# Cryptographic Sortition for Proof of Stake in Bazo

Roman Blum [rblum@hsr.ch]
University of Applied Sciences Rapperswil

July 6, 2018

**Abstract**

Cryptographic sortition is used to randomly select a user who is assigned the right to append the next block to a blockchain. The goal of this paper is to analyze existing blockchain protocols based on Proof of Stake (PoS), such as Algorand and Ouroboros, in order to reevaluate and improve cryptographic sortition in Bazo.

Bazo is a peer-to-peer cryptocurrency which recently moved from Proof of Work (PoW) to chain-based PoS. However, the PoS protocol of Bazo has a vulnerability that is exploitable in the event of a fork where validators unintentionally reveal secret information.

## 1 Introduction

In blockchain protocols based on Proof of Work (PoW) such as Bitcoin [12], miners race to solve a difficult mathematical puzzle. The winner proposes the next block and earns a reward. Since it takes significant computing power to solve this problem, it proves that the miner has done sufficient work to solve the puzzle. However, PoW systems are highly energy-inefficient, since the only purpose of PoW is to reach consensus by solving a problem that is deliberately made hard such that an adversarial attempt to overpower the network is rendered economically infeasible [13].

The wastefulness, i.e., electricity and mining hardware, of PoW-based blockchains has motivated the industry and academia to explore alternative solutions. The challenge lies in the fact that an alternative solution requires

the same secure and decentralized properties like a PoW-based system, but without depleting physical scarce resources [9].

A solution to the problem was proposed in the early times of Bitcoin [17], referred to as Proof of Stake (PoS), where validators vote with their ownership of a certain cryptocurrency unit instead of computational power. The weight of each validator in the network depends on the size of of their stake. PoS seeks to address the computational overhead of PoW by attributing mining power in proportion to the amount of coins held by a miner; however, it faces a number of economical and technical challenges.

# 2 Background and Related Work

The term *cryptographic sortition* was introduced by Micali et al. [10] in 1999 and origins back to the ancient Athens, where sortition was a method of selecting a random sample from a pool of political officials. Equally, cryptographic sortition in PoS is used to select a random subset of users with respect to each users' stake.

The goal of this paper is to reevaluate the PoS design choices of the Bazo blockchain. Bazo [15, 16] is an open source cryptocurrency developed from scratch by students at the University of Zurich and University of Applied Sciences Rapperswil. Initially developed as a PoW-based blockchain, energy consumption considerations and the risk of centralization were reasons to recently switch from PoW to PoS.

Bazo uses a chain-based PoS approach, that is, the algorithm randomly selects a validator and assigns him or her the right to append a new block to the blockchain. In order to become a validator, a user generates a local seed, denoted as $Seed_{Local}$, and publishes a Stake Transaction (StakeTx, further described in Section 4.1) containing a hashed version of $Seed_{Local}$ to the network. A transaction of this type is accepted by the network if the issuer fulfills the minimum staking amount that is needed.

After the minimum waiting time elapsed, each validator participates in the election process in a non-interactive way. Contrary to PoW, a validator is limited to exactly 1 H/s (hash per second) by providing a valid *TimeInSeconds* to fulfill the PoS condition

$$\frac{\text{SHA-256}([Seed_{PrevBlocks}] \cdot Seed_{Local} \cdot BlockHeight \cdot TimeInSeconds)}{Coins} \leq Target. \tag{1}$$

The following description of the properties in Equation 1 is from the official Proof of Stake for Bazo paper [16] and has been slightly adapted:

**List of the Previous Seeds($Seed_{PrevBlocks}$)** Each block contains the $Seed_{Local}$ of the validator who created the block. By including a list of the previous seeds a "stake grinding" attack becomes infeasible, because the local seeds are kept private by other validators of the network.

**Local Seed($Seed_{Local}$)** All parameters in the Equation 1 are the same for each validator in the network except the local seed. The local seed individualises the PoS condition to each validator and therefore, a validator with a low amount of coins also has the possibility to append a block to the blockchain.

**Height of the Block($BlockHeight$)** The height of a block is the number of previously added blocks in the blockchain plus one.

**Amount of Coins($Coins$)** The amount of coins a leader owns. Note that the chances increase proportionally to the number of $Coins$.

**Difficulty of the PoS condition($Target$)** A validator has to fall below a *Target* in order to append the next block to the blockchain. Note that this target determines the speed of the blockchain. As the number of validators in-/decreases the difficulty must be adjusted accordingly.

With attention to the PoS condition, if a validator falls below the $Target$ value, he or she has the right to append the next block to the blockchain. In that case, the validator must publish the $Seed_{Local}$ that it used to fulfill the PoS condition. Additionally, a hashed version of a new $Seed_{Local}$ is also set in the new block.

However, the Bazo blockchain has a vulnerability in the event of a fork. Assume that an accidental fork happens, i.e., two validators fulfill the PoS condition at the same time. Consequently, the blockchain diverges into two potential chains forward, and as a result both chains could potentially grow

side-by-side for an indefinite amount of time. The fork is resolved when one chain becomes longer than the other, i.e., the block height differs, and hence, the network abandons the shorter chain resulting in orphaned blocks. Unfortunately, validators who worked on the shorter chain have published their local seed and would have to publish a new StakeTx with an updated $Seed_{Local}$ to the network. Thus, validators would need to pay fees for the transaction and wait until the minimum waiting time elapsed before they can start staking again. As a result, the chance of a stake grinding attack increases, as described in Section 2.2.

## 2.1 Related Work

There have been several proposals on how PoS could be implemented. The first cryptocurrency to implement PoS was Peercoin [18] in 2012, which uses a hybrid design to combine PoS and PoW methods. The PoW method is only used for the initial minting process, whereas the PoS method is used for the minting of new coins by using a concept called *coin age*. Coin age is simply described as accumulated coin-days a miner has held coins in his or her wallet which in turn increases the chance of meeting a specific hash target in order to append the next block to the blockchain, similar to the PoS condition in Bazo shown in Equation 1.

Bentov et al. [9] formally studied and analyzed existing PoS protocols in 2014, and offered novel construction of a so called *pure* PoS protocol that aims to overcome the problem of rational forks.

Ouroboros [8] was the first scientifically peer-reviewed PoS protocol. In Ouroboros, the physical time is divided into epochs, and each epoch is divided into slots. Each slot has one and only leader who has a sole right to produce a block during his slot. Slot leaders are elected by all stakeholders for the next epoch during the current epoch. The public blockchain and cryptocurrency project Cardano [14] employs Ouroboros in a slightly modified version. In parallel to the development of Ouroboros, Algorand [7] was developed.

Algorand is different in that stakeholders do not rely on some type of randomness, but rather self-election by running the election process on the local computer based on information in the previous block, while the selection runs random and completely automatic.

Meanwhile, Casper [5] is the PoS protocol that is going to replace the current PoW implementation of Ethereum. Casper introduces a new concept called *concensus by bet*. In Casper, anyone can participate by posting a

bond, i.e., putting money on stake. After posting a bond, a validator has the opportunity to bet on a block that will be likely processed and appended to the blockchain. The idea is that validators are incentivized to play by the rules, i.e., being rewarded by betting on eventual consensus or losing money by betting against consensus. Hence, each validator's incentive is to bet in the way that they expect others to bet in the future, driving the process toward convergence [6]. The concept of Casper is analogous to PoW where each miner is betting with their hash power by spending money on electricity and mining equipment erosion.

| | **Release** | **Live** | **PoS Mechanism** |
|---|---|---|---|
| Algorand | 2017 | No | Byzantine Fault Tolerance (BFT) |
| Bazo | 2018 | Yes | Chain-based PoS |
| Casper FFG | 2017 | Testnet only (Ethereum) | Hybrid PoS/PoW as Smart Contract on top of Ethereum |
| Ouroboros | 2017 | Yes (Cardano) | Slot Leader Election and Follow The Satoshi (FTS) algorithm |
| Peercoin | 2012 | Yes (PPCoin) | PoW for coin distribution, PoS for consensus |

Table 1: An overview of the analyzed PoS-based blockchain protocols

## 2.2 Challenges

Fundamentally, the problem of PoS-based blockchains is the leader election process. True randomness is crucial to achieve a fair election among stakeholders, otherwise the algorithm may be prone to manipulation by an adversary [8]. For instance, the adversary could perform computations to try to bias the randomness in their own favor in order to increase the probability of their coins generating a valid block. This class of attack is referred to as "stake grinding", since the adversary could "grind" through many combinations and find favorable parameters to bias the leader election [5]. However, generating bias-resistant distributed randomness in a decentralized trust-less network is considered hard [1], because introducing entropy into the system is often difficult and error-prone [2–4].

# 3 Design

The basic design concept of the PoS protocol employed in the current implementation of Bazo blockchain is kept similar, that is, a validator wishing to participate in the staking process has to publish a StakeTx well in advance in order to join the set of validators, has to fulfill a PoS condition to determine if he or she is eligible to append a block to the blockchain, and a block must contain some kind of information that proves the eligibility of the creator. However, a validator's exposed information should not be abused when a block is considered orphaned in the event of a fork. To put it differently, a validator's secret information is used to generate publicly verifiable information to prove that the validator is eligible to append block at height $h$. This publicly verifiable information must be only valid for height $h$, but invalid for height lower or higher than $h$.

We present two algorithms suitable for the Bazo blockchain in order to process the cryptographic sortition, namely using the concept of Verifiable Random Functions (VRFs) and using the public-key cryptosystem RSA. Both approaches are deterministic (as opposed to probabilistic), that is, for a given plaintext input and secret key it always produces the same cipher text output. It is important to realize that a deterministic encryption scheme is crucial for our design, because otherwise a stake grinding attack may again become feasible with computational effort, i.e., an adversary can produce multiple outputs for the same digital signature.

## 3.1 Sortition using VRFs

VRFs [10,11] are simply described as follows: Given a value $x$ and a key-pair with a public key $pk$ and a private key $sk$, a user can compute

$$\beta = VRF_{hash}(sk, x) \tag{2}$$

and

$$\pi = VRF_{proof}(sk, x). \tag{3}$$

Both the hash $\beta$ and the proof $\pi$ are uniquely determined by $sk$ and $x$. Using $\pi$, everyone can reproduce $\beta$ without having the user to reveal $sk$ as

$$\beta = VRF_{proof2hash}(\pi). \tag{4}$$

Note that this means that

$$VRF_{hash}(sk, x) = VRF_{proof2hash}(VRF_{proof}(sk, x)). \tag{5}$$

Furthermore, using $\pi$ and $pk$, everyone can verify the correctness of $\beta$ with

$$r = VRF_{verify}(pk, \pi, x), \tag{6}$$

where $r$ is either $true$ or $false$. The $VRF_{hash}$ algorithm is deterministic, i.e., it always produces the same output $\beta$ for a given input $x$ and $sk$.

In Algorand, cryptographic sortition is implemented using VRFs for choosing a random subset of users according to per-user weights [7]. With this in mind, VRFs can be used for cryptographic sortition in Bazo. The procedure of PoS introduced in Section 2 works as follows.

A node publishes an empty stake transaction that states whether it wants to join (or leave) the set of validators by setting the $IsStaking$ property to $true$ or $false$. After a node has joined the set and the minimum waiting time has passed, it is eligible to append blocks to the blockchain by fulfilling the PoS condition

$$\frac{\text{SHA-256}([\beta_{PrevBlocks}] \cdot \beta_{Local} \cdot BlockHeight \cdot TimeInSeconds)}{Coins} \leq Target, \tag{7}$$

where

$$\beta_{Local} = VRF_{hash}(sk, BlockHeight) \tag{8}$$

is the local proof of the validator with a private key $sk$, where $BlockHeight$ is the number of previously added blocks in the blockchain plus one, and where $[\beta_{PrevBlocks}]$ is a list of the previous proofs. Note that $BlockHeight$ is used as a parameter to calculate $\beta_{Local}$ for two reasons. First, the block height is publicly known and can be used to verify as shown in Equation 10, and second the digest of $VRF_{hash}$ is only valid for the height the block was published. A published block of a validator contains both $\beta_{Local}$ as well as $\pi_{Local}$. Other validators are now able to verify the eligibility by using $\pi_{Local}$, $\beta_{Local}$ and the public key $pk$ of the validator who fulfilled the Equation 7 and appended the next block with

$$\beta_{Local} = VRF_{proof2hash}(\pi_{Local}) \tag{9}$$

and

$$r = VRF_{verify}(pk, \pi_{Local}, BlockHeight). \tag{10}$$

All signatures in Bazo are based on the elliptic curve digital signature algorithm (ECDSA), a digital signature algorithm based on elliptic curve cryptography (ECC). VRFs can be implemented using ECC or RSA. Thus, the implementation of VRFs has the advantage that the already existing public-private key-pair for signing transactions can be reused for this design approach. Furthermore, $\beta_{Local}$ and $\pi_{Local}$ have a small footprint in a published block, i.e., in the current implementation of Bazo which uses the ECDSA curve P-256, each value is only 32 bytes in size.

## 3.2 Sortition using RSA

RSA is an asymmetric encryption and decryption scheme that consists of a public-private key-pair $(pk, sk)$. Anyone can use $pk$ to encrypt a message $m$ to obtain the cipher text

$$c = RSA_{encrypt}(pk, m), \tag{11}$$

but only the owner of $sk$ can decrypt the message

$$m = RSA_{decrypt}(sk, c). \tag{12}$$

Additionally, RSA can also be used to apply a digital signature to a message. A user in possession of $sk$ can send $m$ along with a signature

$$s = RSA_{sign}(sk, Hash(m)), \tag{13}$$

and others can verify that $m$ indeed origins from the user by comparing $Hash(m)$ with the output of

$$RSA_{verify}(pk, s, m). \tag{14}$$

Note that the $RSA_{sign}$ function can't accommodate messages that are longer than $sk$. Hence, the hash function $Hash()$ is used to minimize the message length.

Similar to VRFs, RSA can be used as cryptographic sortition. Due to the simplicity of this approach and the built-in Golang RSA implementation (package "crypto/rsa") which leads to an easier implementation, RSA will be used as the cryptographic sortition mechanism in Bazo.

# 4 Implementation

We provide an updated description of the protocol design as proposed in the official Proof of Stake for Bazo paper [16]. Descriptions are taken, shortened and revised where necessary.

## 4.1 Stake Transaction (StakeTx)

A node wishing to participate in staking can publish a stake transaction. The transaction contains the property *Key Commitment*, which is the RSA public key $pk$ of the node who published the transaction. The StakeTx consists of the following properties:

**Fee** A fee that has to be paid for the validator.

**Is Staking** A boolean value that indicates whether the node wants to join or leave the set of validators.

**Account** The hash of the public key of the issuer.

**Key Commitment** When a node wants to join the set of validators it must create a pair of public and private key $(pk, sk)$ using RSA. Note that this key-pair is different to the key-pair of the user's wallet. By submitting the public key $pk$, the node commits to this key and it cannot change it in a later step. The $(pk, sk)$ key-pair will be used in the PoS condition which is described in detail in Section 4.3.

**Signature** The signature serves the purpose of authentication. The node digitally signs the transaction with its private key.

## 4.2 Block

A block consists of the following properties:

**Number of StakeTx** The number of StakeTxs that are included in the block.

**StakeTx Data** The hashes of all StakeTxs included in this block in sequential order.

**Time in Seconds** The number of seconds that a validator needs in order to fulfill the PoS condition.

**Commitment Proof** This property stores a signed message of the $Height$ that this block was created. In particular, $RSA(sk, \text{SHA3-512}(Height))$ where $sk$ represents the private key that corresponds to the public key $pk$ that was set in the initial StakeTx of the node. Other validators can use $pk$ in order to verify the proof.

**Height** The height of a block refers to the number of previously appended blocks to the blockchain.

**Slashed Address** A validator can submit a slashing proof when appending a block, i.e., it holds the address of the misbehaving node that must be punished.

**Two Conflicting Block Hashes** These two properties exhibit the block hashes where the same node has appended a block on two competing chains within the slashing window size.

## 4.3 PoS Condition

The introduced PoS condition in Section 2 is updated to

$$\frac{\text{SHA-256}([Proof_{PrevBlocks}] \cdot Proof_{Local} \cdot BlockHeight \cdot TimeInSeconds)}{Coins} \leq Target. \tag{15}$$

The following parameters changed as opposed to Equation 1:

**List of the Previous Proofs ($Proof_{PrevBlocks}$)** By including a list of the previous proofs a stake grinding attack becomes infeasible.

**Local Proof ($Proof_{Local}$)** All parameters in Equation 15 are the same for each validator in the network except the local proof. The local proof individualises the PoS condition to each validator and therefore, a validator with a low amount of coins also has the possibility to append a block to the blockchain.

### 4.4 Validation

Upon receiving a new block, each validator can check if the *Commitment Proof* provided in the block corresponds to the public key *pk* set in the StakeTx when the validator has joined the set of validators. In particular, each validator can verify the origin of the message by the public key of the block creator. Other conditions such as minimum waiting time or PoS condition can be studied in the official Proof of Stake for Bazo paper [16].

## 5 Evaluation

Blockchain always raises a variety of pressing security considerations, and each change request could potentially break the entire system and puts the blockchain and its users at risk.

The usual attack on RSA involves factoring a large number which is the product of two very large prime numbers. In general, finding prime numbers is fairly easy, and multiplying them together to get a single large number is also quite easy. However, starting with a large prime number, finding the factors is *quite* hard. Implemented correctly, i.e., using a profound Go library that does the encryption part, RSA is considered safe. Other attack vectors and countermeasures have been extensively discussed in the official Proof of Stake for Bazo paper [16].

The Bazo blockchain strives for simple, yet secure implementations of fundamental concepts in computer science. This mindset led to decision of using RSA over VRFs. The updated protocol design approach uses RSA to proof the eligibility of a validator who wants to append a block to the blockchain. Thus, the key length is an important security parameter and should be chosen appropriately. According to the National Institute of Standards and Technology (NIST), a key length of 2048 bits is considered secure for the years to come until 2030 [19].

However, using RSA over VRFs has two disadvantages. First, besides the public-private key-pair for signing transactions, a user has to maintain a second pair of keys. Second, assuming that an RSA key size of 4096 bits is chosen, the footprint of the *Commitment Proof* property in a block is 8 times higher in comparison to the VRF values $\beta$ and $\pi$ each having a size of 256 bits.

| | **VRFs** | **RSA** (4096 bits key size) |
|---|---|---|
| Additional key-pair required? | No, key-pair of wallet can be used | Yes, a second key-pair must be maintained |
| Additional Size in StakeTx | 0 Bytes | 512 Bytes |
| Size in Block | $2 \times 32$ Bytes | 512 Bytes |
| Built-in Golang | No, 3rd party package required | Yes, "crypto/rsa" package |

Table 2: An overview of cryptographic sortition using VRFs vs. RSA

# 6 Conclusions

Upon validating and extensively discussing our design approach, we decided to choose sortition using RSA due to simplicity and built-in Golang RSA implementation. The transaction size of RSA proofs is an acceptable disadvantage. The implemented solution is unique and specially built for the PoS protocol of the Bazo blockchain.

# Acknowledgements

Many thanks to Markus Knecht for his helpful comments and suggestions regarding all cryptographic matter of this paper.

# References

[1] E. Syta and P. Jovanovic and E. Kokoris Kogias and N. Gailly and L. Gasser and I. Khoffi and M. J. Fischer and B. Ford. Scalable Bias-Resistant Distributed Randomness. `https://eprint.iacr.org/2016/1067`, 2016.

[2] R. Chirgwin. iOS 7s weak random number generator stuns kernel security. The Register, `https://www.theregister.co.uk/2014/03/16/ios_7_has_weak_random_number_generator/`, 2014.

[3] Z. Gutterman, B. Pinkas, and T. Reinman. Analysis of the Linux random number generator. In *IEEE Symposium on Security and Privacy*, pages 371385, 2006.

[4] J. Kelsey, B. Schneier, D. Wagner, and C. Hall. Cryptanalytic Attacks on Pseudorandom Number Generators. *Fast Software Encryption, Fifth International Workshop Proceedings (March 1998)*, Springer-Verlag, pp. 168-188, 1998.

[5] V. Buterin et al. Proof of Stake FAQ. `https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ`, 2016.

[6] V. Zamfir. Introducing Casper "the Friendly Ghost". `https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/`, 2015.

[7] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. *p51–68*. In *Proceedings of the 26th Symposium on Operating Systems Principles*, Shanghai, China, 2017. `http://doi.acm.org/10.1145/3132747.3132757`.

[8] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. `https://eprint.iacr.org/2016/889.pdf`, 2017.

[9] I. Bentov, A. Gabizon, and A. Mizrahi. Cryptocurrencies without Proof of Work. `https://dblp.org/rec/bib/journals/corr/BentovGM14`, 2014.

[10] S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, New York, NY, Oct. 1999.

[11] S. Goldberg, D. Papadopoulos, and J.Vcelak. Verifiable Random Functions (VRFs). Internet-Draft of the Internet Engineering Task Force (IETF) `https://tools.ietf.org/id/draft-goldbe-vrf-01.html`, 2017.

[12] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008

[13] Bitcoin Wiki. Majority Attack. `https://en.bitcoin.it/wiki/Majority_attack`, visited 30.03.2018.

[14] Cardano. A decentralised public blockchain and cryptocurrency project. `https://www.cardano.org/en/home/`, visited 21.04.2018.

[15] L. Sgier. Bazo - A Cryptocurrency from Scratch. https://files.ifi.uzh.ch/CSG/staff/bocek/extern/theses/BA-Livio-Sgier.pdf, 2017.

[16] S. Bachmann. Proof of Stake for Bazo. https://files.ifi.uzh.ch/CSG/staff/bocek/extern/theses/BA-Simon-Bachmann.pdf, 2018

[17] Bitcoin forum user "QuantumMechanic". Proof of stake instead of proof of work. https://bitcointalk.org/index.php?topic=27787.0, visited 21.03.2018.

[18] S.King and S. Nadal. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. https://peercoin.net/assets/paper/peercoin-paper.pdf, 2012.

[19] D. Giry. Cryptographic Key Length Recommendation, https://www.keylength.com/en/4/, visited 30.04.2018.