(Short Paper): PieceWork: Generalized Outsourcing Control for Proofs of Work

Philip Daian[1], Ittay Eyal[1], Ari Juels[2], and Emin Gün Sirer[1]

[1] Department of Computer Science, Cornell University,
phil@cs.cornell.edu,ittay.eyal@cornell.edu,egs@systems.cs.cornell.edu
[2] Jacobs Technion-Cornell Institute, Cornell Tech
juels@cornell.edu

Abstract. Most prominent cryptocurrencies utilize proof of work (PoW) to secure their operation, yet PoW suffers from two key undesirable properties. First, the work done is generally wasted, not useful for anything but the gleaned security of the cryptocurrency. Second, PoW is naturally outsourceable, leading to inegalitarian concentration of power in the hands of few so-called pools that command large portions of the system's computation power.
We introduce a general approach to constructing PoW called *PieceWork* that tackles both issues. In essence, PieceWork allows for a configurable fraction of PoW computation to be outsourced to workers. Its controlled outsourcing allows for reusing the work towards additional goals such as spam prevention and DoS mitigation, thereby reducing PoW waste. Meanwhile, PieceWork can be tuned to prevent excessive outsourcing. Doing so causes pool operation to be significantly more costly than today. This disincentivizes aggregation of work in mining pools.

1 Introduction

Distributed cryptocurrencies such as Bitcoin [18] rely on the equivalence "computation = money." To generate a batch of coins, clients in a distributed cryptocurrency system perform an operation called *mining*. Mining requires solving a computationally intensive problem involving repeated cryptographic hashing. Such problem and its solution is called a Proof of Work (PoW) [11].

As currently designed, nearly all PoWs suffer from one of two drawbacks (or both, as in Bitcoin). First, due to the computationally intensive nature of PoWs, miners of popular cryptocurrencies such as Bitcoin and Ethereum require massive computing hardware and consume natural resources such as electricity. As mining serves no purpose other than maintaining blockchain security, these resources are otherwise wasted. Second, the cost advantages of special-purpose mining equipment and a desire to reduce the variance of mining rewards incentivize the concentration of mining effort in large *mining pools*. Such concentration of power in the hands of a small number of entities erodes the egalitarian founding principles of most decentralized cryptocurrencies, starting with Bitcoin.

There are several proposed solutions to first problem of costly and difficult-to-repurpose PoWs. Primecoin [14] is an alt-coin in which mining involves discovery of long sequences of prime numbers. The Primecoin PoW achieves a secondary goal beyond blockchain security, but the economic value of its byproduct remains unclear. In Permacoin [16], the mining process is replaced by proofs-of-retrievability [13], which prove that miners are storing a large corpus of data [16]. Permacoin, however, recoups only a small fraction of wasted resource, and does not recycle computational resources. Indeed, despite such efforts, the Bitcoin FAQ[3] continues to claim that, "To provide security for the Bitcoin network, the calculations involved need to have some very specific features. These features are incompatible with leveraging the computation for other purposes".

To address the problem of mining centralization, some work has explored the idea of preventing PoW outsourcing. Examples include Nonoutsourceable Scratch-Off Puzzles [17], 2 Phase-Proof of Work (2P-PoW) [10], and Sign to Mine [1]. The idea behind these schemes is to base mining on the use of a private key that controls mining revenue. Thus outsourcing in, e.g., a mining pool would expose the outsourcer to theft.

Other areas of work on proof of work outsourcing involve studying solutions to attacks on outsourcing work proofs. In such attacks, an unscrupulous worker that finds a full PoW solution might choose not to submit it to the outsourcer, a problem called *withholding*. Workers can, in many cases, act in this way to harm an outsourcer's overall profit at little to no cost to themselves, as they are still getting compensated for partial solutions. (Another, blockchain-level form of this attack is known as the block withholding attack [7] [9].)

Our contribution: PieceWork

We introduce *PieceWork*, a generalized scheme for restructuring standard hash-based PoWs that addresses the two drawbacks of existing PoWs described above. As we explain, PieceWork encompasses a number of existing PoW construction ideas, particularly from [10, 11]. PieceWork decomposes a PoW into two sequential exponentially distributed computational problems called *puzzles*. In PieceWork, a PoW consists of a k_{in}-bit hard *inner* puzzle and a k_{out}-bit-hard *outer* puzzle. We call this modification *two-stage hashing* [10].

Inner puzzles are *outsourceable* as small units of work called *puzzlets*. A miner can delegate puzzlet-solving safely to other, potentially untrusted workers. Puzzlets in PieceWork are also *reusable*, meaning that they can serve *useful goals beyond blockchain security*. These include spam deterrence [8, 2], denial-of-service mitigation [12], MicroMint coin generation [11, 22], Tor relay payments [4], and more. As an enhancement, we show how puzzlets can be computed by workers non-interactively. Our puzzlets are based on the computation recycling ideas ("breadpudding protocols") in [11]. That work predated Bitcoin, though, and thus didn't address distributed cryptocurrencies and problems such as withholding [21], a significant barrier preventing the reuse of outsourced work in PoW currencies today.

In contrast, outer puzzles can be *non-outsourceable*, i.e., solved safely only by the miner receiving the mining reward for a given PieceWork PoW. For example,

[3] Referenced 11 Dec. 2016 at https://en.bitcoin.it/wiki/FAQ.

by leveraging the mechanism 2P-PoW, PieceWork can cause outsourcing of outer puzzles to result in exposure to theft of mining rewards.

PieceWork permits tuning of k_{in} and k_{out}, and thus the amount of permissible outsourcing in a cryptocurrency. Through gradual adjustments to k_{in} and k_{out}, PieceWork thus also supports graceful *migration from outsourceable to non-outsourceable work*. By inducing changes slightly over time, PieceWork can enable a mining community to adjust its equipment and organization over time.

In summary, our contributions in introducing PieceWork are as follows:

– *Unified PoW outsourcing framework:* PieceWork offers a unified PoW construction that incorporates a number of previously proposed ideas on safe (withholding-resistant) outsourcing, reusable PoW work, tunable outsourcing, and prevention of outsourcing in mining pools. PieceWork adapts these ideas, some predating Bitcoin, to modern cryptocurrencies and specifies them precisely, as some proposed ideas include unspecified details.
– *PoW reuse:* By offering concrete examples of computation reuse in PieceWork, we show that PoWs can both enforce blockchain security and serve practical and economically valuable secondary goals—refuting the Bitcoin Wiki claim to the contrary. Additionally, we show how puzzlets can be computed by workers non-interactively, making their deployment practical for the applications we describe.
– *Novel technical extensions:* PieceWork includes novel technical extension to previous ideas, including non-interactive outsourcing and double-harvesting.

2 PieceWork: Two-Stage Hashing, Puzzles, and Puzzlets

We now present the technical details of PieceWork. We start with a description of how hash-based PoWs work, and then describe how such PoWs are modified in PieceWork.

2.1 Background: Hash-based PoWs

Most PoWs in distributed cryptocurrencies adhere to the same general structure as that in Bitcoin, which we focus on for concreteness. Our description here and of PieceWork thus generalize to other cryptocurrencies (e.g., Ethereum).

The Bitcoin PoWs involves finding a valid solution n to the following problem:

$$\begin{aligned}\texttt{SHA-256}^2\{v \parallel B_l \parallel \mathrm{MR}(\mathrm{TR}_1, \ldots, \mathrm{TR}_n) \parallel T \parallel n\}\\ \leq \mathit{target},\end{aligned}$$

where v is a (software) version number, B_l denotes the last generated block, $TR_1, \ldots TR_n$ is a set of valid transactions not yet confirmed, $MR(x)$ denotes the root of the Merkle tree over transactions x, T is the current Unix timestamp, n is a nonce in the space N, and *target* is a 256-bit value that determines the difficulty of the mining operation. It is updated according to the generation times of the last 2016 blocks.

We may abstract away the details of the mining problem by defining

$$X = v \parallel B_l \parallel \mathrm{MR}(\mathrm{TR}_1, \ldots, \mathrm{TR}_n) \parallel T.$$

to be the collection of inputs specific to a block. We let $H(\cdot)$ represent the hashing operation `SHA-256`^{2} and, for brevity, let $Z = target$.

A Bitcoin mining operation then involves, for block value X, the discovery of an input ("nonce") $n \in N$ for which $H(X, n) \leq Z$. We refer to this hash-inversion problem as the "Bitcoin puzzle", designed to achieve several properties essential to the Bitcoin system described in [16]: predictable effort, fast verification, and precomputation resistance.

2.2 Basic PieceWork scheme

PieceWork relies on a hierarchical form of hashing that we call two-stage hashing. In PieceWork, we partition the hash function H into a pair F_{in} and F_{out} of sequentially composed functions that we refer to as the "inner" and "outer" puzzles. A global puzzle is then of the following form:

$$H(X, n) = F_{out}(X, F_{in}(X, n; s)).$$

and is considered valid when the inner and outer puzzles evaluate to below the respective targets. Here, s is an extra input used for the purposes of puzzlet recycling and discussed in detail in Section 3.

We refer to the inner function as a *puzzlet*. A valid solution to a puzzlet is a pair (n, s) that satisfies $I = F_{in}(X, n; s) \leq Z_{in}$.

A solution (n, s) to a puzzlet is also a solution to the global puzzle if it satisfies the additional condition $F_{out}(X, I) \leq Z_{out}$.

Both F_{in} and F_{out} must have the additional desired conditions of being cheap to compute, and being independently identically distributed across instances. The former condition allows for the fast verification required in the global scheme, and the latter allows for an exponential block generation curve that can be tuned predictably by adjusting the target. In general, we focus on hash functions or functions that hash the results of a constant-time function to achieve the latter property. This includes the double-SHA256 scheme currently used in Bitcoin.

In PieceWork, an outsourcer provides a puzzlet to a worker with a specified value of s (whose selection we explain in Section 3). Thus an outsourced puzzlet P takes the form:

$$P = (X, Z_{in}, s).$$

The task of the worker is to find an n such that (n, s) solves a puzzlet. The expected computation of the worker is R/Z_{in} executions of F_{in}. The outsourcer can, however, quickly check the correctness of a solution (n, s) to P.

Each solution to P represents one or more potentially valid preimages for F_{out} for the outsourcer to try. On average, the outsourcer must try R/Z_{out} inputs to F_{out} to find a solution to the global puzzle.

Tunability. Tuning inner and outer puzzles to any desired difficulty is straightforward. By setting Z_{in} and Z_{out}, an expected number of hash iterations $2^{k_{in}}$ and $2^{k_{out}}$ can be enforced for inner and outer puzzles respectively. Such tunability is a feature of 2P-PoW [10], and thus PieceWork can support the migration from higly outsourceable to outsourcing resistant mining proposed there.

Non-outsourceability of outer puzzles. By choosing F_{out} appropriately, it is possible to make outer puzzles non-outsourceable. We discuss possible approaches in Section 3.3.

2.3 Full PieceWork scheme: Adding withholding resistance

Bitcoin puzzles in their current form are in fact already outsourceable. Mining pools can outsource a block solution puzzles to miners (workers in our scheme), and reward these miners for *partial proofs of work*, or solutions to the block problem that satisfy some weaker target than the global difficulty target.

Block withholding arises when a worker *can determine whether her work constitutes a full PoW solution.* In the basic version of PieceWork specified above, a worker can determine whether puzzlet I represents a global puzzle solution. She can then choose to withhold it from the outsourcer.

A solution to this problem is to conceal from a worker whether or not her solution to an outsourced puzzle represents a full PoW solution. In PieceWork, such concealment is possible with a slight enhancement to the basic PieceWork scheme as follows:

$$PW(X, n) = F_{out}(X, F_{in}(X, n; s, r_{in}), r_{out}), \tag{1}$$

where $r_{in} = H_0(r)$ and $r_{out} = H_1(r)$ for distinct hash functions H_0, H_1 and a secret value r. Thus a puzzlet takes the form:

$$P = (X, Z_{in}, s, r_{in}). \tag{2}$$

Note that the dependence between r_{in} and r_{out} is important: If r_{in} were selectable by the outsourcer independently of r_{out}, the outsourcer could, for a single puzzlet solution I, solve for a valid r_{out}, and, with $1/Z_{out}$ work on expectation, easily find a global puzzle solution.

Withholding was called out as urgent on the Bitcoin developer mailing list in 2015 [21]. The mailing list post on block withholding mentions a "two-stage target mechanism" that may perhaps resemble our scheme; we were able to find one public reference to the details such a scheme in [23]. That solution suffers from potential rounding bias, lacks a full specification, and postdates a scheme developed by Back to solve similar withholding problems in original implementations of HashCash [3].

3 Applying PieceWork

We now discuss the application of PieceWork. We explain how puzzlets in PieceWork can be used to recycle computation. We also explain how puzzlets can be computed by workers non-interactively, making their deployment more practical. Then we show how PieceWork may be used to prevent outsourcing.

3.1 Outsourceable Puzzlet Applications

A puzzlet solution has an easily quantifiable expected value for an outsourcer in PieceWork. Suppose that V is the value generated by a successfully mined block. Then the expected value of a puzzlet solution is V/Z. Their value is probabilistic, much like micropayments in [15], but may be made non-probabilistic by an outsourcer joining a traditional mining pool.

By judicious setting of s, outsourceable puzzlets can be used to perform useful computations in other domains. Interactive applications with short timeouts are preferred, allowing for a high probability that a puzzlet will be applicable to the current latest Bitcoin block. In this section, we describe some sample applications and effective choices for s that accomplish these goals.

Spam deterrence Dwork and Naor [8] proposed a scheme in which the sender of a piece of e-mail attaches the solution to a puzzlet. A receiver only accepts e-mail with a valid puzzlet solution. Puzzlets are receiver-specific in this scheme, so a would-be spammer incurs the high cost of solving puzzles for a large number of receivers. Dwork and Naor's puzzle construction was complicated, but can be easily replaced with a hash-based PoW, as in [2].

As a receiver of e-mail cannot easily transmit a newly generated, block-specific value s to a sender *before the sender transmits e-mail*, we propose that $s = H(\text{Digest}\|\text{Header})$ for some CRHF H.

DoS deterrence "Client puzzles" are hash-function inversion puzzles that a client must solve to receive a resource from a server, such as a TCP or TLS connection [12, 20] or DNS query information. This scheme helps deter DoS attacks, as it would require an attacker to solve many puzzles.

We can set $s = H(\text{Client IP}\|\text{fresh})$, with the freshness parameter being a shared random variable to prevent stale puzzle recycling.

MicroMint Rivest and Shamir [22] proposed a digital cash system called MicroMint, in which coins are minted via hash collisions. MicroMint mimics the economics of a real, physical mint, where there is a high base cost for design of coinage, the purchase of machinery, etc. The incremental cost of producing coins, though, is small. Similarly, MicroMint requires many hashes to find the first coinworthy collision. Subsequent collisions accumulate quickly thereafter.

Jakobsson and Juels [11] showed how the problem of computing a hash image can be made moderately hard so that the problem serves as a puzzlet. Their scheme can be easily instantiated in PieceWork. In this case, s is the hash of a secret minting key and an unique puzzlet index. (See [11] for details; some slight modifications to the original scheme are required for PieceWork.)

MicroMint outsourcing in PieceWork can be *combined* with outsourcing for DoS resistance, i.e., a worker can simultaneously help produce MicroMint coins *and* aid in DoS prevention. We call this idea *double-harvesting.*

Tor relay payments Biryukov and Pustogarov [4] proposed mining outsourcing as a means for clients to pay relays in Tor. Their scheme suffers in current schemes like Bitcoin from the withholding problem, and therefore would benefit from PieceWork. In one variant, a relay runs its own mining pool. In a second variant, a relay itself serves as a worker in a mining pool and further outsources work. This latter application motivates a possible variant of PieceWork involving extension to three-phase hashing.

3.2 Non-interactive puzzlet construction

Ideally, a worker could determine a puzzlet on its own. For example, an e-mail sender should not need to interact with the receiver to select a puzzlet [24].

In this case, the outsourcer may publish a public key PK (with corresponding private key SK) such that: (1) r_{in} may be computed from PK and X by the worker by means of a deterministic function f and (2) r_{out} may be computed from SK and X by the outsourcer by means of a deterministic function g. The correctness of r_{out} must additionally be publicly checkable.

As an example, let $(SK, PK) = (x, G^x)$ for G a generator of group $\mathbb{G}$ of order q in which Computational Diffie-Hellman is hard and $x \in_R \mathbb{Z}_q$ a randomly selected secret key. Then $f(PK, X) = H_f(PK, X)$ for a hash function H' and $g(SK, X) = H_g(X)^x$ for a suitable one-way function $H_g : \{0,1\}^* \rightarrow \mathbb{G}$.

The correctness of r_{out} may be proven using a NIZK proof. In this case, g together with the proof constitute a Chaum-Pedersen signature [6].

Alternatively, if $\mathbb{G}$ is an admissible group for a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}'$ [5], then it is possible to verify the correctness of r_{out} by checking the equality $\hat{e}(H_g(X), PK) \stackrel{?}{=} \hat{e}(r_{out}, G)$.

3.3 Non-Outsourceable Puzzlet Applications

An existing approach to outsourcing resistance represented by 2P-PoW and Sign to Mine, outlined informally in [10] and [1] respectively, can easily be plugged into the inner puzzles of PieceWork. These schemes involve puzzles based on the application of a digital signature, rather than a hash function. The proposal is that the private key for the puzzle should be identical to that for spending mining rewards. In our scheme, this would prevent outsourcers from pooling worker resources. In PieceWork, the outer function may be defined as, e.g.:

$$F_{out} = H(SIG_{privkey}(X, F_{in}(X, n; s, r_{in}))), \tag{3}$$

with the inner function representing the standard Bitcoin block solution, optionally at a lower reuseable difficulty. There are a few provisos. First, we emphasize that such nonoutsourceability is heuristic, and not accompanied by formal guarantees in the sense of "weak" outsourceability in [17]. It is possible in principle digital signing can be securely outsourced—meaning that a "helper" can substantially reduce the computation a signer needs to perform in computing a signature without the helper learning the private key. In practice, however, there is no known effective scheme for outsourcing computation in ordinary signature schemes such as RSA and discrete-log-based schemes, e.g., ECDSA [19]. Thus, signing-based puzzles may be heuristically assumed to prevent outsourcing.

Second, it has been argued (including in the comments of [10]) that, rather than disincentivizing large pools, such a scheme could support outsourcing in which workers place money in escrow that they forfeit should they steal mining rewards. We omit discussion of this argument here, but note that escrow schemes are complicated to implement and would disincentivize many workers, given that escrow amounts would need to match block reward amounts.

4 Conclusion

We have shown that computation in Bitcoin and similar cryptocurrencies need not be wasted, and outlined how a configurable percentage of this computation can be repurposed for protection against e-mail spam, denial of service, and other micropayment-style applications. We have established in PieceWork a framework for defining our puzzles, and unified 2-Phase-PoW, Sign To Mine, and tunably outsourceable two-stage puzzles that counter block withholding under a single model. We hope this will help future efforts in the outsourceable cryptocurrency computation space more effectively and rigorously define their schemes.

Acknowledgments

This work is funded in part by NSF grants CNS-1330599, CNS-1514163, CNS-1564102, CNS-1561209, and CNS-1518779, ARO grant W911NF-16-1-0145, and IC3 sponsorship from Chain, IBM, and Intel.

References

1. ziftrcoin: A cryptocurrency to enable commerces. `https://d19y4lldx7po3t.cloudfront.net/assets/docs/ziftrcoin-whitepaper-120614.pdf` (2014), accessed: 2016-11-05
2. Back, A.: Hashcash - a denial of service counter-measure. `http://www.hashcash.org/papers/hashcash.pdf` (2002)
3. Back, A.: Hashcash-amortizable publicly auditable cost functions. Early draft of paper (2000)
4. Biryukov, A., Pustogarov, I.: Proof-of-work as anonymous micropayment: Rewarding a tor relay. In: International Conference on Financial Cryptography and Data Security. pp. 445–455. Springer (2015)
5. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. Advances in Cryptology–CRYPTO 2001 pp. 213–229 (2001)
6. Chaum, D., Pedersen, T.P.: Wallet databases with observers (extended abstract). In: CRYPTO. pp. 89–105 (1992)
7. Courtois, N.T., Bahack, L.: On subversive miner strategies and block withholding attack in bitcoin digital currency. arXiv preprint arXiv:1402.1718 (2014)
8. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: CRYPTO. pp. 139–147 (1993)
9. Eyal, I.: The miner's dilemma. In: 2015 IEEE Symposium on Security and Privacy. pp. 89–103. IEEE (2015)
10. Eyal, I., Sirer, E.G.: How to disincentivize large bitcoin mining pools. `http://hackingdistributed.com/2014/06/18/how-to-disincentivize-large-bitcoin-mining-pools/` (2014), accessed: 2016-11-05
11. Jakobsson, M., Juels, A.: Proofs of work and bread pudding protocols. In: Communications and Multimedia Security. pp. 258–272 (1999)
12. Juels, A., Brainard, J.: Client puzzles: A cryptographic countermeasure against connection depletion attacks. In: NDSS. pp. 151–165 (1999)
13. Juels, A., Jr., B.S.K.: PORs: proofs of retrievability for large files. In: ACM CCS. pp. 584–597 (2007)
14. King, S.: Primecoin: Cryptocurrency with prime number proof-of-work. July 7th (2013)

15. Micali, S., Rivest, R.L.: Micropayments revisited. In: Proc. Cryptography Track at RSA Conference 2002. pp. 149–263 (2002)
16. Miller, A., Juels, A., Shi, E., Parno, B., Katz, J.: Permacoin: Repurposing bitcoin work for data preservation. In: 2014 IEEE Symposium on Security and Privacy. pp. 475–490. IEEE (2014)
17. Miller, A., Kosba, A., Katz, J., Shi, E.: Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 680–691. ACM (2015)
18. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. `http://bitcoin.org/bitcoin.pdf` (2008)
19. Nguyen, P., Stern, J.: The béguin-quisquater server-aided rsa protocol from crypto95 is not secure. In: ASIACRYPT. pp. 372–379. Springer (1998)
20. Nygren, E., Erb, S., Biryukov, A., Khovratovic, D.: TLS client puzzles extension draft-nygren-tls-client-puzzles-01. IETF Internet-Draft (2016), expires 30 Dec. 2016
21. Priest, C.: [bitcoin-dev] we need to fix the block withholding attack. `https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2015-December/012059.html` (2015), accessed: 2016-11-05
22. Rivest, R.L., Shamir, A.: PayWord and MicroMint–two simple micropayment schemes. In: International Workshop on Security Protocols. pp. 69–87 (1997), (Also published in RSA Laboratories' *CryptoBytes*, Spring 1996)
23. Todd, P.: Re: [bitcoin-dev] we need to fix the block withholding attack. `https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2015-December/012069.html` (2015), accessed: 2016-11-05
24. Waters, B., Juels, A., Halderman, J.A., Felten, E.W.: New client puzzle outsourcing techniques for dos resistance. In: ACM CCS. pp. 246–256 (2004)