the malicious miner can generate thousands of valid and different blocks for his supported block. Consequently, it is against our nontransferability definition. Thus the nontransferability is not held.

In all, Merkle Tree based nonoutsourceable puzzles are not feasible under GHOST rule.

2P-PoW based scheme. Besides Merkle Tree based scheme. In [16], Ittay and Emin proposed another constructions informally: Two-Phase Proof of Work (2P-PoW). The details of two phases are as follows (we follow the descriptions given in [16]):

- The double hash of the $header$ ($SHA256(SHA256(header))$) is smaller than a difficulty parameter X, where $header$ includes puz, m, r and so on;
- The $header$ is signed with the coinbase transaction's private key, and the hash ($SHA256(SIG(header, privkey))$) of that signature is smaller than a second difficulty parameter Y.

The verification step is to check whether the two phases are valid. The first phase is similar to the existing Bitcoin cryptopuzzle and current mining pools are good at solving it. X and Y are the variables known as "difficulty" in the Bitcoin network. After they find a solution as usual, phase 2 requires that they would sign the block with the private key that is related to the payment address, and check whether the result after being hashed is smaller than a second difficulty parameter Y.

It can be divided into two cases according to the signature schemes used in the $SIG(\cdot)$ step in the above two-phase proof of work.

- If the $SIG(\cdot)$ step uses deterministic signature scheme, such as RSA, the miners could outsource the first phase to find many candidated $header$s and then check which $header$ satisfies the second phase by themselves.
- If the $SIG(\cdot)$ step uses nondeterministic signature scheme, such as ECDSA, the second phase does not preserve the existing mining infrastructure which can do hash computations very fast. It is because that the nondeterministic signature scheme would have thousands of valid signatures for certain message and miners would spend lots of time finding which signature could satisfy the second phase.

Ittay Eyal and Emin Gün Sirer thought the miners who have invested on the hash computing infrastructure play a critical role in the Bitcoin ecosystem by maintaining the Bitcoin blockchain. Therefore, 2P-PoW based puzzle would adopt deterministic signature scheme. The authors considered that the scheme is nonoutsourceable because the second phase is related to the private key. However, the work of first phase, finding candidated $header$s, can be outsourced. Thus, the scheme still remains part outsourceability and can not reach our goals.

IV. A NONOUTSOURCEABLE PUZZLE UNDER GHOST RULE

A. Our scheme

The common puzzle in Bitcoin can be outsourced because what the workers find does not help to create their own blocks. The key idea of Merkle Tree based nonoutsourceable schemes is to replace m with some secret parameters (e.g. a Merkle Tree) in the process of finding correct random numbers and effective outsourcing protocol would reveal some information about the secret Merkle Tree at the scratch attempts. Thus, the nonoutsourceability is guarded. However, as discussed previously, the Merkle Tree based scheme doesn't hold non-transferability and nonoursourceability at the same time. It is because the payload m does not contribute to finding the solutions that the transferability problem exists in Merkle Tree based schemes. Different from Miller's scheme, the step $SIG(\cdot)$ in 2P-PoW based scheme signes with the private key related to the public key in m. Though m does not contribute to finding the solutions directly, part of important information in it does. Thus, the solution is related to the specific payload and the scheme does not have transferability problems.

Intuitively, the Merkle Tree based scheme is to find some Merkle proofs that satisfy the difficulty and 2P-PoW based scheme uses the private key related to the payment address to sign the $header$ so that the 2P-PoW based scheme does not have transferability problem. Our idea is to combine the strengths of Merkle Tree based schemes and 2P-PoW based scheme. We are going to sign the puzzle and payload with the private key related to the payment address and the solution is related to the signature result. Moreover we adopt nondeterministic signature schemes to replace the nonce.

The (t', μ, d, t_0)-puzzle has three functions $(\mathcal{G}, Find, Verify)$ and is defined as follows:

- $\mathcal{G}(\lambda) \to puz$: generate a puzzle instance.
- $Find(puz, sk, m, t) \to ticket$: The $Find$ algorithm takes a puzzle instance puz, the secret key sk related to the payment address, the payload m, and time parameter t as input. It makes t unit scratch attempts, using $t \cdot t' + t_0$ time steps in total, where $t' = poly(\lambda)$ is the unit scratch time and t_0 can be thought as the initialization and finalization cost of $Find$. The $ticket$ contains the signature $SIG(puz \parallel m)$ such that,

$$H(puz \parallel SIG(puz \parallel m) \parallel m) < 2^{\lambda - d} \qquad (1)$$

  where H is a hash function and $SIG(\cdot)$ is a nondeterministic signature function.
- $Verify(puz, m, ticket) \to \{0, 1\}$: if the ticket is a valid for a specific instance puz, a payload m, it satisfies the following conditions:
  - check whether the signature $SIG(puz \parallel m)$ is a valid signature for the puzzle puz and the payload m with the public key related to the payment address;
  - check whether the inequality (1) is valid.

  If the ticket passed this check, we refer to it as a "good" ticket for (puz, m) and output 1, otherwise, output 0.

As for the $SIG(\cdot)$, we stress that it is a nondeterministic signature scheme with the private key related to the payment address. If we adopt a deterministic signature, then after a failed scratch attempt, miner would choose a new m and try again. If so, most of time would be spent in computing the

signature instead of computing the hash value, which does not preserve the existing mining infrastructure. Thus, here we adopt ElGamal signature scheme [22]. The signature steps are as follows:

- Key generation: p is a large prime number with generator g and randomly choose a secret key x with $1 < x < p-1$. Compute $y = g^x \ mod \ p$. The public key is y and the secret key is x.
- Sign: For the message $(puz \parallel m)$, choose a random k such that $1 < k < p-1$ and $gcd(k, p-1) = 1$ and compute $r = g^k \ mod \ p$ and $s = (H(puz \parallel m) - xr)k^{-1} \ mod \ (p-1)$, where H is a hash function. If $s = 0$, start over again, otherwise output the pair (r, s) as the digital signature of $(puz \parallel m)$.
- Verification: 1) $0 < r < p$ and $0 < s < p-1$; 2) $g^{H(puz \parallel m)} = y^r r^s$. If those two conditions are all satisfied, accept the digital signature, otherwise reject.

Therefore, the inequation (1) becomes

$$H(puz \parallel (r, s) \parallel m) < 2^{\lambda - d} \tag{2}$$

where (r, s) is the digital signature of $(puz \parallel m)$. To avoid spending much time in computing the signature, the first k is generated randomly and then k adds 1 after each failed scratch attempt so that it need not do exponentiation computations.

B. Analysis of our scheme

We prove that our scheme is a scratch-off puzzle. Based on it, we further analyze the nontransferability and nonoutsourceability.

Theorem 1: Our construction in section IV-A is a scratch-off puzzle.

In fact, our construction is similar to the Merkle Tree based construction. In the Merkle Tree based scheme, miners are working on finding some valid proofs that satisfy the difficulty, while miners are finding a ElGamal signature in our scheme. Thus, our construction is also a scratch-off puzzle intuitively. A formal proof is as follows.

Proof 1: Correctness, feasibility and parallelizability are trivial, so we just focus on the incompressibility proof now.

Suppose that there are two random oracles $\{\mathcal{O}_\infty, \mathcal{O}_\in\}$. $\mathcal{O}_\infty$ is to answer the signature while $\mathcal{O}_\in$ is to answer the hash value.

For any miner A with at most t scratch attempts, he does not know the secret parameters and would mine under the help of oracles. The probability of succeeding in finding a valid ticket is no more than $\zeta^+(t)$. And a good scratch attempt consists of two random oracle queries when finding a solution: 1) computing the signature; 2) computing the final hash value. And the hash computation of $(puz \parallel m)$ only needs one time during all the scratch attempts. For the signature, no polynomial-time adversary can output a new one except with negligible probability. For the final hash value, the probability of predicting the hash result is also negligible since we adopt a secure hash function. Thus, it is necessary for the miner A to make numbers of random oracle calls for each good scratch attempt. Therefore, our puzzle is μ-incompressible for an appropriate choice of μ.

Thus, our construction in section IV-A is a scratch-off puzzle.

Since we have proved that our construction is a scratch-off puzzle, we can analyze further the property of non-transferability and nonoutsourceability of which definitions are based on the definition of a scratch-off puzzle. We talk about nontransferability first. Because of the GHOST rule, this property requires that when creating a new block, not only other nodes in the Bitcoin network but also the new block's creator can not ease the work of finding another blocks.

Theorem 2: Our construction in section IV-A is nontransferable.

Proof 2: Without loss of generality, we assume that the miner A has found a solution $ticket$ for (puz, m) and it is divided into two cases:

- For those miners that know the secret key related to A's payment address, though they can easily get a different payload m^*, the probability of $H(puz \parallel m) = H(puz \parallel m^*)$ is negligible. Since the hash value of $(puz \parallel m^*)$ is different, to have the same signature (r, s) is negligible. Thus, to find the solution of (puz, m^*) is still difficult.
- For those normal miners that know nothing about the secret key related to A's payment address, it can be implied from the above case that the $ticket$ for (puz, m) does not help to create a new block.

In all, it is still difficult to create a new block when knowing the solution $ticket$ for (puz, m).

Theorem 3: Our construction in section IV-A is nonoutsourceable.

Proof 3: We assume that the puzzle can be outsourced and then there is an effective outsourcing protocol (t_e, t_w, t_o)-protocol. In aforementioned discussion, we know that for a scratch attempt, there are one hash computation and one signature computation. The hash function is public so it is not a problem for workers to compute the hash value. However the signature is computed with the secret key related to the payment address. If for every scratch attempt, the worker needs to ask the pool operator for one signature result, then t_o would occupy a large fraction of t_e, which shows that the outsourcing is not effective.

Thus, in order to be an effective protocol, the signing work should be done by the workers. As a result, the pool operator needs to share the secret key with all workers. Since the secret key is related to the payment address, anyone who knows the key can spend the reward, such as payment, transfer. Therefore, the pool operator would loss the reward with a high probability if there is a dishonest and malicious worker.

Thus, our construction is nonoutsourceable.

V. Experimental results

In order to show that our scheme is practical, we implemented our puzzle to estimate the verification time and puzzle size (the total size of solution or ticket for the puzzle). And then compare our results with with other schemes. The experiments

TABLE I
ESTIMATED VERIFICATION TIME AND PUZZLE SIZE

Scheme	Verification Time (μs)	Ratio[1]	Size (Byte)	Ratio[2]
Bitcoin	0.039	1	80	1
MT-based weak puzzle	14	359	4500	56
Our scheme (len=128)	16	410	112	1.4
Our scheme (len=256)	31	795	144	1.8
Our scheme (len=512)	100	2564	208	2.6
Our scheme (len=1024)	494	12667	336	4.2

[1] verification time divided by the verification time of Bitcoin.
[2] size divided by the size of Bitcoin.

are executed on a single server node with Intel(R) E7-2830 CPU and the test code is written by C language. For simplicity, we use the "SHA-1" hash function and the "BN" data structure in the openssl library. SHA-1 hash function provides a 160-bit hash value.

Metrics. We are interested in two performance criteria: verification time and puzzle size. Since some cryptocurrencies have much shorter block interval (e.g. Litecoin [2], Dogecoin [14] and Fastcoin [15]) than Bitcoin, we do not want the verification time to be a barrier for practical usage. The verification time displayed in Table I is the average result of 1000 tests. As for the size, we consider the puzzle only and do not include the transacftions in the verification time and size.

The puzzle size of Bitcoin is referred to Bitcoin Developer Reference [23]. For the Merkle Tree based weak puzzle, we set $q = q^{'} = 10$ and the tree has 10 level if the root is at 0-level so that the tree has 1024 leaves. As noted in [12], "this can provide roughly 50 bits of security for the nontransferability property" (Though this nontransferability is different from ours in Definition 3, it shows that these parameters are set reasonably). As for the $SIG(\cdot)$ in our puzzle, we adopt ElGamal signature scheme and the length of the prime is denoted by len. The detailed data is shown in Table I.

Results. From the experiments, we see that the verification time of these nonoutsourceable puzzles are all hundred times of Bitcoin. However, the absolute verification time are acceptable , even based on 1024-bit prime. At the time of writing this paper, each block is estimated to have 1750 transactions [24]. If we consider the transaction verification time, the puzzle verification time would be a tiny part of the full verification time. In addition, our puzzle size outperforms Merkle Tree based puzzle size since we do not need so many tree nodes for puzzle computation. Consider a transaction of about 500 bytes [24] and the maximum block size of 1 MB [13], we can spare more space to hold 8 more transactions when compared with Merkle Tree based puzzle. The cost per transaction is estimated about 6 USD [24] at the time of writing this paper, so each block in our construction can earn 48 USD more.

VI. CONCLUSIONS

Avoiding centralized mining pools and scalability are two important issues in blockchain. Existing solutions can only work on either one issue, but not both. In particular, the nonoutsourceable puzzles can resolve the former issue, but are not compatible with high-rate transaction processing protocol such as GHOST. In this paper, we provide a novel puzzle that can tackle both issues at the same time. We formally prove that our proposed puzzle is secure and experimental results show that our construction is practical.

ACKNOWLEDGEMENT

This project is in part supported by RGC funding, Hong Kong (Project No.: CityU C1008-16G).

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[2] Litecoin project. Accessed December 13, 2016. [Online]. Available: https://litecoin.org/
[3] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 397–411.
[4] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," *URL https://github.com/ethereum/wiki/wiki/% 5BEnglish% 5D-White-Paper*, 2014.
[5] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, "Permacoin: Repurposing bitcoin work for data preservation," in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 475–490.
[6] K. Delmolino, M. Arnett, A. E. Kosba, A. Miller, and E. Shi, "Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab." *IACR Cryptology ePrint Archive*, vol. 2015, p. 460, 2015.
[7] E. Heilman, F. Baldimtsi, and S. Goldberg, "Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 43–60.
[8] G. Zyskind, O. Nathan *et al.*, "Decentralizing privacy: Using blockchain to protect personal data," in *Security and Privacy Workshops (SPW), 2015 IEEE*. IEEE, 2015, pp. 180–184.
[9] V. Buterin. Bitcoin network shaken by blockchain fork. Accessed December 13, 2016. [Online]. Available: https://bitcoinmagazine.com/articles/bitcoin-network-shaken-by-blockchain-fork-1363144448
[10] Decentralized bitcoin mining pool. Accessed December 13, 2016. [Online]. Available: http://p2pool.org/
[11] L. Luu, Y. Velner, J. Teutsch, and P. Saxena, "Smart pool: Practical decentralized pooled mining." *IACR Cryptology ePrint Archive*, vol. 2017, p. 19, 2017.
[12] A. Miller, A. Kosba, J. Katz, and E. Shi, "Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 680–691.
[13] Block size limit controversy. Accessed December 13, 2016. [Online]. Available: https://en.bitcoin.it/wiki/Block_size_limit_controversy
[14] The dogecoin project. Accessed December 13, 2016. [Online]. Available: http://dogecoin.com/
[15] Fastcoin. Accessed December 13, 2016. [Online]. Available: http://www.fastcoin.ca/

[16] E. Ittay and G. S. Emin. (2014) How to disincentivize large bitcoin mining pools. Blog post. Accessed December 10, 2016. [Online]. Available: http://hackingdistributed.com/2014/06/18/how-to-disincentivize-large-bitcoin-mining-pools/
[17] C. Decker and R. Wattenhofer, "A fast and scalable payment network with bitcoin duplex micropayment channels," in *Symposium on Self-Stabilizing Systems*. Springer, 2015, pp. 3–18.
[18] J. Poon and T. Dryja, "The bitcoin lightning network," 2015.
[19] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, 2016, pp. 45–59.
[20] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 507–527.
[21] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 528–547.
[22] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Workshop on the Theory and Application of Cryptographic Techniques*. Springer, 1984, pp. 10–18.
[23] Bitcoin developer reference. Accessed December 11, 2016. [Online]. Available: https://bitcoin.org/en/developer-reference#block-headers
[24] Blockchain info. Accessed December 12, 2016. [Online]. Available: https://blockchain.info