The Dusk Network Whitepaper

Toghrul Maharramov
Dusk Network
toghrul@dusk.network

Dmitry Khovratovich
Dusk Network
dmitry@dusk.network

Emanuele Francioni
Dusk Network
emanuele@dusk.network

Fulvio Venturelli
Dusk Network
fulvio@dusk.network

August 30, 2019
Version 2.0.0

Abstract

Dusk Network protocol is a blockchain-based distributed ledger secured via a novel state machine replication algorithm, enabling a permission-less participation in the process of state transition validation while simultaneously providing strong guarantees about the finality of the state transitions. The protocol is built to preserve user anonymity on both the transaction and state layers.

Contents

1 Introduction **3**

2 Our Contributions **4**

3 Abstract Definition of the Protocol **4**

4 Cryptographic Primitives **6**
- 4.1 Hash Functions . . . 7
  - 4.1.1 SHA-3 . . . 7
  - 4.1.2 Poseidon . . . 7
- 4.2 Elliptic Curves . . . 8
  - 4.2.1 Ristretto . . . 8
  - 4.2.2 Zerocaf . . . 9
- 4.3 Signature Schemes . . . 9
  - 4.3.1 EdDSA . . . 9
  - 4.3.2 bLSAG/MLSAG . . . 10
  - 4.3.3 BLS . . . 11
- 4.4 Zero-knowledge Proofs . . . 11
  - 4.4.1 Commitment Scheme . . . 12
  - 4.4.2 Bulletproofs . . . 13

5 Reaching Consensus **14**
- 5.1 Definitions . . . 14
- 5.2 Segregated Byzantine Agreement . . . 17
- 5.3 Proof-of-Blind Bid . . . 18
- 5.4 Block Generation . . . 19
- 5.5 Block Reduction . . . 20
- 5.6 Block Agreement . . . 21
- 5.7 Block Generator . . . 22
- 5.8 Provisioner . . . 22
- 5.9 Reputation Module . . . 22

6 Injecting Privacy Into Smart Contracts **23**

7 Future Work **23**
- 7.1 Transaction Model . . . 23
- 7.2 Block Compression . . . 24
- 7.3 Networking . . . 24

1 Introduction

The idea of digital currencies deployed in a distributed network secured via cryptographically and game-theoretically sound primitives rather than trust has been a point of discussion in limited circles of enthusiasts for decades before being formalized for the first time by David Chaum [Cha82]. Between then and the introduction of Bitcoin [Nak08] in 2008, numerous researchers [Cha82; LSS96; Wei98; VCS03; Sza05] in the field attempted to propose a viable digital currency protocol with mainstream acclaim.

The first mainstream breakthrough happened with release of the Bitcoin whitepaper [Nak08], which ushered a new era of research and enthusiasm. Built on top of a novel digital ledger called "blockchain" and secured via a Proof-of-Work consensus protocol, inspired by [DGN04] and [Bac02], Bitcoin became the first truly decentralized digital currency, inspiring the work on other decentralized applications, such as decentralized DNS [Nam11] and distributed state machine [Woo19]. Soon after the release of Bitcoin, researchers started discovering numerous issues previously unbeknownst to the creator/s of Bitcoin. [KCW13; ES18; GKL15; SSZ17; PSS17; Bon16] have discovered deficiencies in the assumptions outlined in the Bitcoin whitepaper with regards to the consensus protocol and the economic model. Also, the paper [RS12] published by Ron and Shamir has been the first of many to demonstrate the ease of transaction analysis and the lack of anonymity that the Bitcoin users maintain.

The issue of excessive energy consumption required to retain the security guarantees of the data stored in the ledger has been another point of contention for the Bitcoin protocol. Throughout the years, multiple researchers have tackled the issue with various solutions, the majority of which revolved around a concept of "one-vote-per-share" instead of "one-vote-per-CPU". The idea of Proof-of-Stake was first formalized in the Peercoin whitepaper [KN12], followed by [Ben+14; BG17]. A more formal approach was taken by [DPS16; Kia+17; Dav+18]. The protocols referenced above belong to a family of "chain-based" Proof-of-Stake protocols, which essentially emulate the Proof-of-Work family of protocols while maintaining similar security assumptions. The downside of probabilistic finality of "chain-based" was tackled by Algorand [Mic16], which utilized various novel techniques to guarantee instant finality while retaining the "permission-lessness" of the underlying protocol. Unfortunately, the protocol came with it's own disadvantages, mainly revolving around the security assumptions (67% of the circulating supply is required to be honest and participate in the consensus execution) as well as the committee (2000+) and certificate sizes.

Understanding the importance of anonymity, researchers began working on techniques to convert Bitcoin into an anonymity-preserving protocol. The initial idea was to utilize mixers, trusted services which

combine the inputs and outputs of multiple users into a single transaction. The downsides of the service was the reliance on trust as well as the lack of obfuscation of the amounts involved. The initial resurgence of interest in anonymity-preserving digital currencies was followed by the publications of [Sab13; Hop+19; Max15; NMM16; Poe16; Fau+18; Bun+19], which took differing approaches to the problem with differing outcomes. The resulting rise of interest, has seen multiple projects, such as Monero and Zcash, surge to popularity with preservation of anonymity being the main selling point.

2 Our Contributions

Our contributions include the development of a novel Private Proof-of-Stake protocol (to be discussed more thoroughly in **Section 5.3**), a permission-less Proof-of-Stake protocol with statistical finality guarantees (**Section 5**), a quasi-Turing-complete Virtual Machine with zero-knowledge proof verification capability (**Section 6**) and a confidentiality-preserving account-based transaction model (**Section 7.1**).

3 Abstract Definition of the Protocol

The core of the Dusk Network protocol is comprised of a digital ledger, called blockchain. The concept of a blockchain was introduced in the Bitcoin whitepaper [Nak08] and has become a mainstay among the digital currency platforms. In a blockchain, data is assembled into primitives called "blocks", which are cryptographically linked together. The Dusk Network protocol block includes a "block header" (to be described below), a body containing the transactional data and a certificate containing the consensus-related data. The block headers contain the essential information about the corresponding block including the round integer, a seed, a hash of the of the certificate discussed above, the root of the Merkle Tree containing the transactions included in the block and a root of the Patricia-Merkle tree containing the contract state data, alongside a cryptographic fingerprint (to be explored in **Section 4.1**) of the previous block, effectively forming a "chain" of chronologically descending blocks, starting from the latest block, also known as a "head".

Block
`blockHeader`
`body`
`certificate`

Table 1: Dusk Network block structure

Block Header
previousBlockHash
round
seed
certificateHash
txRoot
stateRoot

Table 2: Dusk Network block header structure

However, the data incorporated into the blockchain cannot be considered immutable in a distributed environment, unless the blockchain is secured through a state machine replication algorithm, otherwise known as a "consensus protocol" in modern literature. Consensus protocols, briefly discussed in the **Section 1**, are a family of algorithms crafted to produce a consistent log (i.e. a store of data) under various scenarios. In this section, we are mostly concerned with Byzantine Fault-Tolerant [LSP82] protocols, designed to tolerate up to a certain threshold of participants exhibiting arbitrary behaviour (either due to a software bug or an adversarial corruption).

The Bitcoin consensus protocol [Nak08], otherwise known as "Nakamoto consensus", in honour of the pseudonymous author of the Bitcoin whitepaper, as well as other protocols belonging to the family of Proof-of-Work protocols are examples of probabilistic consensus protocols. Probabilistic consensus protocols, unlike protocols with instant or near-instant finality, cannot provide guarantees that the data included in the log is not reversed at a later stage, instead relying on the data being added to the log later on featuring the aforementioned data until the probability of the former data being reversed is considered to be negligible (statistically insignificant). In the meantime, the intermediate logs of some honest participants (participants adhering to the protocol rules) can deviate from the logs of the other honest participants, in an event called a "fork". Forks tend to reconcile after a certain period with the entire set of honest participants stabilizing on a uniform log, meaning that there exists a single agreed upon "truth". On the other hand, consensus protocols with instant or near-instant finality cannot or have a negligible probability of a fork. Sometimes referred to as "BFT-based" protocols, inspired by Practical Byzantine Fault-Tolerance [CL99], a revolutionary solution of the Byzantine Generals' Problem [LSP82], this family of protocols rarely features a "permission-less" protocol (a common feature amongst the probabilistic protocols), therefore deviating towards a more centralized execution as the security of the protocol relies on a constant preapproved set of consensus participants. Segregated Byzantine Agreement is a permission-less Proof-of-Stake protocol with statistical finality. Statistical finality implies that the probability of a fork is negligible. SBA promotes the segregation of roles through an introduction of two distinct consensus roles:

Block Generator and Provisioner. Block Generators utilize a novel Private Proof-of-Stake mechanism, called "Proof-of-Blind Bid" (to be explored in **Section 4.2**), to compete for the leadership of the round and produce a candidate block as a result. On the other hand, Provisioners utilize their stakes to compete for the formation of the committees to reach an agreement on a uniform candidate block and appending the finalized block to the existing blockchain.

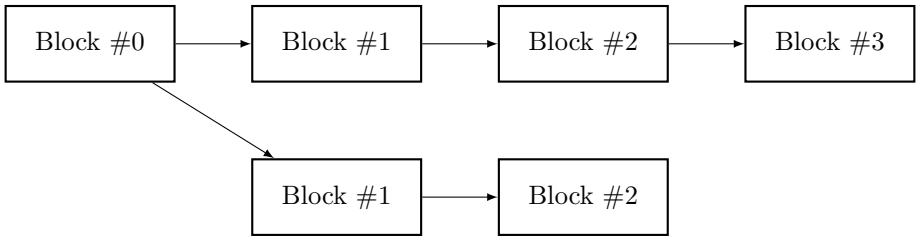


Figure 1: A "forked" chain

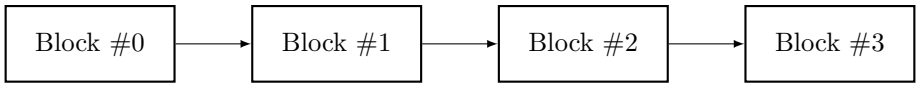


Figure 2: A "uniform" chain

The data included in the blocks consists of block metadata and user transactions. Conceptually, the Dusk Network protocol can be split into two layers: a transactional layer and a state layer. Transactional layer encompasses the entire set of transactions included in the blocks whereas the state layer is acted upon through state-transitional transactions, which represent a subset of transactions. The state is not directly stored in the block, rather having a cryptographic representation of a state included in the block header (`stateRoot`; see **Table 2**). The state consists of the contract-related datastore.

4 Cryptographic Primitives

The section outlines the cryptographic primitives which are utilized extensively in the protocol. They provide the security, anonymity and immutability required to fulfill the mission of the Dusk Network protocol .

4.1 Hash Functions

Cryptographic hash functions compress arbitrary long strings to fixed length outputs

$$\mathcal{H}(S) \longrightarrow T$$

and are the foundation of a number of cryptographic protocols. Dusk Network uses hash functions in several capacities: to compress long messages for signatures, to aggregate large sets into Merkle trees, to compute score functions in consensus, etc. When a statement on the hash function input is proven in zero-knowledge, we utilize a zero-knowledge proof-friendly hash function called Poseidon (**Section 4.1.2**), whereas for the other use cases we utilize a faster SHA-3 function (**Section 4.1.1**).

Only secure hash functions are used. The security of hash functions boils down to the following three properties:

1. **Preimage resistance**. It is infeasible for given T to find any S such that $\mathcal{H}(S) = T$.
2. **Second preimage Resistance**. For any known S it is infeasible to find $S' \neq S$ such that $\mathcal{H}(S) = \mathcal{H}(S')$.
3. **Collision resistance**. It is infeasible to find any $S \neq S'$ such that $\mathcal{H}(S) = \mathcal{H}(S')$.

For all hash functions that we use there exists no attack violating these properties that takes less than 2^{128} hash function calls, i.e. the hash functions have security level of 128 bits against all attacks.

In addition to the properties above, we assume that the hash functions we utilize can instantiate random oracles [BR93]. The latter is an imaginary object which produces an unpredictable output from any input that has not been previously asked from it. Dusk Network utilizes several protocols, that are proven secure assuming the underlying hash function is a random oracle.

4.1.1 SHA-3

SHA-3 [Dwo15] hash function is an international standard, a modification of a winning proposal [Ber+13] of a public competition held between 2006 and 2010. So far no attacks violating the security properties of SHA-3 have been found even for variants with twice smaller rounds. The Dusk Network protocol utilizes the SHA-3-256 variant for 128-bit security.

4.1.2 Poseidon

Poseidon [Gra+19] is a recently developed family of hash functions with zero-knowledge-friendly design. Concretely, Poseidon can be compactly

represented as an arithmetic circuit over a prime field, which is convenient for many zero-knowledge proof systems developed recently, including the Bulletproofs (**Section 4.4.2**) system we use. We utilize Poseidon in protocols where we prove statements about pre-images to hash functions or their composition in trees: digital signatures, blind bid computation, and membership proofs for large sets. Poseidon yields significant improvement in both zero knowledge proof generation and verification time, being 4-5 times faster than Pedersen hash used in Zcash [Hop+19] and more than 100 times faster than SHA-256.

4.2 Elliptic Curves

Elliptic curves is a widely used cryptographic primitive, which is popular for signatures, proof protocols, key agreement and many other use cases. The main advantage of elliptic curves is that finding the discrete logarithm problem in a group of curve points is hard, and the 128-bit security is achieved for rather small fields (from 256- to 384 bits) compared to integer fields where much bigger (2048 bits and higher) modulus is required. The discrete logarithm problem is defined as follows. For given curve points G and H on the curve find integer x such that $s \cdot G = H$ where $\cdot$ is the scalar multiplication in the group.

The Dusk Network protocol utilizes Elliptic-curve-based Diffie-Hellman Key Agreement (ECDH) to generate shared symmetric key for secure communication.

4.2.1 Ristretto

The number of points on a curve, also called the curve order, is a composite number for some curves, whereas some protocols, e.g. zero-knowledge proof systems, prefer prime-order groups. Moreover, it is desired that any point on a curve can be mapped efficiently to such a group.

Ristretto [LV18] creates a prime order group from the group of points on the fast Curve25519 (though it can be used for other curves with cofactors 4 and 8). The resulting group is 8 times smaller than the order of Ed25519 and the corresponding scalar field is called Ristretto field. As a result, Ristretto keeps the high performance of Curve25519 while providing a prime order group for ZK protocols.

Details are as follows. Curve25519 has the following parameters:

Curve equation: $y^2 = x^3 + 486662 \cdot x^2 + x$
p: $2^{255} - 19$
order: $2^{252} + 27742317777372353535851937790883648493$
co-factor: 8

Curve25519 is birationally equivalent to a ed25519 [J B+11], a Twisted Edwards curve.

4.2.2 Zerocaf

As we want elliptic curve signatures efficiently verifiable in zero knowledge, and also want to prove the knowledge of a private key for 1-of-many public keys, we utilize Doppio curve whose prime field is identical to the Ristretto scalar field. Zerocaf [PP19] is an implementation of the Doppio curve, which has the following parameters:

$$\textbf{Curve equation}: y^2 = x^3 + 346598 \cdot x^2 + x$$
$$p\text{: } 2^{252} + 27742317777372353535851937790883648493$$
$$\textbf{order}: 2^{249} - 151450387072189107654823447297780854 01$$
$$\textbf{co-factor}\text{: } 8$$

The use of a Ristretto scalar field allows for the capitalization of the speed of Twisted Edwards curve arithmetic, without the pitfalls through security or implementation

4.3 Signature Schemes

With a digital signature one can authenticate a message by binding his private key to it in a verifiable way. To process a long message, a cryptographic hash function is employed (**Section 4.1**), whereas a private key is mixed with a hash on an elliptic curve (**Section 4.2**). Signature schemes have to adhere to the following properties:

1. **Unforgeability** – No efficient adversary can produce a signature for a given message with non-negligible probability.
2. **Message binding** – No efficient adversary can find another message for which the same signature value is valid even if he knows the private key.
3. **Non-malleability** – The signature value can not be modified to another value valid for the same message.

4.3.1 EdDSA

EdDSA is an elliptic-curve-based digital signature scheme proposed in [J B+11]. It is known for its fast performance thanks to the curve choice (the same used in Ristretto **Section 4.2.1**). The signature scheme works as follows:

1. Public key $PK = s \cdot G$, where $s = \mathcal{H}(sk_1)$, sk_1 is the private key and G is the group generator.
2. $R = r \cdot G$, where $r = \mathcal{H}(sk_2, M)$, where M is the message and sk_2 is the signature private key.
3. $S = r + \mathcal{H}(R, PK, M) \cdot s$.

The signature is (R, S).

To verify the alleged signature (R, S), one checks the following equation:

$$8 \cdot S \cdot G = 8 \cdot R + 8 \cdot \mathcal{H}(R, PK, M) \cdot PK.$$

EdDSA is used in the message authentication scheme of the Dusk Network protocol to preserve the integrity of the peer-to-peer message exchange.

4.3.2 bLSAG/MLSAG

Ring signature is a primitive that allows a signer to hide within a set of predefined public keys (a *ring*). When used in digital currencies, ring signatures are utilized to prove coin spending, so a double-spent protection is required. A *linkable* ring signature gives this protection as signing two messages with the same key is detected. Formally, a secure linkable ring signature scheme has to satisfy the following properties:

1. **Anonymity** – The identity of the signer can not be deduced from the signature by an efficient adversary. The ring signature only reveals that the private keys belongs to one of the identities in the ring.
2. **Linkability** – The private key used to sign two different messages will be linked.
3. **Exculpability** – The identity of the signer can not be deduced from the signature even if all the private keys are known.

Based on the earlier works, [LKW04] defined a novel ring signature scheme called Linkable Spontaneous Anonymous Group signature. Unfortunately, in LSAG, the linkability was only offered in the signatures utilizing the same rings. To mitigate that, Adam Back had proposed [Bac02] a modification to LSAG that came to be known as Back Linkable Spontaneous Anonymous Group signature. bLSAG works as follows:

1. $PK_i = sk_i \cdot G$.
2. $I = sk_i \cdot \mathcal{H}^q(PK_i)$, where $\mathcal{H}^q$ is a hash to curve function.
3. Generate random numbers $\alpha \in \mathbb{Z}_q$ and $\forall_{j \neq i} r_j \in \mathbb{Z}_q$.
4. Calculate $c_{i+1} = \mathcal{H}(M, \alpha \cdot G, \alpha \cdot \mathcal{H}^q(PK_i)$, where M is the message to be signed.
5. $\forall_{j \neq i} c_{j+1} = \mathcal{H}(M, r_j \cdot G + c_j \cdot PK_j, r_j \cdot \mathcal{H}^q(PK_j), c_k \cdot I)$.
6. Calculate $r_i = \alpha - sk_i \cdot c_i$.

The signature is $(c_1, r_1, ..., r_n, I)$. To verify it, one calculates the value of c_1.

[NMM16] improves bLSAG to enable signing of multiple inputs with multiple private keys in a scheme called Multi-Linkable Spontaneous Anonymous Group signature.

4.3.3 BLS

Boneh-Lynn-Shacham signatures [BLS01] utilize pairing-friendly elliptic curves, such as BN-256 [NNS10] and BLS12-381 to create a short aggregatable signature scheme. Since signatures are elements of an elliptic curve, they are short.

Let e be a bilinear pairing $\mathbb{G}_0 \times \mathbb{G}_1 \leftarrow \mathbb{G}_\mathbb{T}$ with three groups of order q where G_0 and G_1 are the generators for groups $\mathbb{G}_0$ and $\mathbb{G}_1$. The BLS signature is defined as follows:

1. Public key $PK = sk \cdot G_1$, where sk is the secret key.
2. Signature $\sigma = (\mathcal{H}(m))^{sk} \in \mathbb{G}_0$, where m is the (hash of the) message to be signed.

The alleged signature σ is verified as follows:

$$e(G_1, \sigma) = e(PK, \mathcal{H}(m))$$

BLS signature scheme enables signature aggregation, which is utilized in the Dusk Network protocol to decrease the footprint of the consensus protocol execution on the block. Let $\mathcal{H}_2^m$ be SHAKE128, a variable-output-length hash function with output length m bits. Then define:

1. $T = (t_1, t_2, \ldots, t_n) \in (Z_{2^{128}}^n) = \mathcal{H}_2^{128n}(PK_1, PK_2, \ldots, PK_n)$
2. Aggregated signature is $\sigma = \sigma_1^{t_1} \cdots \sigma_n^{t_n}$.

To verify the alleged aggregated signature σ, do

1. Compute aggregated public key $apk = PK_1^{t_1} \cdots PK_n^{t_n}$
2. Verify $e(G_1, \sigma) = e(apk, \mathcal{H}(m_1))$.

4.4 Zero-knowledge Proofs

Zero-knowledge protocols are essential to Dusk Network as we protect both the privacy of protocol users and confidentiality of the data they exchange with.

A secure zero knowledge proof of knowledge protocol has the following properties:

1. **Completeness** – An honest Prover $\mathcal{P}$ succeeds in convincing the Verifier $\mathcal{V}$ of the statement.

2. **Soundness** – Malicious Prover fails to prove the truth of the statement.

3. **Zero-knowledgeness** - The proof reveals no information other than the fact that the statement is true.

4.4.1 Commitment Scheme

Commitment is an important cryptographic primitive that allows committing to a certain value without revealing it but later being able to prove that a certain value was committed to. Formally, a secure commitment scheme must be hiding (value can not be extracted) and binding (no other value can be opened to).

Pedersen Commitment

Pedersen Commitment scheme is defined as

$$\mathcal{C}(x; r) = r \cdot G + x \cdot H$$

where G, H are group elements whose discrete logarithm relation is unknown, $+$ is the group operation and $\cdot$ is a scalar multiplication in the group. In our case the group is the group of points on an elliptic curve.

Pedersen commitment is information hiding (not even immensely powerful adversary can deduce the message) but only computationally binding (efficient adversaries can not open a commitment to a different message). It is partially homomorphic:

$$\mathcal{C}(x_1; r_1) + \mathcal{C}(x_2; r_2) = \mathcal{C}(x_1 + x_2; r_1 + r_2).$$

To guarantee that the discrete logarithm of G to H is unknown, the latter is a hash of the former:

$$H \leftarrow \texttt{hashToPoint}(\mathcal{H}(G)).$$

Vector Pedersen Commitment

Pedersen commitment scheme can be extended to vectors $\mathbf{v} = \{v_1, ..., v_n\}$ to be committed:

$$\mathcal{C}(\mathbf{v}) = r \cdot G + \sum_i v_i \cdot H_i = r \cdot G + \mathbf{v} \cdot \mathbf{H}$$

with $\mathbf{H} = \{H_1, ..., H_n\}$ is generated as follows:

$$H_i \leftarrow \texttt{hashToPoint}(\mathcal{H}(G||i)) \text{ for all } i \in \{1, ..., n\}.$$

4.4.2 Bulletproofs

Bulletproofs [Bun+18] is a zero-knowledge protocol designed for range proofs for committed values and for proofs of computational integrity of arithmetic circuits[1]. The proofs are short (logarithmic in the circuit size) and do not require a trusted setup (i.e. the circuit description and supplementary data needed for proofs are generated using only public randomness). However, the verification time scales linearly with circuit size. This makes Bulletproofs more suitable for proving the integrity of small circuits.

We use Bulletproofs to prove computational integrity of several circuits, most of which invoke a Poseidon hash function. Even though the performance of Poseidon is slower than regular hash functions such as SHA-3, it is smaller as a circuit thus making the Bulletproofs proof shorter and its generation and verification faster.

[1] Computations in a prime field using only multiplication and addition gates.

5 Reaching Consensus

As discussed in the earlier sections, consensus protocols play a vital role in the security of distributed storage and state machines, so the inevitability of the fact that the entire section would be dedicated to the vastly complex topic of state machine replication is fairly self-evident. The section concentrates on Segregated Byzantine Agreement, a novel consensus protocol briefly mentioned in the **Section 3**. The section will first highlight definitions and the security assumptions of the consensus protocol before assessing the important highlights of the protocol in question.

5.1 Definitions

The definitions and the security assumptions of the design tend to shape the use cases of the protocol. Segregated Byzantine Agreement is no exception to the rule with the goal being to define the most efficient Proof-of-Stake protocol which would persist to be permission-less with near-instant or instant finality. SBA is a permission-less Proof-of-Stake protocol with statistical finality guarantees. The protocol relies on the Honest Majority of Money (an Adversary can corrupt consensus participants controlling up to f percent of the total stake value [$\geq 3f+1$]) assumption.

The node definitions are inspired by [PS17]. In the particular case, a node can be described as a deterministic Turing Machine which is differentiated through the secret K in Block Generators (see **Section 5.7**) case and a public key pk in Provisioners (see **Section 5.8**) case. Nodes joining the network can be honest or corrupt. Once a node becomes corrupt, the stake under the control of the particular node is considered to be corrupt. A node is defined as an honest node if it does not deviated from the prescribed protocol execution rules. On the other hand, a corrupt node is permitted to exhibit arbitrary behaviour, meaning that it can deviate from the prescribed protocol execution rules. The number of nodes is orthogonal to the security of the consensus protocol, as long as the Honest Majority of Money assumption is preserved, due to each staked DUSK being abstractly evaluated as a unique node. An Adversary $\mathcal{A}$ is permitted to corrupt an honest node by issuing a `CORRUPT` message as long as the percentage of total stake under the control of the corrupt nodes does not exceed f. The nodes can be *awake* or *asleep*, with a node asleep for less than n rounds being called a *light sleeper* (for brevity, we assume that the corrupt nodes are always awake) and a node asleep for more than n round being called a *deepsleeper*. An Adversary $\mathcal{A}$ is permitted to put the node to sleep by issuing a `SLEEP` message as long as the total size of the stake under control of the honest *awake* nodes exceeds $2f$. An Adversary $\mathcal{A}$ is permitted to wake up the node by issuing a `WAKEUP` message. If a node has been asleep for more that n rounds before receiving a `WAKEUP` message, it respawns as a new node.

Definition. A consensus protocol has statistical finality when the probability of a fork during a single execution round is negligible.

Proof. In case of Segregated Byzantine Agreement, the fork can be produced by "double-voting" during the consequent steps of the execution round (two Block Reduction steps (see **Section 5.5**) and a Block Agreement step (see **Section 5.6**). A "double-vote" occurs when a node votes for two separate candidate blocks in the same voting step. Taking the node assumptions defined above into account, a "double-vote" can only be produced by a corrupt node, meaning that in order to produce a fork, an Adversary has to receive the control of the supermajority in three consequent consensus steps. The probability of the aforementioned event happening can be defined through a formula below.

Assumption. $\frac{|honest| \cap |awake|}{|corrupt|} \geq 1 + \epsilon$, where $\epsilon > 1$.
The Adversary $\mathcal{A}$ is allowed to corrupt and coordinate nodes in control of up to f percent of the total stake. $\mathcal{A}$ is a mildly adaptive Adversary, meaning that the corruption of a node takes place t rounds after the `CORRUPT` message is issued. $\mathcal{A}$ does not have a power to shuffle the order of the messages delivered to the honest nodes.

The nodes on the network are assumed to have loosely synchronized clocks and the network itself is presumed to be weakly synchronous. A weakly synchronous network can delay the messages up to a bound Δ, which is unknown.

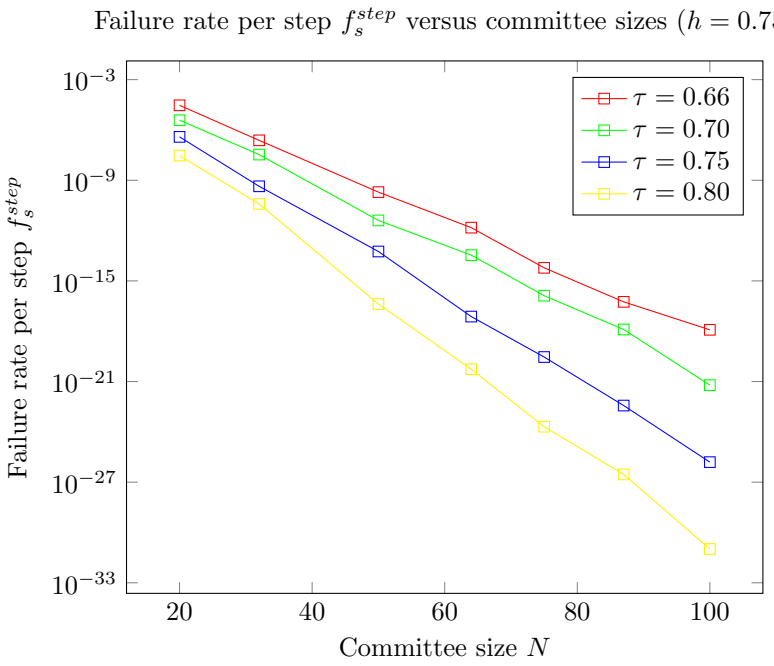
Failure Rate

The failure rate is the probability of the safety guarantees of the protocol step (excluding Block Generation [**Section 5.4**] being breached. In particular, the failure rate indicates the probability of an Adversary $\mathcal{A}$ obtaining a supermajority in a committee. The probability function is outlined in a formula below, where N is the committee size, τ is the threshold of votes in a committee required to proceed to the next step and h is the ratio of the honest participants:

$$\sum_{k=\texttt{floor}(\tau \cdot N+1)}^{N} \binom{N}{k} \cdot (1-h)^k \cdot h^{N-k} = \sum_{k=\texttt{ceil}(\tau \cdot N)}^{N} \frac{N! \cdot (1-h)^k \cdot h^{N-k}}{k! \cdot (N-k)!} \leq f_s^{step}$$

Figure 3: The formula calculating the failure rate per step f_s^{step}.

The probability of an Adversary $\mathcal{A}$ successfully creating a fork is equal to the probability of an Adversary $\mathcal{A}$ obtaining a supermajority in the two consequent Block Reduction steps ($([Pr]_{BR}^s)^2 \leq (f_s^{step})^2$) and a Block Agreement step ($([Pr]_{BA}^s \leq f_s^{step})$) or $([Pr]_{BR}^s)^2 \cdot [Pr]_{BA}^s = (f_s^{round})^3 \leq f_s^{round}$, which is mapped on a graph below with varying committee sizes:



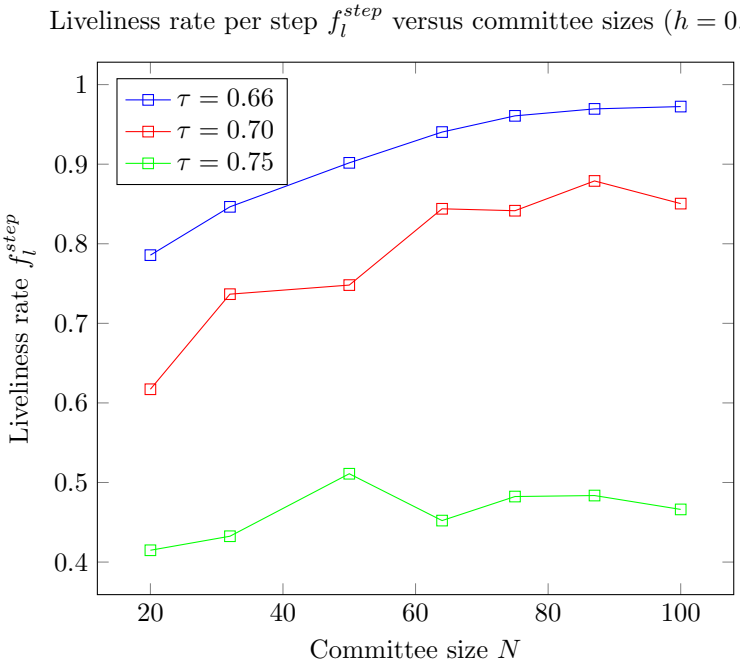
Liveliness Rate

The liveliness rate indicates the probability of an honest majority being obtained in a committee. The probability function is outlined in a formula below:

$$1 - \sum_{k=1}^{\texttt{floor}(\tau \cdot N)} \binom{N}{k} \cdot h^k \cdot (1-h)^{N-k} = 1 - \sum_{k=1}^{\texttt{floor}(\tau \cdot N)} \frac{N! \cdot h^k \cdot (1-h)^{N-k}}{k! \cdot (N-k)!} \leq f_l^{step}$$

Figure 4: The formula calculating the failure rate per step f_l^{step}.

The probability of a successful consensus round termination is equal to the probability of an supermajority obtained in the consequent Block Generation ($[Pr]_{BG}^l \geq h$), two Block Reduction ($([Pr]_{BR}^l)^2 \geq (f_l^{step})^2$) and Block Agreement ($[Pr]_{BA}^l \geq f_l^{step}$) steps or $[Pr]_{BG}^l \cdot ([Pr]_{BR}^l)^2 \cdot [Pr]_{BA}^l = h \cdot (f_l^{step})^3 \leq f_l^{round}$, which is mapped on a graph below with varying committee sizes:



5.2 Segregated Byzantine Agreement

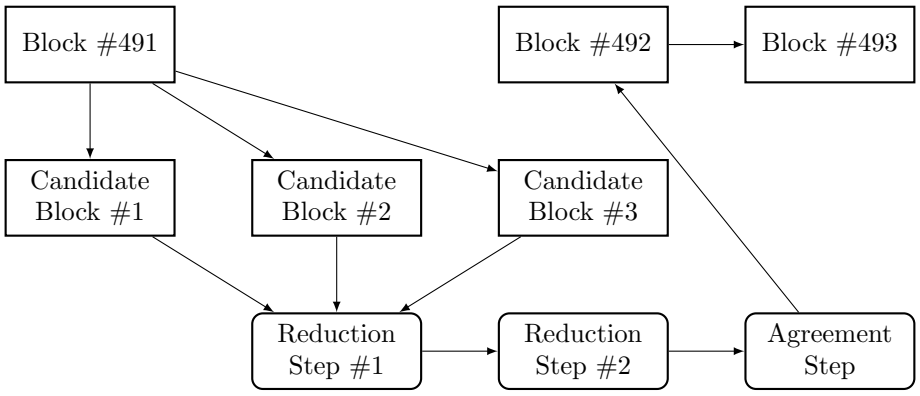


Figure 5: Segregated Byzantine Agreement

The roles in the protocol are split between two distinct node types: Block Generators and Provisioners. Block Generators retain their privacy, with the proofs of stake computed in zero-knowledge to preserve the anonymity of the Block Generator. On the other hand, Provisioners are required to deanonymize their stakes and remain transparent about their activities in the consensus while their stake remains valid.

Both bids and stakes have a registration maturity of $2 \cdot t$, which begins when a Bid or Stake transaction is included in a final block and is required to elapse before the node is eligible to participate in the consensus.
Segregated Byzantine Agreement is made up of a single loop containing three phases: Block Generation (**Section 5.4**), Block Reduction (**Section 5.5**) and Block Agreement (**Section 5.6**). The first two rely on the weakly synchronous assumptions of the network highlighted in **Section 5.1** while the Block Agreement phase is asynchronous. The asynchronicity of the third phase is vital to enable the consensus to remain "fork-free" even under the most strenuous conditions defined in **Section 3.1**.

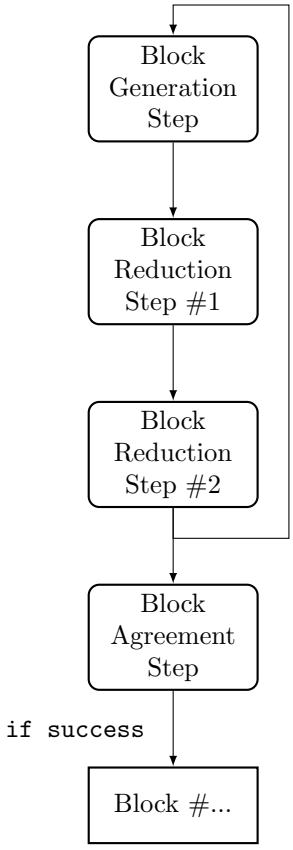


Figure 6: Segregated Byzantine Agreement round

5.3 Proof-of-Blind Bid

The Proof-of-Blind Bid protocol is an implementation of a Private Proof-of-Stake protocol conceptualized by Dusk Network. Private Proof-of-Stake protocols fulfill the requirements of Proof-of-Stake protocols

while additionally preserving the anonymity of the participants of the protocol execution trace and the values of the corresponding stakes. The Proof-of-Blind Bid protocol enables the underlying Proof-of-Stake protocol to extract the leaders for the corresponding round in a black-box manner. The Proof-of-Blind Bid protocol represents a more efficient implementation of Private Proof-of-Stake to [GOT19].

The Proof-of-Blind Bid protocol requires the valid bids to be stored in a Merkle Tree. From there, when competing for a leadership position to generate a candidate block, a Block Generator is required to prove the inclusion of his bid in the Merkle Tree, followed by the proof of knowledge of the secret K (the cryptographic hash of which is included in the bid). Afterwards, the Block Generator computes his/her score and proves the correctness of the computation before propagating the score alongside the candidate block. The process can be conceptualized in a function defined below:

Function $\mathcal{F}_{PoBB}$

$\mathcal{F}_{PoBB}$ is a function to generate a blind bid score and proof of the correct computation which takes the *bid* (where bid represents a tuple of $(commitment, \mathcal{H}(K))$, bid value d and secret K as an input and outputs *score* and proof π

$\mathcal{F}_{PoBB}(bid, d, K, context)$:

1. Check if *bid* belongs to the Merkle Tree bid set $\mathcal{T}$.
2. If true, compute the *score* using the *seed*, *round*, *step*, *bid*, d and K..
3. If $score > difficulty$, then compute the proof π. The proof π has to consist of the following statements:
   (a) $bid \in \mathcal{T}$.
   (b) Knowledge of the secret K.
   (c) Correctness of the computation of the *score*,
4. Return *score* and π.

5.4 Block Generation

The Block Generation function is utilized in Segregated Byzantine Agreement to participate in the generation of the candidate blocks based on the outcome of the Proof-of-Blind Bid execution. If a Block Generator has successfully executed the Proof-of-Blind Bid protocol and received a score greater than

the predefined difficulty, then it can proceed with the Block Generation.

Function $\mathcal{F}_{BG}$

$\mathcal{F}_{BG}$ is a function to produce a *candidateblock*,

$\mathcal{F}_{PoBB}(bid, d, K, context)$:

1. Check if *bid* belongs to the Mekle Tree bid set $\mathcal{T}$.
2. If true, call $\mathcal{F}_{PoBB}(bid, d, K, context)$.
3. If $score > difficulty$, then create *candidateblock*.
4. Propagate *score* and π followed by *candidateblock*.

5.5 Block Reduction

The Block Reduction function is utilized in Segregated Byzantine Agreement is based on [TA84], which reduces the multivariable inputs to a single variable output before proceeding to Binary Agreement. However, unlike BBA★ [Mic17], the Turpin and Coan algorithm is not utilized as a reduction function for a Binary Agreement protocol. Instead, the aforementioned algorithm is run in a 3-phase loop until a favourable outcome is reached. The function can be conceptualized as defined below:

Function $\mathcal{F}_{BR}$

$\mathcal{F}_{BR}$ is a function to reach an agreement on a uniform value.

$\mathcal{F}_{BR}(stake, x, context)$:

1. Check if *stake* belongs to the stake set $\mathcal{S}$.
2. If true, check if the node is elected for a committee in round r and step s.
3. If true, start timer t and propagate x.
4. If $2f$ messages with x are received, check if the node is elected for a committee in round r and step $s+1$, restart timer t and propagate x. Else, if timer t expires, check if the node is elected for a committee in round r and step $s+1$, restart timer t and propagate *nil*.
5. If $2f$ messages with x are received, proceed to $\mathcal{F}_{BA}$. Otherwise, if $2f$ messages with *nil* are received or the timer t expires, abort the execution and return to $\mathcal{F}_{BG}$.

5.6 Block Agreement

Block Agreement is an asynchronous function running in parallel with the main loop. Successful termination of the function indicates that the main loop has been successfully executed. The function provides a statistical guarantee that at least one honest and *awake* node has received a set of votes exceeding the minimum threshold required to successfully terminate the respective phase of the protocol.

Function $\mathcal{F}_{BA}$

$\mathcal{F}_{BA}$ is a function utilized to reach the terminate of the execution loop of the round if the previous two phases have been executed correctly.

$\mathcal{F}_{BA}(x)$:

1. Check if *stake* belongs to the stake set $\mathcal{S}$.
2. If true, check if the node is elected for a committee in round r and step s.
3. If true, propagate a set of at least $2 \cdot (2f + 1)$ signatures extracted from the $\mathcal{F}_{BA}$ messages.
4. If $2f$ valid messages are received, terminate the execution loop.

5.7 Block Generator

Block Generator is the first of the two node types eligible to participate in the consensus. To become a Block Generator, a node has to submit a Bid transaction.

The Block Generator is eligible to participate in one phase - Block Generation phase. In the aforementioned phase, Block Generators participate in a non-interactive lottery to be able to forge a candidate block.

5.8 Provisioner

Provisioner is the second of the two full-node types eligible to participate in the consensus. To become a Provisioner, a node has to submit a Stake transaction. Unlike a Block Generator, a Provisioner node is required to deanonymize the value of the stake to be able to participate in the consensus. While it is technically possible to obfuscate the stake value, the team has decided against the latter as the addition of stake value obfuscation would have slowed down the consensus and simultaneously increased the block size.

The Provisioner is eligible to participate in two phases - Block Reduction and Block Agreement.

5.9 Reputation Module

The reputation module, originally formalized in [BFK17], enables Dusk Network to increase the security of the consensus protocol by assigning the Provisioners with a reputation based on their participation record in the consensus. The

technique permits the consensus to prevent potentially malicious behaviour as well as penalize the nodes with slow network connection or running a bug-ridden implementation of the protocol.

6 Injecting Privacy Into Smart Contracts

The state layer of the protocol, briefly discussed in the **Section 3** would have been an impossible feat without an underlying distributed state machine. In the particular case of Dusk Network, the state machine is realized in form of a Virtual Machine. In particular, the Dusk Network protocol includes native support of a Turing-complete Virtual Machine. The Virtual Machine is based on WebAssembly [Web17], a binary instruction set format for a stack-based Virtual Machine. The Virtual Machine defined by Dusk Network includes the native support of zero-knowledge proof verification, enabling the preservation of privacy on the state layer and shifting the majority of computational workload from the distributed state machine to the user. The native support of zero-knowledge proof verification is also vital to support the Confidential Security Contract (XSC) [Mah19] standard proposed.

7 Future Work

As the research and development of the Dusk Network protocol continues, the three subsections listed below represent the main priorities for the team, in terms of advancement of the security and the featureset of the protocol.

7.1 Transaction Model

Since the publication of the Bitcoin whitepaper [Nak08] more than a decade ago, two transactional models have gained an unequivocal duopoly amongst the digital currency protocols. UTXO (Unspent Transaction Output), introduced in the aforementioned whitepaper, implements a transaction structure consisting of the so-called "inputs" and "outputs", which balance to zero (including the transaction fees). The "input" in a transaction references an unspent "output", fulfilling the spending conditions defined in the previously mentioned "output", while the "output" defines the spending conditions for the future spenders to fulfill. The reader should note that the "inputs" are required to be spent in their entirety, meaning that if Alice wants to send 2 tokens to Bob and her "input" includes 5 tokens, she needs to include a "change output" of 3 tokens to herself in the transaction. On the other side of the aisle, an account model, pioneered by Ethereum [Woo19], stores the total balance of the address instead of tracking the unspent "outputs" available to that particular address. A special field labelled nonce plays a crucial role in preventing the "double-spend" attacks in protocols utilizing account models and the user is required to increment the nonce of the account when transacting while the network nodes are

required to keep track of the nonces attributed to the existing accounts. Various proposals [Leu+19] have been published throughout the years to alleviate the need for nonces, though none are currently utilized on a large scale.

The aforementioned duopoly is broken in the privacy-oriented digital currencies. Both Monero [NMM16] and Zcash [Hop+19] employ variations of the UTXO model, though neither directly reference the "outputs" in the "inputs" (with Zcash's JoinSplit transaction type being the exception), instead electing differing obfuscation techniques to obscure the correct input. The breakthrough in the direction of an account model geared towards anonymity-preserving protocols comes with the publication of [KV19], which formalizes an account model with Private memory.

Dusk Network protocol will utilize [KV19] in the state layer, enabling the functionality specified in [Mah19].

7.2 Block Compression

The scalability trilemma, term coined by Vitalik Buterin, represents the biggest problem yet to be solved in the blockchain field. [Cro+16] outlines numerous improvements which could potentially improve the scalability of the distributed networks. Alongside other potential solutions, the team at Dusk Network has decided to specifically tackle one issue relating to block propagation. Propagation of the block is seen as the main bottleneck preventing Segregated Byzantine Agreement to achieve a major speed-up. The team has begun working on an implementation of [Pin+17] as well as [Din+19] in order to assess the performance of the two algorithms and integrate the better-performing one into the Dusk Network protocol.

7.3 Networking

The original Dusk Network whitepaper had a section dedicated to the anonymous communication over the distributed network. Anonymous communication protocols, such as [RSG98], enable the users to communicate with other over the network without revealing their IP-addresses. During the early testing of the Dusk Network Devnet, the team had implemented a I2P protocol based on garlic routing to assess the communication penalty incurred due to the anonymity preservation. The results indicated a substantial slowdown in the message propagation times in comparison to the network with disabled I2P layer. As a result, the team has made a decision to work on the protocol improvements before deploying the anonymous network layer. The current version of the layer resembles [Kov18] more closely than the previous iteration, discussed above.

References

[Cha82] David Chaum. "Blind Signatures for Untraceable Payments". In: (1982). DOI: 10.1007/978-1-4757-0602-4_18.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. "The Byzantine Generals Problem". In: *ACM Transactions on Programming Languages and Systems* 4.3 (1982), pp. 382–401. DOI: 10.1145/357172.357176.

[TA84] Russell Turpin and Brian A. Coan. "Extending binary Byzantine agreement to multivalued Byzantine agreement". In: *Information Processing Letters* 18 (1984), pp. 73–76. DOI: 10.1016/0020-0190(84)90027-9.

[BR93] Mihir Bellare and Phillip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols". In: *Proceedings of the First Annual Conference on Computer and Communications Security* (1993). DOI: 10.1145/168588.168596.

[LSS96] Laurie Law, Susan Sabett, and Jerry Solinas. "How to Make a Mint: The Cryptography of Anonymous Electronic Cash". In: (1996).

[RSG98] M.G. Reed, Paul Syverson, and D.M. Goldschlag. "Anonymous connections and onion routing". In: *Selected Areas in Communications, IEEE Journal on* 16 (1998), pp. 482–494. DOI: 10.1109/49.668972.

[Wei98] Dai Wei. "b-money". In: (1998).

[CL99] Miguel Castro and Barbara Liskov. "Practical Byzantine Fault Tolerance". In: *OSDI* (1999).

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. "Short Signatures from the Weil Pairing". In: ASIACRYPT '01 (2001), pp. 514–532.

[Bac02] Adam Back. "Hashcash - A Denial of Service Counter-Measure". In: (2002).

[VCS03] Vivek Vishnumurthy, Sangeeth Chandrakumar, and Emin Sirer. "KARMA: A Secure Economic Framework for Peer-To-Peer Resource Sharing". In: (2003).

[DGN04] Cynthia Dwork, Andrew Goldberg, and Moni Naor. "On Memory-Bound Functions for Fighting Spam". In: *Lecture Notes in Computer Science* (2004). DOI: 10.1007/978-3-540-45146-4_25.

[LKW04] Joseph Liu, Victor K. Wei, and Duncan Wong. "Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups". In: *IACR Cryptology ePrint Archive* 2004 (2004). DOI: 10.1007/978-3-540-27800-9_28.

[Sza05] Nick Szabo. *Bit Gold*. 2005.

[Nak08] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: *Cryptography Mailing list at https://metzdowd.com* (2008).

[NNS10] Michael Naehrig, Ruben Niederhagen, and Peter Schwabe. "New Software Speed Records for Cryptographic Pairings". In: *IACR Cryptology ePrint Archive* 2010 (2010). DOI: 10.1007/978-3-642-14712-8_7.

[J B+11] Daniel J. Bernstein et al. "High-Speed High-Security Signatures". In: *Journal of Cryptographic Engineering* 2 (2011), pp. 124–142. DOI: 10.1007/978-3-642-23951-9_9.

[Nam11] Namecoin. *Namecoin*. 2011. URL: https://namecoin.org.

[KN12] Sunny King and Scott Nadal. "PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake". In: (2012).

[RS12] Dorit Ron and Adi Shamir. "Quantitative Analysis of the Full Bitcoin Transaction Graph". In: (2012). DOI: 10.1007/978-3-642-39884-1_2.

[Ber+13] G.M. Bertoni et al. "Keccak". In: (2013), pp. 313–314. DOI: 10.1007/978-3-642-38348-9_19.

[KCW13] Joshua A. Kroll, Ian C. Davey, and Edward W. Felten. "The Economics of Bitcoin Mining, or Bitcoin in the Presence of Adversaries". In: (2013).

[Sab13] Nicolas van Saberhagen. "CryptoNote v 2.0". In: (2013).

[Ben+14] Iddo Bentov et al. "Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake". In: *SIGMETRICS Performance Evaluation Review* 42 (2014), pp. 34–37.

[Dwo15] Morris Dworkin. "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions". In: (2015).

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. "The Bitcoin Backbone Protocol: Analysis and Applications". In: (2015), pp. 281–310. DOI: 10.1007/978-3-662-46803-6_10.

[Max15] Gregory Maxwell. "Confidential Transactions". In: (2015).

[Bon16] Joseph Bonneau. "Why Buy When You Can Rent?" In: 9604 (2016), pp. 19–26. DOI: 10.1007/978-3-662-53357-4_2.

[Cro+16] Kyle Croman et al. "On Scaling Decentralized Blockchains". In: *Bitcoin and Blockchain* 9604 (2016), pp. 106–125. DOI: 10.1007/978-3-662-53357-4_8.

[DPS16] Phil Daian, Rafael Pass, and Elaine Shi. "Snow White: Robustly Reconfigurable Consensus and Applications to Provably Secure Proof of Stake". In: (2016).

[Mic16] Silvio Micali. "ALGORAND: The Efficient and Democratic Ledger". In: (2016).

[NMM16] Shen Noether, Adam Mackenzie, and The Monero Research Lab. "Ring Confidential Transactions". In: *Ledger* 1 (2016), pp. 1–18. DOI: 10.5195/LEDGER.2016.34.

[Poe16] Andrew Poelstra. “Mimblewimble”. In: (2016).

[BFK17] Alex Biryukov, Daniel Feher, and Dmitry Khovratovich. “Guru: Universal Reputation Module for Distributed Consensus Protocols”. In: *IACR Cryptology ePrint Archive* 2017 (2017).

[BG17] Vitalik Buterin and Virgil Griffith. “Casper the Friendly Finality Gadget”. In: (2017).

[Kia+17] Aggelos Kiayias et al. “Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol”. In: (2017), pp. 357–388. DOI: 10.1007/978-3-319-63688-7_12.

[Mic17] Silvio Micali. “Byzantine Agreement, Made Trivial”. In: (2017).

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. “Analysis of the Blockchain Protocol in Asynchronous Networks”. In: (2017), pp. 643–673. DOI: 10.1007/978-3-319-56614-6_22.

[PS17] Rafael Pass and Elaine Shi. “The Sleepy Model of Consensus”. In: (2017), pp. 380–409. DOI: 10.1007/978-3-319-70697-9_14.

[Pin+17] A Pinar Ozisik et al. “Graphene: A New Protocol for Block Propagation Using Set Reconciliation”. In: (2017), pp. 420–428. DOI: 10.1007/978-3-319-67816-0_24.

[SSZ17] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. “Optimal Selfish Mining Strategies in Bitcoin”. In: (2017), pp. 515–532. DOI: 10.1007/978-3-662-54970-4_30.

[Web17] WebAssumbly. *WebAssembly*. 2017. URL: https://webassembly.org.

[Bun+18] Benedikt Bunz et al. “Bulletproofs: Short Proofs for Confidential Transactions and More”. In: (2018), pp. 315–334. DOI: 10.1109/SP.2018.00020.

[Dav+18] Bernardo David et al. “Ouroboros Praos: An Adaptively-Secure, Semi-synchronous Proof-of-Stake Blockchain”. In: (2018), pp. 66–98. DOI: 10.1007/978-3-319-78375-8_3.

[ES18] Ittay Eyal and Emin Sirer. “Majority Is Not Enough: Bitcoin mining is vulnerable”. In: *Communications of the ACM* 61 (2018), pp. 95–102. DOI: 10.1145/3212998.

[Fau+18] Prastudy Fauzi et al. “QuisQuis: A New Design for Anonymous Cryptocurrencies”. In: *IACR Cryptology ePrint Archive* 2018 (2018).

[Kov18] Kovri. *The Kovri Project*. 2018. URL: https://gitlab.com/kovri-project/kovri.

[LV18] Isis Lovecruft and Henry de Valence. *Ristretto*. 2018. URL: https://ristretto.group/ristretto.html.

[Bun+19] Benedikt Bunz et al. “Zether: Towards Privacy in a Smart Contract World”. In: *IACR Cryptology ePrint Archive* 2019 (2019).

[Din+19] Donghui Ding et al. “Txilm: Lossy Block Compression with Salted Short Hashing”. In: (2019).

[GOT19] Chaya Ganesh, Claudio Orlandi, and Daniel Tschudi. “Proof-of-Stake Protocols for Privacy-Aware Blockchains”. In: (2019), pp. 690–719. DOI: 10.1007/978-3-030-17653-2_23.

[Gra+19] Lorenzo Grassi et al. “Starkad and Poseidon: New Hash Functions for Zero Knowledge Proof Systems”. In: *IACR Cryptology ePrint Archive* 2019 (2019).

[Hop+19] Daira Hopwood et al. “Zcash Protocol Specification, Version 2019.0.6”. In: (2019).

[KV19] Dmitry Khovratovich and Mikhail Vladimirov. “Full privacy in account-based cryptocurrencies v 0.12”. In: (2019).

[Leu+19] Derek Leung et al. “Vault: Fast Bootstrapping for the Algorand Cryptocurrency”. In: (2019).

[Mah19] Toghrul Maharramov. “Confidential Security Contract (XSC) Standard”. In: (2019).

[PP19] Carlos Perez and Luke Pearson. “Zerocaf”. In: (2019).

[Woo19] Gavin Wood. “ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER”. In: (2019).