Brief Announcement: ZeroBlock: Timestamp-Free Prevention of Block-Withholding Attack in Bitcoin

Siamak Solat(✉) and Maria Potop-Butucaru

UPMC-CNRS, Sorbonne Universités, LIP6, UMR, 7606 Paris, France
{Siamak.Solat,Maria.Potop-Butucaru}@lip6.fr

Abstract. Bitcoin was recently introduced as a peer-to-peer electronic currency in order to facilitate transactions outside the traditional financial system. The core of Bitcoin, the Blockchain, is the history of all transactions committed by the system. This distributed ledger is similar to a distributed shared register where miners write and read blocks. New blocks in the Blockchain contain the last transactions in the system and are added by miners after a block mining process that consists in solving a difficult cryptographic puzzle. Although, the reward is the main motivation for the mining process in Bitcoin, it also may be an incentive for attacks such as *selfish mining*. In this paper we propose and theoretically analyze a solution for one of the major problems in Bitcoin: *selfish mining* or *block-withholding* attack. This attack is conducted by adversarial miners in order to either earn undue rewards or waste the computational power of *honest* miners. Contrary to the best to date solution for preventing *block-withholding* [6], our solution, *ZeroBlock*, prevents this attack by using a novel timestamp-free technique that exploits the Poisson nature of the proof-of-work and the current knowledge on the propagation of information in Bitcoin [2]. Note that previous solutions are vulnerable to forgeable timestamps. Additionally, our solution is compliant with miners churn.

1 Introduction

In the last few years crypto-currencies are in the center of the research ranging from financial, political and social to computer science and pure mathematics. Bitcoin [1] was one of the starters of this concentration of forces. It targeted the creation of a system where transactions between individuals can escape the strict control of the banks and financial markets.

Bitcoin was introduced as a pure peer-to-peer electronic currency or crypto-currency. It aims at fully decentralization of electronic transactions. Bitcoin allows to perform online transactions directly from one party to another one "without" the interference of a financial institution as a "trusted third party" [1]. It uses digital signatures to verify the bitcoin ownership and employs Blockchain in order to prevent double-spending attacks. In this attack the same bitcoin can be spent several times by a dishonest party. Blocks in the blockchain are created via a proof-of-work (cryptographic puzzle) [5] performed by *honest* parties (miners that follow the protocol). Blockchain is further broadcasted via a peer-to-peer overlay in order to agree on a common history of the transactions in the system.

Bitcoin is still vulnerable to various attacks including double-spending [7], *selfish* mining [4], Goldfinger [8], 51% attack [8] etc. In this paper we focus the *selfish* mining attack. Recently, [3] provided a full description of incentives to withhold or *selfish* mine in Bitcoin. That is, to force *honest* miners to waste their computational power such that their public blocks become useless (as *orphan* block), whereas the private chain of the *selfish* miners is accepted as a part of the Blockchain. To this end, the *selfish* miners reveal selectively their private blocks to make useless the blocks made by *honest* miners.

Our contribution. Our solution builds on the following simple idea: if a *selfish* miner keeps a block private more than a fixed interval of time, its block will be rejected by all the *honest* miners. Zeroblock scheme strives to reduce the probability of *intentional* forks that are result of block-withholding attacks. With ZeroBlock scheme a selfish mining pool cannot achieve more than its expected reward. Only with a low probability, selfish mining pool may create intentionally an *unprofitable* fork. We accentuate “unprofitable”, because this fork does not lead to more reward for selfish mining pool, but also reduces selfish pool’s likelihood to earn unexpected reward regardless of to its mining power. Thus, selfish mining pool is not incentivized to create such fork if its purpose is to achieve more reward. Furthermore, we prove that the maximum probability of such *intentional* fork is very low (≈ 0.04) when selfish pool uses its maximum hashing power. We further extend ZeroBlock in order to be tolerant to miners churn. The details of our solutions and the correctness proofs are proposed in [9].

2 ZeroBlock Algorithm

The key idea of our solution is that each block must be generated and received by the network within *a maximum acceptable time for receiving a new block* interval, *mat* (see Eq. 6 below). Within a *mat* interval a *honest* miner receives or discovers a new block. Otherwise, it generates a dummy block. The computation of each *mat* interval is done locally by each miner based on the following Bitcoin parameters: the *expected delay for a block mining* and the *information propagation time* in the Bitcoin network.

Expected delay for a block mining in Bitcoin depends mainly on the difficulty of proof-of-work. The major part of proof-of-work consists in discovering a byte string, *nonce.* As pointed out in [2] proof-of-work in Bitcoin is a Poisson process and causes blocks to be discovered randomly and independently. Moreover, in Bitcoin, the difficulty of proof-of-work required to discover a block is periodically adjusted such that, on average, *one* block is expected to be discovered every *10 min.* Hence, the difficulty of proof-of-work is updated every 2016 blocks. It means that regarding to this adjustment (i.e. one block per 10 min) 2016 blocks, on average, is expected to be generated in 14 days. If 2016 blocks are discovered in a shorter time, the difficulty of proof-of-work will be increased and if they are generated in a longer time, difficulty of proof-of-work will be decreased.

The proof-of-work works as follows:

$$if\, H(pb + nonce) < T\, then proof\text{-}of\text{-}work succeeded \tag{1}$$

where *pb* represents the hash of the previous block, *nonce* is the answer of proof-of-work that must be found by miners, T is *target*, '+' is concatenation operation and H is the hash function.

Each mining pool can estimate the difficulty of proof-of-work using Eq. 2.

$$D = \frac{maxTarget}{T} \tag{2}$$

where D is the difficulty of proof-of-work, T is current *target* and *maxTarget* is maximum possible value for *target* that is $(2^{16}$ - $1)2^{208} \approx 2^{224}$. Since the hash function produces uniformly a random value between 0 and $2^{256} - 1$ thus, the probability that a given *nonce* value would be the answer of proof-of-work is as follows (Eq. 3):

$$Prob(nonce\ is\ answer) = \frac{target}{2^{256}} = \frac{2^{224}}{D \times 2^{256}} \approx \frac{1}{D \times 2^{32}} \tag{3}$$

The number of hashes to discover a block is $D \times 2^{32}$ in expectation. If a mining pool can calculate hashes at a rate *php* (we call this as pool's hashing power), then the expected time (or average time) *avt* in which this pool can discover a block is as follows (Eq. 4):

$$avt_{pool} = \frac{D \times 2^{32}}{php} \tag{4}$$

When we replace *php* by hashing power of the network, *nethp*, we can use Eq. 3 for the entire network as follows (Eq. 5):

$$avt_{net} = \frac{D \times 2^{32}}{nethp} \tag{5}$$

According to the relation between *time, difficulty of proof-of-work, hashing power of the network* in Eq. 5, Bitcoin network adjusts D such that regarding to hashing power of the network, the average time for block generation rate remains 10 min.

To calculate the *maximum acceptable time for receiving a new block*, *mat*, we use Eq. 6 below:

$$mat = avt_{net} + ipt \tag{6}$$

where avt_{net} is given by the Eq. 5 and *ipt* is the information propagation time in Bitcoin network as estimated in [2].

Algorithm 1. ZeroBlock algorithm

```
index ← 0                                                    ▷ index of mat
mat[index] ← 0                              ▷ mat at the beginning is set to zero
$avt_{net}$ ← block generation average time          ▷ according to equation (6)
localChain ← Genesis
FlagNewBlock ← False
nonce ← 0
HPrB ← 0                                               ▷ hash of previous block
T ← target
newChain ← Null
ansPoW ← 0                                                  ▷ answer of PoW
scounter() ← 0                                  ▷ scounter() is a seconds counter
while (True) do
    if (FlagNewBlock = False) AND (mat[index] $\neq$ 0) then
        dummy Zeroblock ← SHF(getHead(localChain)) + SHF("FixedStringZB") + index
        localChain ← join(dummy Zeroblock,localChain)
    end if
    index ← index + 1
    refresh(mat[index])
    while (scounter() $\leq$ mat[index]) do
        newChain ← checkInput()
        if (newChain $\neq$ Null) then
            HPrB ← SHF(getHead(localChain))
            if (FHF(HPrB,newChain.ansPoW) $\leq$ T) then           ▷ proof-of-work is done
                localChain ← newChain
                newChain ← Null
                FlagNewBlock ← True
                Break
            end if
        end if
        if (scounter() < $avt_{net}$) then
            if (FlagNewBlock = False) then
                HPrB ← SHF(getHead(localChain))
                if (FHF(HPrB , nonce) $\leq$ T) then            ▷ proof-of-work succeeded
                    ansPoW ← nonce
                    localChain ← join(GenerateBlock(),localChain)
                    BroadcastBlock(localChain,ansPoW)
                    FlagNewBlock ← True
                    nonce ← 0
                    Break
                end if
                nonce ← nonce + 1
            end if
        end if
    end while
end while
```

The ZeroBlock algorithm (Algorithm 1) uses the following parameters and definitions: *ipt* : information propagation time in Bitcoin network that is an average delay for propagation a block into the network. This average delay has been estimated by simulation in [2]. *avt* : block generation rate that has been set by Bitcoin protocol according to which the difficulty of proof-of-work is adjusted regarding to the hashing power of the network using Eq. 5. *mat* : maximum acceptable time for receiving a new block that is computed by Eq. 6. During a *mat* interval if a miner cannot solve the proof-of-work, it has to generate a dummy Zeroblock. *unpermitted block-withholding* : occurs when a *selfish* mining pool discovers a new block and keeps the block private after the end of the current *mat* interval. *Dummy Zeroblock* : is generated locally by miners. It includes the index of *mat* interval and the hash of previous block. It is generated by honest miners to prevent *unpermitted block-withholding*. Note that our solution uses *standard Bitcoin blocks* discovered by solving the proof-of-work and *dummy blocks* that are generated by the Zeroblock algorithm for which miners do not need to solve any proof-of-work. The dummy Zeroblocks time generation is therefore ignored when adjusting the difficulty of the proof-of-work. *orphan block* : a block that has been discovered but is then rejected by the network. *genesis block* : the first block of a Blockchain on which all miners have a consensus. *correct chain* : a chain whose blocks have been discovered and inserted correctly according to the described protocol. *creative miner*: a miner that in a *mat* interval can solve proof-of-work and then generates a new block.

References

1. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Consulted **1**(2012), 28 (2008)
2. Decker, C., Wattenhofer, R.: Information propagation in the bitcoin network. In: 2013 IEEE Thirteenth International Conference on Peer-to-Peer Computing (P2P). IEEE (2013)
3. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 436–454. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45472-5_28
4. Eyal, I.: The miner's dilemma. 2015 IEEE Symposium on Security and Privacy (SP). IEEE (2015)
5. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 139–147. Springer, Heidelberg (1993). doi:10.1007/3-540-48071-4_10
6. Heilman, E.: One weird trick to stop selfish miners: fresh Bitcoins, a solution for the honest miner (Poster Abstract). In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014. LNCS, vol. 8438, pp. 161–162. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44774-1_12
7. Decker, C., Seider, J., Wattenhofer, R.: Bitcoin meets strong consistency. In: Proceedings of the 17th International Conference on Distributed Computing and Networking, Singapore (2016)
8. Kroll, J.A., Davey, I.C., Felten, E.W.: The economics of Bitcoin mining, or Bitcoin in the presence of adversaries. In: Proceedings of WEIS, vol. 2013 (2013)
9. Solat, S., Potop-Butucaru, M.: ZeroBlock: Preventing selfish mining in Bitcoin in CoRR abs/1605.02435 (2016). http://arxiv.org/abs/1605.02435