Extended abstract of a presentation at Scaling Bitcoin Workshop 2019 (no proceedings).

Proof-of-Verification for Proof-of-Work: Miners Must Verify the Signatures on Bitcoin Transactions

Kanta Matsuura (The University of Tokyo)

1 Introduction

1.1 Problem Description

Public-key primitives cost a lot in computation. Therefore, in the Proof-of-Work (PoW) of Bitcoin, rushing miners may skip the verification of the signatures on Bitcoin transactions to find a successful nonce as soon as possible. Such dishonest behaviors may allow malicious transactions to be included in a new block, and reduce the security of Bitcoin. Due to the open-source nature of Bitcoin, the above risk could not be ignored. One might think that participants in a mining pool would not care much about the computational cost of signature verification. However, if the scalability of Bitcoin is improved and the number of transactions per block is increased, their incentive to skip signature verification would get even bigger.

This extended abstract is a proposal of speaking (and discussing) how to avoid such behaviors without doing harm to the scalability of Bitcoin.

1.2 Digital Signature

As an example, let us revisit the Schnorr's signature scheme [1]. This scheme employs a subgroup of order q in $\mathbf{Z}_p^*$ where p is a large prime. By using a secure hash function $h : \{0,1\}^* \to \mathbf{Z}_q$, the scheme is described as follows.

Key generation:

1. Generate a large random prime p and a generator g of the multiplicative group $\mathbf{Z}_p^*$.
2. Select a random integer x such that $1 < x \leq p-2$.
3. Compute $y = g^x \bmod p$.
4. Let (p, g, y) be the public key, and let x be the private key.

Signature generation on a document m:

1. Select a random integer k such that $1 \leq k \leq q-1$.
2. Compute $r = g^k \bmod p$, $e = h(m\|r)$, and $s = xe + k \bmod q$.
3. The signature for m is the pair (s, e).

Signature verification:

1. Compute $v = g^s y^{-e} \bmod p$ and $e' = h(m\|v)$.
2. Accept the signature if and only if $e' = e$.

2 Proposal

In the case of the Schnorr's signature scheme, the parameter v (or its hashed value) can be a proof of having verified the signature without skipping the heavy modular exponentiation; v is a reconstruction of r, and the verifier would not know its value without the modular exponentiation [2]. In many of digital signature schemes (e.g. DSA [3] and a shortened DSS [4]) based on the hardness of discrete logarithm (DL) problem, we can find such parameter(s) that can be a Proof-of-Verification (PoV). This applies also to

DL-based multi-signatures. For example, in the case of a simple Schnorr multi-signature [5], $g^s \bmod p$ (or its hashed value) can be a PoV. Likewise, in the case of compact multi-signatures for smaller blockchains [6], [7], $g^s \bmod p$ (or its hashed value) can be a PoV.

One way of using the PoV is to concatenate it with the other inputs to the hash function in PoW (Fig. 1). Then the other blockchain nodes can verify whether the miner has really verified the signature. Another way is to simply append the PoV to the hashed value in the PoW (Fig. 2).

On receiving a new block reported by a miner, other blockchain nodes verify it. If one or more PoVs are incorrect, then the block reported by the miner is disqualified and discarded.

The order of concatenating (or appending) the PoVs of different transactions in the same block can be controlled in the same way as in the hash-tree aggregation of transactions. Thus the use of proposed PoVs would not cause significant overhead of such control mechanisms.

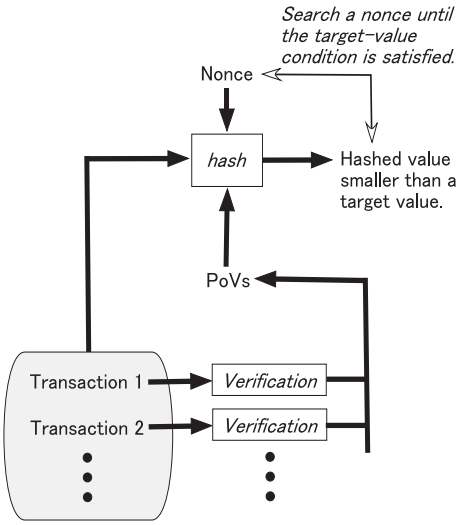

Figure 1: Concatenating Proofs-of-Verification with the inputs to the PoW hashing.

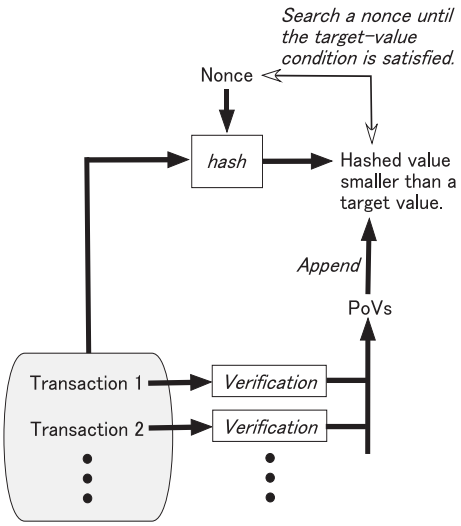

Figure 2: Appending Proofs-of-Verification to the PoW hashed value.

3 Discussion

If the PoVs are concatenated with the nonce, the other blockchain nodes can verify the validity of the nonce only if they themselves also verify the signatures.

If the PoVs are appended to the hashed value in the PoW, the other blockchain nodes can verify the validity of the nonce even if they skip the signature verification (and hence, the PoV verification as well); this would allow probabilistic (or somehow scheduled) skip for the purpose of reducing the computational workload of honest blockchain nodes. However, communication overhead arises due to the appended PoVs.

From the viewpoint of scalability and efficiency, such implementation options should be scrutinized in a realistic environment including mining pools. If this proposal is accepted, our presentation would include a report of some experiments and suggestions for the scrutiny. Finally, it should be noted that the same PoV mechanism can be considered for many of the PoW-based altcoins where DL-based (multi-)signatures can be used.

Acknowledgment

This work was partly supported by JSPS KAKENHI Grant Number JP17KT0081.

References

[1] C. P. Schnorr: Efficient signature generation by smart cards, *Journal of Cryptology*, Vol. 4, pp. 161–174, 1991.

[2] K. Matsuura, H. Imai: Modified aggressive modes of Internet Key Exchange resistant against Denial-of-Service attacks, *IEICE Transactions on Information and Systems*, Vol. E83-D, No. 5, pp. 972–979, May 2000.

[3] D. W. Kravitz: Digital signature algorithm, U. S. Patent # 5,231,668, July 1993.

[4] Y. Zheng, Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption), Proc. Crypto'97, LNCS 1294, pp. 165–179, Springer, August 1997.

[5] G. Maxwell, A. Poelstra, Y. Seurin and P. Wuille: Simple Schnorr multi-signatures with applications to Bitcoin, *Designs, Codes and Cryptography*, Vol. 87, Issue 9, pp. 2139–2164, September (online first in February) 2019.

[6] D. Boneh, M. Drijvers and G. Neven: Compact multi-signatures for smaller blockchains, Scaling Bitcoin 2018, October 2018.

[7] D. Boneh, M. Drijvers and G. Neven: Compact multi-signatures for smaller blockchains, Proc. Asiacrypt 2018, LNCS 11273, pp. 435–464, Springer, December 2018.