Proof-of-Stake

Research Seminar in Cryptography

Janno Siim

Project advisor: Michał Zając

1 Introduction

Blockchain is a decentralized database containing linked data blocks that are resistant to modifications. Blockchain was first introduced for constructing Bitcoin cryptocurrency in [Nak09]. After that, a lot of research effort has been put to studying blockchain and many alternative constructions have been proposed and implemented.

Blockchain protocol contains a number of *miners* that collect users' data, called *transactions*, and store that data in discrete chunks, called *blocks*. Blocks are linked together by hashing to form a blockchain (see Fig. 1 for blockchain of Ouroboros system). Miners always extend the longest available valid chain.

The key component in Bitcoin protocol (and many of the follow-up protocols) is the *proof-of-work* (PoW) puzzle solving. A miner can issue a new block only if it has solved a computationally difficult PoW challenge. In case of Bitcoin, miners need to find a hash of the previous block (by appending small *nonce* to it) that has a certain number of zeros in the beginning. The miner that finds it first can issue the next block and her work is rewarded in cryptocurrency.

Reward system and common interest of holding up the value of the currency should keep the majority of miners behaving honestly. This in turn implies that once a block $\mathbf{B}$ is deep enough in the chain, then it will be computationally difficult to construct a longer valid chain which does not contain $\mathbf{B}$. Security of Bitcoin has been studied in depth in [GKL15].

PoW comes together with an enormous energy demand. Current estimate is that Bitcoin protocol's energy needs are comparable to energy consumption of Ireland [OM14]. Increase in usage will only make the situation worse.

This has made blockchain enthusiasts and researchers think of alternatives for PoW. From an abstract view, there are two properties that make a blockchain work:

- There is a **randomized *leader* election process** for issuing each block. In PoW, the miner who happens to find the solution for the PoW challenge fastest, is elected to be the leader and hence has the right to issue the next block. Note that more computational power a miner has, the more likely she is to be elected.
- There is an **incentive structure** that keeps miners behaving honestly and extending the blockchain. In Bitcoin this is achieved by miners getting a block discovery reward and transaction fees from users. Additionally, subverting the protocol would reduce the trust (and possibly usability) of the currency, which in turn would reduce the price. Hence, a miner would end up undermining the currency that she itself collects.

Based on those two charateristics, several alternatives to proof-of-work have been proposed:

Proof-of-Stake (PoS) The leader is elected with a probability proportional to the amount of *stake* it owns in the system. In the simplest case, stake is the amount of currency, but it can also be (for example) the age of the coin that a miner holds.

Proof-of-Burn (PoB) Miner's election chances are proportional to the amount of currency she destroys i.e. "burns". Essentially it models buying CPU power for proof-of-work but the actual computation never happens. In comparison, PoS allows a miner to "have her cake and eat it too". [1]

Proof-of-Space Similar to proof-of-work, but instead computational power, miners invest disk space [Fuc15].

Most well-studied of those is proof-of-stake which is also the focus of this report. In section 2 we give a general overview of proof-of-stake protocols, section 3 gives some mathematical background, and in section 4 we give a more detailed look of one particular proof-of-stake protocol – Ouroboros [KRDO17].

2 Proof-of-Stake

As mentioned in the previous section, proof-of-stake is one of the main candidates to solve the energy demand problem in the current blockchain protocols such as Bitcoin and Ethereum.

To best of our knowledge, idea for proof-of-stake originates from the user Quantummechanic in bitcointalk[2] forum:

[1] It is not obvious if destroying coins makes PoB better or worse than PoS. Burning coins might help to avoid the attacks in PoS related to transfering stake from one account to another since burnt coins are bound to a concrete account. However, like PoW mining, PoB mining still needs some type of useless energy loss (to obtain stake you need money which in turn is earned by energy consumption in real life).

[2] For full post see `https://bitcointalk.org/index.php?topic=27787.0`

> *... I'm wondering if as bitcoins become more widely distributed, whether a transition from a proof of work based system to a proof of stake one might happen. What I mean by proof of stake is that instead of your "vote" on the accepted transaction history being weighted by the share of computing resources you bring to the network, it's weighted by the number of bitcoins you can prove you own, using your private keys. ...*

We discuss concrete PoS blockchain protocols later, but roughly all of them use the following strategy. Each party has a certain amount of stake in a blockchain (typically the amount of cryptocurrency). For each block there is a randomized leader election process; the party that gets elected can issue the next block. More stake a party has, more likely it is to get elected as a leader. Similarly to PoW, block issuing is rewarded (for example by collecting transaction fees from parties that want their data to be included).

Although this approach reduces the energy demand, also new issues arise that were not present in PoW-based blockchains.

Grinding attack Malicious parties can try to bias the election process in their favour. This could allow them to collect more rewards than their actual fair share or they could more easily double spend their money.

Nothing at stake attack In PoS it is much easier to construct alternative chains. In PoW, a party would lose CPU time by working on an alternative chain, whereas in proof-of-stake, party seemingly does not lose anything by also mining on an alternative chain.

A number of proof-of-stake protocols have been implemented: PIVX, Peercoin, Blackcoin and many more. In addition, second highest valued cryptocurrency – Ethereum – plans to move from a PoW blockchain to a PoS blockchain [BG17]. These proposals mostly lack in proper security analysis (Ethereum might be an exception here). This seems important, especially in the light of recently discovered vulnerability in IOTA[3] (one of the highest valued cryptocurrencies at the moment of writing) that used a self-designed hash-function which turned out not to be collision resistant.

Some constructions do have a more rigorous cryptographic analysis [BPS16, GHM+17, PS17, KRDO17, DGKR17]. For no particular reason, we focus on Ouroboros [KRDO17] in this report.

3 Preliminaries

By $\{0,1\}^*$ we denote the set of all finite length binary vectors. Security parameter is denoted by λ. We write $f(\lambda) = \mathsf{negl}(\lambda)$, if function f is negligible in λ.

[3] For attack details see `https://github.com/mit-dci/tangled-curl/blob/master/vuln-iota.md`

If element d is uniformly randomly picked from set $\mathcal{D}$, then we write $d \leftarrow_r \mathcal{D}$. By $\Pr[d \leftarrow \mathcal{D} : \mathcal{P}(d)]$ we denote the probability of predicate $\mathcal{P}$ holding over the distribution $\mathcal{D}$. PPT stands for probabilistic polynomial time.

Digital Signature. Digital signature scheme is a tuple of PPT algorithms $\Sigma = (\mathsf{KGen}, \mathsf{DS}, \mathsf{Ver})$, such that

– $\mathsf{KGen}(1^\lambda)$ outputs a key pair $(\mathsf{vk}, \mathsf{sk})$, where vk is a public verification key and sk is a private signing key.

– $\mathsf{DS}_{\mathsf{sk}}(m)$ takes in a message m and outputs a signature σ under the secret key sk.

– $\mathsf{Ver}_{\mathsf{vk}}(m, \sigma)$ outputs 1 if σ is a valid signature of message m under the verification key vk, and otherwise outputs 0.

Informally, we say that signature scheme Σ has *the existential unforgeability against chosen message attacks (EUF-CMA)* , if a PPT adversary cannot create a new valid signature even after seeing polynomially many valid signatures.

Hash Function. Hash-function family is a map $\mathcal{H} : \mathcal{K} \times \mathcal{M} \to \{0,1\}^\lambda$. Here, $\mathcal{K}$ is the key space and $\mathcal{M}$ is the message space. Each key $k \in \mathcal{K}$ defines a hash function of this family.
Definition 1. *We say that a hash-function family $\mathcal{H} : \mathcal{K} \times \mathcal{M} \to \{0,1\}^\lambda$ is collision-resistant if for any PPT adversary $\mathcal{A}$,*

$$\Pr[k \leftarrow_r \mathcal{K}, (m_1, m_2) \leftarrow \mathcal{A}(k) : \mathcal{H}(k, m_1) = \mathcal{H}(k, m_2)] = \mathsf{negl}(\lambda).$$

We denote a fixed hash function as $h_k(m) := \mathcal{H}(k, m)$. Moreover, for convenience we leave out the index and just write $h(m)$ in the rest of this report.

Commitment Scheme. Commitment scheme is a tuple of efficient algorithms $(\mathsf{KGen}, \mathsf{Com}, \mathsf{Ver})$ such that

– $\mathsf{KGen}(1^\lambda)$ outputs a commitment key pk.

– $\mathsf{Com}_{pk}(m)$ outputs a commitment c and an *opening* d for the message m.

– $\mathsf{Ver}_{\mathsf{pk}}(c, d)$ checks that d is a valid opening for the commitment c.

Typically a commitment scheme should satisfy at least the following two properties. Informally, we say that a commitment scheme is *(perfectly) hiding*, if $\mathsf{Com}_{\mathsf{pk}}(m)$ does not reveal any information about the message m. Commitment scheme is *(computationally) binding*, if no PPT adversary can find c, d_1, and d_2 such that $\mathsf{Ver}_{\mathsf{pk}}(c, d_1) = \mathsf{Ver}_{\mathsf{pk}}(c, d_2) = 1$ and $d_1 \neq d_2$.

Publicly Verifiable Secret Sharing (PVSS) Scheme. In a (t, k)-PVSS scheme [Sta96], a dealer $\mathcal{D}$ wishes to share a secret x between users $U_1, \ldots, U_k$ such that at least $t+1 \leqslant k$ parties need to collaborate to recover x. Additionally, PVSS is publicly verifiable in two different sense. Firstly, it is possible to verify that dealer provided correct shares to the parties, and secondly, it is possible to verify in the recovery phase that party outputted the correct share.

To obtain public verifiability, each user U_i generates and publishes encryption key pk_i of a public key cryptosystem. Then, instead of dealer sending shares directly to parties (as is done in regular secret sharing schemes), dealer publishes encryptions of shares that users can decrypt to obtain their share.

In more detail, (t, k)-PVSS scheme is a tuple of efficient algorithms (Deal, $\mathsf{Recover}$, $\mathsf{Pro}, \mathsf{Ver}_\mathsf{D}, \mathsf{Ver}_\mathsf{R}$) such that

– $\mathsf{Deal}(x, \{pk_i\}_i)$ outputs ciphertexts $(c_1, \ldots, c_k)$ under respective public keys and a proof $\pi_\mathcal{D}$.

– $\mathsf{Ver}_\mathsf{D}(\{pk_i\}_i, \{c_i\}_i), \pi_\mathcal{D})$ outputs 1 if proof $\pi_\mathcal{D}$ is correct and 0 otherwise.

– $\mathsf{Recover}(\{x_i\}_{i \in S})$ outputs x, givent that $S \subseteq \{1, \ldots, n\}$ and $|S| \geqslant t + 1$.

– $\mathsf{Pro}(x_i, \mathsf{pk}_i, c_i)$ outputs a proof $\pi_{\mathcal{R}_i}$.

– $\mathsf{Ver}_\mathsf{R}(x_i, c_i, \pi_{\mathcal{R}_i})$ outputs 1 if proof $\pi_{\mathcal{R}_i}$ is valid and 0 otherwise.

Informally, we require the following security properties (i) no information about x can be recovered with less than $t+1$ shares, (ii) if $\mathsf{Ver}_\mathsf{D}(\{pk_i\}_i, \{c_i\}_i), \pi_\mathcal{D}) = 1$, then sharing was done correctly, (iii) if $\mathsf{Ver}_\mathsf{R}(x_i, c_i, \pi_{\mathcal{R}_i}) = 1$, then c_i encrypts the message x_i.

Robust Transaction Ledger. We avoid a completely formal and general definition of a blockchain (next section has a detailed description for a concrete protocol). Let us take the intuitive view developed in the previous sections and try to understand what it means for a blockchain protocol to be secure.

For this purpose Garay et al. propose in [GKL15] a definition of a *robust transaction ledger*. Informally, blockchain protocol is a robust transaction ledger if it satisfies the following properties:

- *Persistence* – If an honest party has a transaction tx in a block that is k blocks deep, then all honest parties have tx in the same position in their local chain.
- *Liveness* – if all honest nodes attempt to include a transaction tx, then after at most u slots, honest nodes declare tx as stable.

Here, k (depth parameter) and u (wait time) behave as additional security parameters, bigger k or u, more likely it is that the corresponding property holds.

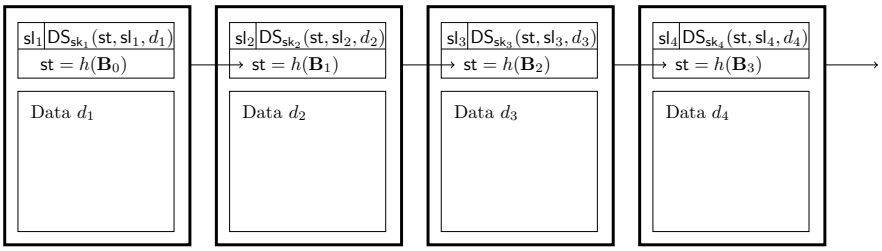

Figure 1: Blockchain of Ouroboros. Each block contains a slot number sl_i, a hash st of the previous block, transaction data d_i, and the slot leader's signature on the content.

4 Ouroboros

The protocol is run by n stakeholders $U_1, \ldots, U_n$. Each stakeholder U_i has stake s_i and a digital signature key pair $(\mathsf{vk}_i, \mathsf{sk}_i)$, out of which vk_i is public to everyone.

We divide time into equal length *slots* of length Δ and assume that message of an honest party is always delivered in time Δ (i.e. synchronous communication model). We index slots by consecutive natural numbers and denote them by $\mathsf{sl}_0, \mathsf{sl}_1, \mathsf{sl}_2, ...$, starting from the first slot. Counting from sl_1, each R consecutive blocks form an *epoch*.

A *block* $\mathbf{B}$ *issued by stakeholder* U_i is a tuple $(\mathsf{st}, \mathsf{sl}_i, d, \mathsf{DS}_{\mathsf{sk}_i}(\mathsf{st}, \mathsf{sl}_i, d))$, where

– $\mathsf{st} \in \{0,1\}^\lambda$ is the current state of blockchain,

– $\mathsf{sl}_i \in \mathbb{N}$ is the slot number,

– $d \in \{0,1\}^*$ is the transaction data.

Only the *genesis block* $\mathbf{B}_0$ at slot sl_0 has a different structure. Genesis block is a tuple $(\mathsf{sl}_0, \rho, (\mathsf{vk}_1, s_1), \ldots, (\mathsf{vk}_n, s_n))$, where $\rho \in \{0,1\}^\kappa$ is uniformly random bitstring (See Fig. 1).

The leader of each slot is randomly elected based on the amount of the stake. Challenge is to construct a protocol that picks the leader randomly, but stakeholders still agree that process was done correctly. This is done in two steps.

First, we run a multi-party coin tossing protocol to generate a random bitstring ρ. This is run among leaders of the previous epoch. Coin tossing protocol guarantees that if majority of parties are honest, then ρ is uniformly random.

Secondly, we apply a deterministic function F that given uniformly random ρ, a vector of stakes $(s_i)_{i=1}^n$, and sl_j outputs a stakeholder U_k (roughly) with the probability $s_k / \sum_{i=1}^n s_i$. Since F is deterministic, each stakeholder can check that $\mathsf{F}(\sigma, s_1, \ldots, s_n, \mathsf{sl}_j) = U_k$.

Constructing function F is relatively straightforward. We just need to guarantee that for any $k \in \{1, \ldots, n\}$, proportionally $s_k / \sum_{i=1}^n s_i$ of bitstrings in $\{0,1\}^\kappa$ are mapped to U_k. So if $\sum_{i=1}^n s_i = 2^\kappa$, we could simply output stakeholder U_k with the biggest index k such that $\sum_{i=1}^k s_i \leqslant \rho$. If the total stake is not precisely a power of 2, then some care needs to be taken to avoid bias.

Little bit more challenging is the multi-party coin tossing protocol and this we discuss in the following.

4.1 Multi-Party Coin Tossing

Two-party Coin Tossing. Let us first remind the classic two-party coin tossing [Blu83]. Alice and Bob want to fairly flip a coin over the telephone but neither trusts one another. So, Alice cannot just flip a coin and tell it to Bob or vice versa. It also does not work if both Alice and Bob flip a coin and then combine the result by some operation (for example by XORing the bits). One party will always learn about the flip of the other party first and thus has the opportunity to pick its coin in a way that gives a biased outcome.

First, instead of flipping a single coin, let us generate uniformly random bitstring instead (i.e. we want to toss many coins at once). Two-party coin tossing can be easily done with a commitment scheme. Alice samples a bitstring $\rho_A \leftarrow_r \{0,1\}^n$ and sends $\mathsf{Com}_{\mathsf{pk}}(\rho_A)$ to Bob. Bob samples a bitstring $\rho_B \leftarrow_r \{0,1\}^n$ and sends ρ_B to Alice. Finally, Alice sends ρ_A to Bob and both parties compute $\rho = \rho_A \oplus \rho_B$.

Intuitively, since ρ_A and ρ_B are picked independently, then ρ is uniformly random given that at least one of the two parties is honest. Note however that Alice has the opportunity to learn ρ even when she does not respond with ρ_A. In some applications this might cause problems.

Multi-Party Coin Tossing. We now consider the coin tossing protocol proposed for Ouroboros. Let us set the epoch length to $R = 10k$ slots. Let $U_1, \ldots, U_R$ be the parties participating in the coin tossing protocol. They have respective cryptosystem keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_R$. In the beginning of each epoch, parties run the protocol in Fig. 2.

4.2 Analysis

Very briefly, we discuss the security analysis of the Ouroboros protocol. They use a novel combinatorial proof technique that we will not cover. Even a proper statement of the main security theorem is not possible with the amount theory covered in this report.

In short, they prove that Ouroboros protocol is a robust transaction ledger (with high probability) with a depth parameter k and the wait time $u = 2k$. However, this comes with a number of assumptions, e.g., synchronous communication (as discussed before), corrpution queries of the adversary have a delay, amount of stake adversary can shift during a single epoch is bounded. As of recent, a new version of this protocol has been proposed [DGKR17], Ouroboros Praos, that reduces some of those assumptions.

Commitment Phase. Each party U_i:

1. Samples $\rho_i \leftarrow_r \{0,1\}^{R \cdot \log \tau}$.
2. Computes shares $(\rho_i^{(1)}, \ldots, \rho_i^{(R)}) \leftarrow \mathsf{Deal}(R, \rho_i)$.
3. Posts to blockchain $\mathsf{Enc}_{\mathsf{pk}_1}(\rho_i^{(1)}), \ldots, \mathsf{Enc}_{\mathsf{pk}_R}(\rho_i^{(R)}), C_i = \mathsf{Com}_{pk}(\rho_i)$.

This phase lasts for $4k$ slots.

Reveal Phase. Each party U_i:

1. Aborts if blockchain contains valid (encryptions of) shares from less than majority.
2. Posts an opening of his commitment C_i.

This lasts for $4k$ slots.

Recovery Phase.

1. For any party U_j that did not open its commitment parties send shares $\rho_j^{(i)}$ to blockchain. This allows recovery of ρ_j.
2. Each party can compute $\rho = \sum_{i=1}^{R} \rho_i$.

This phase lasts for $2k$ slots.

Figure 2: Multi-party coin tossing protocol.

References

[BG17] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv*, abs/1710.09437, 2017. 2

[Blu83] Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *SIGACT News*, 15(1):23–27, January 1983. 4.1

[BPS16] Iddo Bentov, Rafael Pass, and Elaine Shi. Snow white: Provably secure proofs of stake. *IACR Cryptology ePrint Archive*, 2016:919, 2016. 2

[DGKR17] Bernardo Machado David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake protocol. *IACR Cryptology ePrint Archive*, 2017:573, 2017. 2, 4.2

[Fuc15] Georg Fuchsbauer. Spacemint: A cryptocurrency based on proofs of space. *IACR Cryptology ePrint Archive*, 2015:528, 2015. 1

[GHM+17] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. *IACR Cryptology ePrint Archive*, 2017:454, 2017. 2

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. *The Bitcoin Backbone Protocol: Analysis and Applications*, pages 281–310. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. 1, 3

[KRDO17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. *Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol*, pages 357–388. Springer International Publishing, Cham, 2017. 1, 2

[Nak09] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf, 2009. 1

[OM14] K. J. O'Dwyer and D. Malone. Bitcoin mining and its energy footprint. In *25th IET Irish Signals Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014)*, pages 280–285, June 2014. 1

[PS17] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, PODC '17, pages 315–324, New York, NY, USA, 2017. ACM. 2

[Sta96] Markus Stadler. *Publicly Verifiable Secret Sharing*, pages 190–199. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996. 3