The Coordicide

Coordicide Team, IOTA Foundation

May 2019

Contents

1 Introduction **4**

2 State of the art **6**
2.1 Current IOTA implementation 6
2.2 Coo-less IOTA network 7
2.3 New research challenges 7

3 Node accountability **8**
3.1 Global node identities 9
3.2 Sybil protection 9

4 Auto Peering **10**
4.1 Peer discovery 10
4.2 Choosing neighbors 11
4.3 Network reorganization 12
4.4 Bootstrapping 13

5 Rate control **13**
5.1 Rate control algorithm 14
5.2 Implementation details 14
5.3 Verifiable delay functions 14

6 Consensus **16**
6.1 Tip selection 17
6.1.1 Random walk-based TSA 18
6.1.2 Alternative TSA 20
6.2 Voting 21
6.2.1 Fast probabilistic consensus 22
6.2.2 Cellular consensus 23

7 Conclusion **26**

Glossary

Byzantine node A participant in a distributed system which tries to damage its operation intentionally, e.g., by not forwarding messages to other participants.

Consensus The problem of agreeing on a specific data or value in distributed multi-agent systems in presence of faulty processes.

Coordinator A trusted entity issuing milestones in order to guarantee finality and protect the Tangle against attacks.

Dictionary attack A form of brute force attack technique for defeating an authentication mechanism by trying to determine its passphrase by trying millions of likely possibilities, such as words in a dictionary.

Eclipse attack A cyber-attack that aims to isolate and attack a specific user, rather than the whole network.

Genesis The first transaction ever generated in the Tangle.

Heartbeat A periodic signal generated by hardware or software to indicate normal operation or to synchronize other parts of a computer system.

History The list of transactions directly or indirectly approved by a given transaction.

Milestone A special transaction issued by the Coordinator. The properties of the milestones are discussed in detail in Section 2.1.

Neighboring nodes Nodes sharing the same link in a network.

Node A machine which is part of the IOTA network. Its role is to issue transactions and to validate existing ones.

Peering The procedure of discovering and connecting to other network nodes.

Proof-of-Work A piece of data which is difficult (costly, time-consuming) to produce but easy for others to verify and which satisfies certain requirements.

Rainbow table attack A precomputed table for reversing cryptographic hash functions.

Random walk A mathematical object that describes a path that consists of a succession of random steps on some mathematical space.

Salt A random number used as an additional input to a one-way function that hashes data.

Social engineering The use of deception to manipulate individuals into divulging confidential or personal information that may be used for fraudulent purposes.

Sybil attack A sybil attack is an attempt to gain control over a peer-to-peer network by forging multiple fake identities.

Tip A transaction that has not been approved yet.

Transaction A message that transfers funds or information between two nodes. A transaction is *solid* if its entire history is known.

Acronyms

ASIC Application-Specific Integrated Circuit.

CA Cellular Automata.

CLIRI Coo-Less IOTA Reference Implementation.

DAG Directed Acyclic Graph.

DLT Distributed Ledger Technology.

IF IOTA Foundation.

IRI IOTA Reference Implementation.

PoW Proof-of-Work.

TSA Tip Selection Algorithm.

VDF Verifiable Delay Function.

1 Introduction

IOTA's vision aims to establish a real-time economy for Internet-of-Things and the future Internet through a secure zero fee payment and data transmission system. Realizing this vision is subject to a combination of features that cannot be found in current distributed ledger technologies (DLTs): First, it is required significantly higher throughput than blockchains which have an intrinsic bottleneck forcing transactions to be aggregated under a chain-type data structure; second, fees can be considered a barrier for micro transactions, but they are necessary in PoW-based DLTs where the network distinguishes between miners and users. Conversely, IOTA utilizes a directed acyclic graph (DAG) structure as explained in the IOTA white paper [30] which permits a theoretical infinite throughput[1]. Furthermore, enabling each network participant to both issue and approve transactions allows IOTA to eliminate the fees found in blockchain architecture, thus facilitating a micropayment-ready network (see also [31]).

One common problem for early stage DLTs is that the networks are not robust enough for proposed security mechanisms to function as intended, since such security mechanisms presuppose a mature network. Therefore, it is typical that DLTs employ various "bootstrapping" security measures at the outset, ensuring network growth to the mature stage can take place[2]. Thus, in its current implementation, IOTA relies on a centralized Coordinator to provide security given the risk of dishonest actors seeking to undermine the nascent network. IOTA's definition of consensus requires a confirmed transaction to be referenced (either directly or indirectly) by a signed transaction issued by the Coordinator. In other words, the Coordinator can be thought of as a "finality device".

We believe that the vision of cryptocurrency networks based on Nakamoto consensus can be improved upon by changing the key underlying assumption about those controlling the majority of the network's hashing power being considered "honest" by definition (the "longest chain wins" rule). In IOTA, the requirement for honest actors to control a majority of the network's hashing power is currently replaced by the use of the Coordinator. The Coordinator is a temporary measure as the IOTA network develops beyond Nakamoto's vision for network consensus. The Coordicide project is focused on the removal of the Coordinator through the implementation of several network components, as discussed in this working paper. Despite these additional components, all existing fundamental design features of the Tangle remain in-place.

In line with the IOTA Foundation's charter as a non-profit organization, our goals as a research department include transparency, collaboration, and community engagement. We aim to open our research work in order to obtain feedback from academia as well as the broad community of enthusiasts. One note of caution, however, is in order: Since our research is highly dynamic in nature, proposed ideas need to be simulated and tested in order to develop

[1] The actual throughput is bounded by hardware limitations and by law of physics.

[2] For example, Bitcoin has in the past employed checkpoints. See https://en.bitcoin.it/wiki/Checkpoint_Lockin

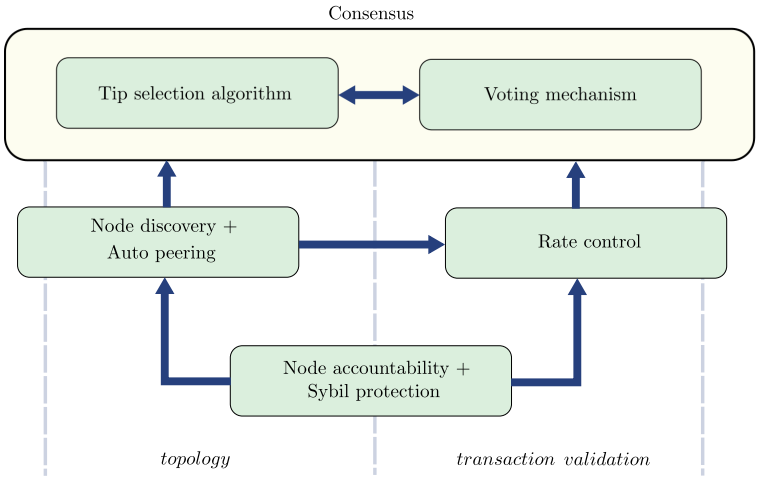


Figure 1: Interconnection between the Coordicide building blocks.

specific network components which we feel confident to deploy on the main network. We stress that some of the ideas presented here are works in progress and as such are not fully fleshed out. They are therefore likely to be modified as we make progress and perform simulations.

In order to get rid of the Coordinator, a number of challenges must be solved. This working paper covers those challenges, which are summed up by the building blocks in Fig. 1. In the following, we give a concise overview about the current state of the Coordicide as well as future research directions:

- *Node accountability.* In Section 3 we propose the concept of *global node IDs* and we describe a novel Sybil protection mechanism that does not require node owners to risk or disclose their funds. Identifying the issuing node of a message is fundamental to enforce a specific network topology (through auto peering) or to penalize bad behaviours (through rate control).

- *Auto peering and node discovery.* An automated process to discover and reliably connect to neighbors is needed in every distributed system. In Section 4 we discuss an auto peering proposal for the Tangle.

- *Rate control.* To ensure the network does not exceed its capacity, in Section 5 we introduce a mechanism to control the rate of transactions that are propagated through the network. This method selectively filters some transactions out according to the statistics of the issuing node.

- *Consensus.* The previous building blocks lead to an extended consensus framework described in Section 6. This is made up of two complementary building blocks: First, we describe the current status of research on tip selection algorithms (Section 6.1); then, to proactively resolve certain con-

flicts, we describe two voting mechanisms where nodes query other nodes to find out their current opinion on the network status (Section 6.2).

2 State of the art

In the introduction, we mentioned that the current IOTA main network uses the Coordinator to reach consensus and, more generally, to guarantee the security of the network. However, this centralized component should only be considered as a necessary bootstrapping mechanism, rather than a long-term solution. In this section, we first discuss the current status of the IOTA main network implemented through the IOTA reference implementation (IRI) software[3], and then we describe the challenges we are facing when building a *Coo-less* network (i.e., a network without the Coordinator) according to the IOTA white paper [30]. Since the Coordinator and its milestones are currently deeply embedded in IRI, removing those dependencies not only implies comprehensive changes to the software, but also leads to new research questions.

2.1 Current IOTA implementation

The current IOTA main network is implemented according to the IRI software, in which the Coordinator plays an important role. In the following we describe the main tasks implemented in the current main network, not all of them strictly related to consensus:

- *Manual peering.* In order to join the Tangle, a node is required to connect to some existing nodes (*peering*). The current IRI software only permits *manual* peering, i.e., a node has to manually look for the addresses of other Tangle's nodes. Peering is fundamental to propagate transactions and to synchronize to the current status of the ledger. As for the latter, milestones are useful anchors to determine whether two nodes have fallen out of synchronization: If a node's latest solid milestone is much older than its peers, it is probably lagging behind.

- *Rate control mechanism.* In order to issue a transaction, a node must solve a cryptographic puzzle (Proof-of-Work). This is necessary to guarantee that nodes do not spam arbitrarily the network, or to avoid that they inject more transactions than the network can handle.

- *Tip selection strategy.* Approving transactions is a fundamental procedure which leads to the DAG structure of the Tangle. To approve a transaction, a node must verify that no inconsistencies with respect to the ledger state are introduced. Although it is not possible to enforce which transaction to validate, the IOTA white paper suggests a tip selection algorithm based on a random walk which: (i) Discourages lazy behavior and encourages approving fresh tips; (ii) continuously merges small branches into a single

[3] https://github.com/iotaledger/iri

large branch, thus increasing confirmation rate; (iii) in case of conflicts, kills off all but one of the conflicting branches.

- *Consensus.* The main role of a milestones is to determine the consensus. The Tangle applies a simple rule: A transaction is confirmed if and only if it is referenced by a milestone. In IRI, this is reflected in the `getBalances` and `getInclusionStates` API calls, which indicate how many tokens an account has and whether a transaction is confirmed, respectively.

Furthermore, we also want to highlight that milestones are used to optimize the IRI code: For instance, rather than compute the full ledger state starting from the genesis, an intermediate state is saved for each milestone; similarly, milestones are used in *local snapshots*, i.e., the IRI pruning mechanism, which allows nodes to avoid storing older parts of the Tangle.

2.2 Coo-less IOTA network

As a preliminary implementation for a Coo-less network, we are building *CLIRI*, which stands for *Coo-less IRI*. At its core, it is a fork of IRI, with all Coordinator-related components removed. The main purpose of CLIRI is to provide a working testbed for running the first Coo-less IOTA network, on which we could emulate the various Coordicide proposals. This is a necessary first step towards understanding the challenges that a Coo-less main network will one day face.

As discussed above, the Coordinator plays a crucial role in the current IOTA implementation. For this reason, building CLIRI introduces a number of challenges. For its first iteration, we follow the original IOTA white paper [30] when possible, and we choose heuristic algorithms and simplified models otherwise:

- *Ledger validation.* Since rewriting the ledger computation logic without milestones is a significant effort, as a first iteration CLIRI only supports zero-value transactions.
- *Local snapshots.* We remove local snapshots on CLIRI, and we simply discard the entire database automatically at weekly intervals.
- *Random walk starting point.* CLIRI chooses a tip at random, and then backtracks until reaching a transaction "far enough" in the past.

CLIRI is currently at an early development stage, with the first testnet launched on March 5th, 2019.

2.3 New research challenges

Apart from the aforementioned "engineering" choices for CLIRI, its logic is based on the IOTA white paper, and thus it shares the same modeling assumptions. The most significant of these is the *(assiduous) honest transaction majority* condition [8]: Specifically, to be considered valid, the white paper consensus algorithm requires that the majority of transactions *always* come from

honest network participants, i.e., honest actors need to *own* a majority of the hashing power *and* to *constantly produce transactions*. The implication is that honest nodes need to continuously send transactions, regardless of whether they are actually using the network or not. Furthermore, achieving this hashing majority must be expensive, otherwise it would be easy for malicious agents to buy enough hashing power and overtake the network. In addition to this incentivization problem, issuing transactions is subject to Proof-of-Work (PoW). Due to its complexity, slow nodes would be excluded from participating in the network.

The above concerns directly lead to the following research questions which will be investigated throughout the paper:

- *Rate control.* A more efficient rate control algorithm is needed to solve the following tradeoff: If the PoW difficulty is too high, then small devices (e.g., phones or sensors) would take an unreasonably long amount of time to compute it, and will therefore be unable to send transactions; on the other hand, low difficulty can favor network congestion and/or spam attacks.
- *Consensus.* We need a consensus mechanism which is solid under the honest transaction majority assumption without the support of the Coordinator.

In the next section, we will introduce the notion of node identity which is a prerequirement to the solution of the above research topics.

3 Node accountability

In a network without the Coordinator, several applications require to reliably associate transactions or other messages with the node which issued them. These applications include:

- *Rate control.* In an overload scenario, where the nodes are trying to issue more transactions than the overall network can handle, particularly transactions originating from the most heavily contributing nodes should be blocked or penalized.
- *Voting-based consensus mechanisms.* To prevent double voting and to associate votes with node weights, the actual votes must be linked to node IDs.

In Section 3.1, we suggest a way to associate global identities to nodes. Since this may expose the network to potential Sybil attacks, in Section 3.2 we introduce *mana*, a novel anti-Sybil mechanism.

3.1 Global node identities

In order to identify nodes, it is necessary to introduce global node identities. To this end, we envision using common public key cryptography to sign certain data and to link it to its issuing node in a tamper proof way. Additionally, we require that the issuing node adds its public key to every signed message. This way, every node can verify the authenticity of the issuing node without the need for some form of global database of IDs and keys. It is important to note that these mechanisms only need to be implemented to protect the communication layer and that keys, IDs and signatures do not need to be stored in the Tangle once processed by the node. This allows for greater flexibility as the actual signing scheme can be exchanged without any impact on stored data. In contrast to any data stored in the Tangle, the communication layer, therefore, does not necessarily require the use of post-quantum cryptography right now, but it can be swapped when quantum attacks become more imminent in the future.

When node identities are relevant, a distributed system becomes vulnerable to Sybil attacks [16], where a malicious entity masquerades as multiple counterfeit identities. This would overcome any mechanism that relies on a limited number of such identities and would open the network to coordinated attacks. A possible way to deal with this problem is described in the following section.

3.2 Sybil protection

One very common way to make such a Sybil attack harder is the so-called resource testing, where each identity has to prove the ownership of certain difficult-to-obtain resources. Since in the cryptocurrency world users own a certain amount of tokens, we propose a Sybil protection mechanism based on the ownership of such tokens. However, instead of requiring the identity to proof the ownership itself, we allow each user of the network issuing transactions to assign tokens to any node of his choosing. We call these tokens *mana*; they serve as a hard to obtain resource as well as some form of "reputation" which can be assigned to trustworthy nodes. The fundamentals of the actual mechanism are described below:

- When a transaction is issued, it generates a double flow: It (i) transfers data or tokens from one address to another, and (ii) adds virtual tokens (called *mana*) to some nodes. The amount of *mana* corresponds to the tokens transferred.
- The node ID that should receive the *mana* must be specified in the signed part of the transaction. The node gets credited with the *mana* after a certain time. This is necessary to prevent nodes from generating a new ID for every message they issue.
- As soon as the actual tokens are transferred again, the corresponding *mana* is deducted from the previously referenced node, and can potentially be reassigned to a new node.

We stress here that this process does not influence the actual balances in any way, but it is only used to give higher weight to "trusted" nodes.

The amount of *mana* people can delegate is determined by how many tokens they own, which means that people who own more tokens will have a larger influence in this process. In particular, nodes could accumulate large amounts of *mana* without having much stake in the network of their own. In a traditional *proof-of-ownership* Sybil protection mechanism, each node has to prove that it owns a certain amount of collateral. Conversely, delegating *mana* brings several key advantages: As *mana* is credited as part of regular transactions, nodes do not have to constantly use their account's private keys to sign, which would pose a severe security risk; furthermore, this approach does not need incentives for node operators to own or declare a high amount of tokens; finally, users can issue additional *mana* to nodes providing good service to the community.

Since we have now established reliable node identities, we can use these identities to discover and connect to other nodes in the network.

4 Auto Peering

In IOTA, a node is the machine owning all the information about the Tangle. In order for the network to work efficiently, nodes exchange information each other to be kept up-to-date about the new ledger state. Currently, a manual peering process is used for nodes to mutually register as neighbors. However, manual peering might be subject to attacks (e.g., social engineering) to affect the network topology. To prevent these attacks, and to simplify the setup process of new nodes, we introduce a mechanism that allows nodes to choose their neighbors *automatically*. The process of nodes choosing their neighbors without manual intervention by the node operator is called *auto peering*.

Specifically, in this section we propose an auto peering mechanism which achieves two important goals: First, it creates an infrastructure where new nodes can easily join the network; second, we make sure that an attacker cannot target specific nodes during the peering process, i.e., we ensure the network to be secure against Eclipse attacks.

4.1 Peer discovery

Every node chooses its neighbors from a list of potential peering partners. In a permissionless environment, this list changes over time since nodes can continuously join or leave the network. To keep this list up-to-date, we assume that nodes periodically communicate a subset of their known peers with others. This mechanism is simple and effective as it allows every node to learn about other network participants. It is important to note that this mechanism only requires to have access to a large enough fraction of the network such that the list of potential peering partners contains "enough" nodes[4].

[4]The required number of potential peers needed in the list depends on the gossip protocol as well as global system parameters such as the number of neighbors.

4.2 Choosing neighbors

Nodes choose half of their neighbors themselves and let the other half be comprised of neighbors that choose them. The two distinct groups of neighbors are consequently called:

- *Chosen neighbors.* The neighbors that the node proactively choose from his list.
- *Accepted Neighbors.* The neighbors that choose the node as their peer.

In order to select *chosen neighbors* from the list of potential peering partners, we measure the distance between two nodes through the distance function d defined as

$$d(nodeId_1, nodeId_2, \zeta) = hash(nodeId_1 + \zeta) \oplus hash(nodeId_2),$$

where ζ is a public *salt*[5].

In order to connect to new neighbors, each node with ID $ownId$ and public salt ζ keeps a list of potential peers sorted by their distance $d(ownId, \cdot, \zeta)$. Then, the node sends them peering requests in ascending order containing its own node ID, its current public salt and its address (i.e., IP + port). After that, the requested node can decide to either accept or reject the connection as explained below. The connecting node repeats this process until it has established connections to enough neighbors. Those neighbors make up its list of *chosen neighbors*. This entire process is also illustrated in Algorithm 1.

Algorithm 1: Select *chosen neighbors*

Input: desired amount of neighbors k, current list of chosen neighbors $\mathcal{C}$, list of potential peers $\mathcal{P}$

```
P_sorted ← sortByDistanceAsc(P, ownId, ζ)
foreach p ∈ P_sorted do
    peerRequest ← sendPeerRequest(p)
    if peerRequest.accepted then
        append(C, p)
        if |C| > k/2 then
            return
    else
        append(P, peerRequest.proposedCandidates)
        P_sorted ← sortByDistanceAsc(P, ownId, ζ)
```

[5]Salts defend against dictionary attacks or against their hashed equivalent, the precomputed rainbow table attack.

Similarly to the previous case, in order to accept neighbors, every node with ID $ownId$ must generate a *private* salt ζ^*. When it receives a peering request from a node with ID $remoteId$, it measures $d(ownId, remoteId, \zeta^*)$ and only accepts the request if at least one of the following conditions is satisfied:

- The connecting node is closer than an existing *accepted neighbor*.
- The connecting node does not have enough neighbors.

When a node rejects the peering request because it does not match the above requirements, it can use the public salt to propose new potential peers among its list. This is more formally explained in Algorithm 2.

Algorithm 2: Filter *accepted neighbors*

Input: incoming peering request r, desired amount of neighbors k, current list of accepted neighbors $\mathcal{A}$

```
if |A| < k/2 then
    accept(r)
else
    distance_r ← distance(ownId, r.nodeId, ζ*)
    foreach a ∈ A do
        distance_a ← distance(ownId, a.nodeId, ζ*)
        if distance_r < distance_a then
            accept(r)
            drop(a)
            return
    reject(r)
```

4.3 Network reorganization

The public and the private salts help to create an asymmetric perception of the network, which is supposed to discourage an attacker from harming the system. In fact, the only way to target a node in the auto peering process is by brute forcing different node identities and hoping to get closer (in terms of distance d) than an existing neighbor. To prevent brute force attacks from being successful, we let the salts be valid only for a certain amount of time, after which the node updates both its *chosen neighbors* and its *accepted neighbors*[6].

This frequent reorganization brings a twofold benefit: First, it prevents attackers from affecting the network topology; second, it favors new nodes that want to join the network as their peering requests will be accepted with larger probability.

[6] Another approach of reducing an attackers ability to control the network topology is by including a "global source of randomness" when generating the salts.

4.4 Bootstrapping

A new node who wants to join the network initially knows nothing about the network. It neither knows the state of the ledger nor who is currently part of the network. To allow new nodes to get a first list of "other peers", we implement a hard coded list of trusted "entry nodes" that will be run by the IF or trusted community members and that answer to peering requests from new nodes.

This is common practice and is handled this way in virtually all distributed networks.

5 Rate control

A basic goal of every communication network is to handle the traffic injected by its nodes by limiting the rate of transactions joining the network. In fact, such a traffic could lead to unpleasant situations such as network congestion, due to resource limitations, or spam, due to malicious actors:

- *Congestion control.* In most networks, there are circumstances where the incoming traffic load is larger than what the network can handle. If nothing is done to restrict the influx of traffic, bottlenecks can slow down the entire network. A similar analysis can be applied to distributed ledgers, where the incoming traffic (i.e., transactions issued by the nodes of the network) exploits limited resources such as bandwidth, computational power, or disk space. Additionally, nodes can lose synchronization with each other, sometimes without being aware of it.

- *Spam detection.* Gossip protocols (which are currently implemented to forward transactions in the IOTA network) are an efficient and reliable way to disseminate information. These protocols have nevertheless a drawback: They are unable to limit the dissemination of spam messages. Indeed, messages are redundantly distributed in the network and it is enough that a small subset of nodes forward spam messages to have them received by a majority of nodes.

Rate limitation strategies for communication networks are well studied in the context of both congestion control [24] and spam detection [17]. As for DLTs, PoW is a built-in rate limitation mechanism, not only used to reach consensus. However, PoW leads to undesirable side effects such as mining races: The discrepancy between smaller general purpose devices and optimized hardware with respect to the PoW performance is several orders of magnitude. Hence, any rate control based on PoW would eventually leave smaller devices behind. A new transaction rate control mechanism for the Tangle is therefore required to deal with the global and per-node limitations of the network.

5.1 Rate control algorithm

In a pure PoW-based architecture, a high difficulty value would prevent low-power nodes from issuing transactions, which is not desirable, especially in the context of Internet-of-Things; on the other hand, low difficulty can quickly lead to network congestion. We propose an adaptive PoW algorithm to allow every node to issue transactions while penalizing spamming actions.

In our algorithm, when a node decides to issue a transaction, it must solve a cryptographic puzzle where the difficulty is a function of the *mana* owned and of the number of transactions issued recently. Assume node i generates n_i^T transactions in the previous $T \gg h$ time units where h is a bound on the network latency which means that, if a message is sent at time t, then all online nodes will receive the same message within time $t + h$. The same node i has to set the difficulty of the PoW to d_i defined by

$$d_i = d_0 + w(s_i, n_i^T),$$

where d_0 is the basic difficulty, and w is a function that depends on the *mana* s_i and on n_i^T. The time window T, the difficulty d_0 as well as the function w are parameters chosen depending on the fairness level we are aiming for.

As an additional security measure, we require that the total number of transactions issued by a user is limited, i.e.,

$$n_i^T \leq z(s_i), \qquad \forall i, \tag{1}$$

where $z : \mathbb{R}^+ \to \mathbb{R}^+$ is a function that depends on the *mana* s_i such that the larger the *mana* of a node, the higher the number of transactions the same node can issue. The threshold of Eq. (1) ensures that even a user with infinite computational power cannot arbitrarily spam the network.

5.2 Implementation details

For the sake of simplicity, we assume incoming transactions are checked in the same order as they are issued by the sending node. As the expected time needed to perform the PoW is typically much larger than the network latency h, this is a reasonable assumption.

When a transaction is seen for the first time, the node stores the *id* of the node issuing the transaction, the time t_0 at which it is received and the PoW difficulty. The identity *id* of the issuing node as well as its *mana* s_{id} can be determined using the methods described in Section 3. Based on this information, it can then be checked that the number of transactions issued in the recent T time units by the same node does not exceed the allowed maximum $z(s_{id})$ based on its *mana* s_{id} and that the difficulty of the most recent transaction is indeed sufficient. This idea is more formally described in Algorithm 3.

5.3 Verifiable delay functions

While the adaptive rate control algorithm described in this section mitigates some of the drawbacks of the PoW, we believe that, in the current era of dis-

Algorithm 3: Rate control algorithm

Input: incoming transaction t, set of known transactions $\mathcal{X}$, time window T, basic difficulty d_0, weight function w.

Output: forward or ignore t.

$t_0 \leftarrow$ `time(`t`)`
$id \leftarrow$ `nodeId(`t`)`
$\mathcal{T} \leftarrow t' \in \mathcal{X}$ such that `time(`t'`)` $\in (t_0 - T, t_0]$ and `nodeId(`t'`)` $= id$

if $|\mathcal{T}| < z(s_{id})$ **then**
  if `difficulty(`t`)` $\geq d_0 + w(s_{id}, |\mathcal{T}|)$ **then**
    return forward t

return discard t

tributed ledger ecosystems, the need for more efficient algorithms is evident. In the following, we present a more sustainable mechanism that might be used as a replacement of the PoW component: *verifiable delay functions* (VDFs).

Informally, the VDFs are special functions that are (i) difficult to evaluate, even under the assumption of using unbounded parallelism (i.e., using an infinite number of CPUs) [6] and (ii) easy to verify. Various researchers have proposed different VDFs based on specific number-theoretic functions (e.g., modular exponentiation [17, 27], supersingular isogenies over elliptic curves [15], pairings over elliptic curves, injective rational maps between extensions of finite fields [5]). Compared to PoW, these functions bring the following advantages:

- VDFs can be considered more environment friendly since they avoid mining races.
- As they are not parallelizable, they make inefficient the usage of dedicated hardware (e.g., ASIC), inherently solving the problem of unfairness between slow and fast nodes.

The condition for resistance against parallelization is what makes the quest for such functions an interesting and highly non-trivial problem. From an implementation point of view, the main figure of merit is the ratio between the time needed to compute the solution of the function (evaluation) and the time needed to verify its correctness (verification). Table 1 offers good insights on the comparison between different proposed VDFs with respect to this performance metric.

The first ideas about verifiable delay functions can be traced back to the seminal paper of Dwork and Naor in the field of spam protection [17], but it is only after the recent paper by Boneh *et al.* [5] that the interest in the development and implementation of VDFs has substantially increased. In fact, VDFs are already an essential ingredient in some DLT designs (e.g., Chia Network[7]).

[7] www.chia.net

VDF	ratio
Exponentiation-based (RSW)	8000:1
Supersingular isogenies	400:1
Pairings over elliptic curves	300:1
Injective rational maps	n/a

Table 1: Evaluation/verification ratio for different VDFs.

Furthermore, [5] has shown a potential application for decentralized randomness. We believe that VDFs can be of great help in replacing algorithms based on PoW as they are able to restrict the capabilities of nodes with strong hashing power.

6 Consensus

Due to the propagation delay of transactions in the network, not all nodes share the same vision of the Tangle at the same time. This might lead to situations where the validation process lets multiple transactions in conflict with each other join the Tangle. It is a fundamental assumption of the IOTA white paper that the Tangle itself can indeed contain conflicting transactions. In case of a conflict, however, the nodes need to decide which transaction(s) should eventually be considered valid, i.e., they need to come to a *consensus* on those conflicting transactions.

In the IOTA white paper this is solely achieved by consistently applying the tip selection algorithm (TSA), i.e., the mechanism used by (honest) nodes to select the transactions to approve, which currently uses a biased random walk. In case of a conflict, this bias will eventually leave all but one of the conflicting branches behind. However, as already stated throughout this document, this approach is only sufficient under the honest transaction majority assumption. Furthermore, the conflict resolution is slow, which leads to leave transactions that chose the "wrong" branch behind, creating the need for reattachments.

In this work, we present a novel consensus mechanism which integrates a voting system, helping to deal with the aforementioned issues. Whilst the voting models have their limitations, they have been successfully applied in a wide rage of engineering and economical applications [2, 26, 33], leading to the emerging science of sociophysics [10]. For the sake of presentation, we decouple the consensus in two main components (see also Fig. 2):

- *Tip selection algorithm.* In Section 6.1, we present a few important enhancements to the random walk-based algorithm proposed in the IOTA white paper with the objective of increasing the overall throughput and the robustness against parasite chain and splitting attacks.

- *Voting mechanism.* In Section 6.2, we describe two voting mechanisms

where nodes communicate to each other to decide, in case of a conflict, which transaction(s) should be accepted in the Tangle.

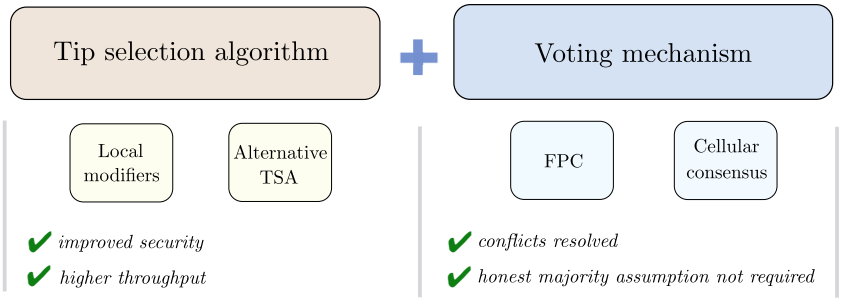


Figure 2: Our novel consensus mechanism adds a voting layer to improve security and to deal with the honest majority assumption.

6.1 Tip selection

The Tangle is a data structure built in accordance with the following rule:

> *In order to join the Tangle, a transaction has to validate two existing transactions.*

The validation of a transaction is a procedure that verifies whether an address owns the tokens spent[8]. If transaction y validates transaction x, we say that y *directly* approves x. Conversely, if there is not a directed edge between the transactions x and y, but there exists a directed path between them, then we say that y *indirectly* approves x.

The IOTA white paper originally uses only a TSA based on a biased random walk to determine the tips to approve. This TSA has a double benefit: First, it is effective against selfish nodes that do simple selections (e.g., at random or at the genesis) to avoid the effort of tip selection; second, it makes the Tangle more resistant against malicious nodes performing, e.g., parasite chain attacks (see Fig. 3). However, the white paper proposal comes with its own limitations as discussed in Section 2; in particular, an honest transaction majority is a necessary condition for the security of the algorithm.

In the rest of this section, we provide an overview of the current status of the research on TSAs: Specifically, in Section 6.1.1 we introduce a random walk-based TSA which aims to improve the original white paper algorithm; then, in Section 6.1.2 we investigate an alternative approach where two different algorithms are used to choose each one of the two tips to approve.

[8]The actual validation process in IOTA is more complex, and we invite the interested reader to visit https://docs.iota.org for more information.

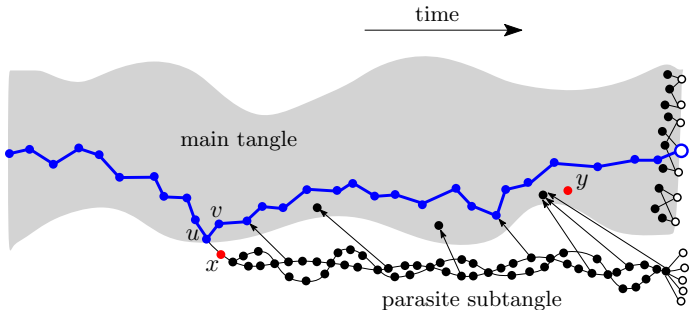


Figure 3: An illustration of the parasite chain attack: The transaction x spends the funds that the attacker wants to spend once more in transaction y, before revealing the subtangle containing x. The grey area contains many transactions which are not shown for sake of clarity. The thick blue line corresponds to a trajectory of a typical random walk.

It is important to stress that, since the TSA is not (and cannot be) enforced, the choice of the particular TSA is, *ultimately*, up to the node's owner. Therefore, the actors will choose their TSAs in a *reasonable* way, as argued in [31]. Because of this intrinsic freedom of choice, and also because the "space" of all possible TSAs is enormous, it is crucial to have all reasonable options on the table. We are absolutely not obliged to be ever content with a particular version of TSA; instead, our vision is that, similarly to the human society itself, the system will continue evolving. Therefore, although the existing random walk-based TSAs are already doing a good job, we will describe below several over promising options and research directions, which would hopefully invite the academic community to provide more valuable input.

6.1.1 Random walk-based TSA

The TSA of the white paper is a biased random walk that starts from the genesis transaction and walks towards the tips. Once the walk has reached a tip, that tip is chosen as the first direct approvee. A second walk is then performed to choose the second direct approvee. In order to prevent the random walk from "lazy" tips choosing to validate old transactions, the transition probability from transaction x to transaction y is biased and proportional to

$$\mathbf{P}[x \to y] \propto \exp\{\alpha \cdot w_y\}, \tag{2}$$

where $\alpha > 0$ is a weight parameter, and w_y is the cumulative weight[9], i.e., the number of transactions that directly or indirectly approves y. In order to obtain the actual probability, one can simply normalize Eq. (2). Since every node sees a different set of tips, it is thus not possible to impose a given TSA.

[9]The definition of cumulative weight in the white paper is slightly more general and also considers the work needed to issue transactions.

In this section, we provide an alternative random walk-based algorithm to improve security and performance. As already specified in the text, although the TSA cannot be enforced, we expect that a node would follow the "best" algorithm available. The main idea is to introduce a local (i.e., per-node) view of the Tangle such that some transactions are preferred based on various kinds of information locally available at the nodes. For instance, the time of solidification[10] could be used to reduce the effectiveness of parasite chain attacks: Since an attacker would require some time to build a subtangle, an honest node could decide to penalize a transaction that appears later than it should. Such a local view is commonly called *local modifiers* [29].

In general, local modifiers enable multiple features: They strengthen the security of the Tangle, especially against parasite chain and splitting attacks; they reduce the dependency on PoW and the cumulative weight calculations; they enable the network to survive a temporary loss of honest transaction majority.

Let t_x (resp. t_y) be the time of solidification of transaction x (resp. transaction y) at a given node. According to the above discussion, the transition probability from transaction x to transaction y becomes proportional to

$$\mathbf{P}[x \to y] \propto \exp\left\{\alpha \cdot w_y - \beta \cdot (t_y - t_x)\right\}, \quad \text{if } t_y - t_x < D, \tag{3}$$

where $\beta > 0$ is a weight parameter, and D is the time difference cutoff parameter. If $t_y - t_x \geq D$, then the transition probability becomes equal to 0, i.e., the corresponding edges are excluded both from the random walk and from the cumulative weight calculation. Basically, we can envision a system where the TSA is performed on a subset of the existing transactions, depending on the local view of the nodes. This idea will be expanded in Section 6.2.

It is interesting to understand how the introduction of the solidification time can increase the robustness of the Tangle against parasite chain and splitting attacks. To perform a parasite chain attack, a malicious actor attaches a hidden subtangle (approving a double spending transaction) after that the original transaction containing the funds has already been accepted, and tries to make it growing. The splitting attack consists of the situation where an agent splits the Tangle in two branches: As one of the branches grows, the agent publishes transactions on the other branch to maintain both alive. The objective is to double spend and damage the network. What both those situations have in common is the fact that transactions are hidden for some time. From a node perspective, this situation looks like a new transaction approving an old one. The solidification time discovers this atypical time difference, and reduces exponentially the probability of such transactions being chosen by the TSA. Apart from the above examples, any attack using hidden transactions or unusual long gap in solidification time will increase its robustness by using local modifiers.

It is also important to mention that Eq. (3) can be easily extended in order to actually model any local information known by the node, such as issuing node reputation. As another natural example of such an extension, consider

[10] The time of solidification of transaction x is the time at which not only the transaction itself, but also all transactions referenced by x have been received by a given node.

two transactions x, y, where y approves x. Let us now define the *sibling number* $s(y, x) \in \mathbf{N}$, in the following way: If $y_0, \ldots, y_k$ are the transactions that approved x directly, and the node heard them in this consecutive order, then we define $s(y_i, u) = i$. Let us consider the following modification of (3):

$$\mathbf{P}[x \to y] \propto \gamma^{s(y,x)} \exp\left\{\alpha \cdot w_y - \beta \cdot (t_y - t_x)\right\}, \quad \text{if } t_y - t_x < D, \tag{4}$$

where $\gamma \in (0, 1]$ is the sibling number importance parameter. In (4), the walker currently located at u would thus give some (additional) preference to those "successors" of u which were seen earlier by the corresponding node. To explain why it is reasonable to adopt a rule of this kind, note first that, *on average*, each transaction will be (directly) approved by two transactions, and so a very high number of the direct approvals can be considered suspicious. That can be indeed a strategy of a malicious actor aimed at "censoring" a virtuous transaction (i.e., preventing it to be chosen by the TSA): By issuing a lot of other transactions which are attached to the same place, the attacker hopes that the TSA would typically select one of "his" transactions rather than that virtuous transaction. Now, observe that the above modification will indeed be efficient in protecting against such a development.

6.1.2 Alternative TSA

The aleatory nature of the random walk-based TSA leaves open the possibility that not all transactions will eventually be validated. In general, this can be considered as a feature since it allows to "leave behind" potentially harmful parts of the Tangle. However, if a legitimate transaction is not validated within a certain delay, one can assume that it would not be validated anymore and it should be reattached to newer tips. Since this introduces an overhead in the system as long as a lower confirmation rate, we are also exploring a novel algorithm which aims to ensure that *all* transactions are approved in finite time [21].

The key intuition is that high values of α (see previous subsection) in the white paper TSA favour longer paths from the genesis to the tips, hence the probability of selecting older tips decreases with time. By contrast, small values of α allow older tips to be selected, but they make the Tangle vulnerable to double spending attacks. The proposed approach aims to combine the best properties of the two scenarios (large and small α) through the use of two different algorithms for selecting each of the tips: The first tip is selected by using the white paper algorithm with a high value of α to guarantee security by ensuring that honest tips get selected preferentially; as for the second tip, we use a random selection to ensure that no tips get left behind by the first, more accurate selection.

Some directions still require to be investigated more carefully. For instance, there is no deterrence against a node that only selects tips at random to reduce its validation overhead. In this case, additional controls may be needed to prevent such lazy behaviors. For example, one possible counter-measure could

use the node accountability feature of Section 3: The participants of the network may somehow penalize nodes which issue transactions that approve "what should not be approved".

6.2 Voting

In this section we discuss a voting mechanism, that is an additional layer to secure the consensus against potential attacks. The idea is that nodes query other nodes about their current opinion of the ledger, and adjust their own opinion over the course of several rounds based on the proportion of other opinions they have observed. For this section, we only consider algorithms that find consensus on the value of a single bit, i.e., of a single conflict. The result of this consensus process can then be used to mark a transaction as either "liked" or "disliked".

The general idea is to let the nodes talk to each other in order to resolve the conflicts *pro-actively*. The conflict resolution is performed starting from an initial opinion on the ledger status described as follows: Consider a transaction v. If in a given interval a node does not see any other transaction spent from the same address, we say that the node *likes* transaction v; otherwise, the node *dislikes* v[11]. The decision whether a transaction is liked or disliked must then be taken into account for the tip selection. The most straightforward way of integrating this in any random walk-based TSA is to simply remove the corresponding edges of disliked transactions and, thus, excluding them and any transaction in their future cone from the tip selection.

After that, we periodically apply a voting scheme to every transaction in the Tangle where each node asks for the opinion of some of its neighbors. After the vote, a transaction is either definitely liked or definitely disliked by a node, and this value will never change. We would like to keep monotonicity in the sense that if a node likes u then it likes any transaction u approves, and if the node dislikes v then it dislikes any transaction that approves v, see Figure 4. To achieve this, we can safely assume that we can only like transaction v when we like all of its past cone, and if we dislike v then we dislike all of its future cone.

In the following two subsections, we will describe two voting mechanisms we are considering. The first one, called *Fast Probabilistic Consensus*, is certified by rigorous mathematical proofs; however, this solution requires nodes to accept connections from nodes which are not neighbors, and uses decentralized randomness, that needs to be acquired as part of an additional layer. On the other hand, the cellular automaton approach of Section 6.2.2 does not have those requirements and seems to be somewhat faster from the first simulations; however, this scheme lacks rigorous proofs and requires a stricter auto peering solution to avoid Eclipse attacks, and formation of "islands" of adversarial nodes. The two solutions can be considered as different non-mutually exclusive implementations of the voting mechanism, and they can be used in combination to build a bullet-proof framework.

[11] It is important to note, that this rule does not include reattachments: If $v_1, \ldots, v_k$ are all reattachments of the same transaction, we either like all or none of them.

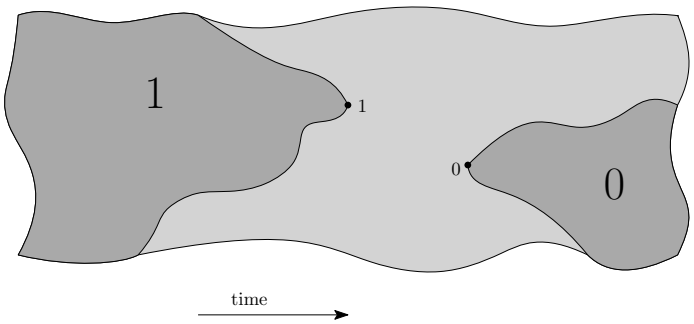


Figure 4: Votes must be consistent

6.2.1 Fast probabilistic consensus

The paper [32] introduces a protocol of low communicational complexity which allows a set of nodes to come to a consensus on a value of a bit by means of (possibly randomized) voting (see e.g. [3, 13, 14, 18, 19, 25] and references therein for the vast available literature on this subject; we mention also the classical work on (probabilistic) Byzantine consensus protocols, see e.g. [1, 4, 7, 11, 20, 22, 34], where, typically, the communicational complexity is much larger). The distinguishing feature of the mechanism described in [32] is that a larger number of *adversarial* (or *Byzantine*) nodes is allowed, which may be a (fixed) proportion of the total number of nodes. Those adversarial nodes intend to either delay the consensus, or break it (i.e., make at least a couple of honest nodes come to different conclusions). It is shown that, nevertheless, the protocol works with high probability when its parameters are suitably chosen, and some explicit estimates on the probability that the protocol finalizes in the consensus state in a given time are also provided (Theorems 2.1 and 2.4).

Differently from the classical work in this area, it is not required that the consensus should be achieved, with high probability, on the initial majority value. Rather,

- if, initially, no *significant majority*[12] of nodes prefer 1, then the final consensus will be on the value 0 with high probability;
- if, initially, a *supermajority*[13] of nodes prefer 1, then the final consensus will be 1 with high probability.

To explain why this is relevant in cryptocurrency applications, consider a situation when there are two contradicting transactions; for example, one of them transfers all the balance of address A_1 to address A_2, while the other transfers all

[12] Loosely speaking, a significant majority is something statistically different from the 50/50 situation; for example, the proportion of 1-opinion is greater than ϕ for some fixed $\phi > 1/2$.

[13] Again, this is a loosely defined notion; a supermajority is something already *close* to consensus, e.g., more than 90% of all nodes have the same opinion.

the balance of address A_1 to address $A_3 \neq A_2$. In the case when neither of the two transactions is strongly preferred by the nodes of the network, by declaring both invalid we are on the safe side. On the other hand, it would not be a good idea to *always* declare them invalid. Indeed, if we do this, then a malicious actor would be able to exploit it in the following way: First, he places a legitimate transaction, e.g., to buy some goods from a merchant. When he receives the goods, he publishes a double-spending transaction as above in the hope that *both* would be canceled by the system, and so he would effectively receive his money back (or at least take the money away from the merchant). To avoid this kind of development, it would be desirable if the first transaction (payment to the merchant) which, by that time, have probably gained some confidence from the nodes, would stay confirmed, and only the subsequent double-spend gets canceled.

A special feature of the protocol of [32] is that it makes use of a sequence of random numbers which are either provided by a trusted source or generated by the nodes themselves using some decentralized random number generating protocol, see e.g. [9, 28, 35, 36]. It is important to observe that, even if from time to time the adversary can get (total or partial) control of the random number, this can only lead to delayed consensus, but he cannot convince different honest nodes of different things, i.e., safety is not violated. Also, it is not necessary that really *all* honest nodes agree on the same number; if most of them do, this is already enough for the protocol (that is, the specific task of random number generation does not require any sort of "strong consensus").

We do not describe all the details of the proposed protocol here, and refer the interested reader to [32] instead.

6.2.2 Cellular consensus

The second implementation discussed in this paper is *cellular automata* (CA). The use of a CA approach has originally been developed as a formal description of the Ising model [23]. One of the biggest advantages of the CA-based techniques over other consensus algorithms is the opportunity to achieve a very high level of parallelism. This advantage alone is a sufficient incentive for deeper studies. CA brings the following novel key properties:

- Every node acts as a cellular automaton [12] that, in the presence of conflicts, changes its opinion only based on the state of its direct neighbors and always adopting the majority opinion.
- The set of neighbors of a node does not change during one run of the consensus algorithm. In this case, the reorganization mentioned in Section 4 must only happen for different runs, i.e., different conflicts.
- When evaluating the opinions of neighbors, nodes will require a "proof" that includes the opinions of the neighbors' neighbors. This will allow nodes to monitor each others' behavior and prevents a node from lying independently of its neighbors.

- Misbehaving neighbors, i.e., neighbors that hold an opinion that is inconsistent with this proof, will be dropped immediately. This information is then also broadcasted to the network for other nodes to verify and mark that corresponding node as malicious and prevent future connection attempts.

At the beginning of each round, every node sends a "heartbeat" of its current status. This includes its signed current opinion, as well as the opinions of each of its neighbors from the previous round, each signed by the issuing node. Since the previous opinions of the neighbors cannot be faked, every node receiving this heartbeat can validate that the current opinion is indeed correct and follows the rules of the consensus mechanism.

We formalize the above ideas in the consensus protocol described in the next subsection.

Algorithm 4: Send heartbeat

Function `heartbeat`(*Node i, Round m*)**:**
  foreach *neighbor* $j \in N_i$ **do**
    send opinion $X_m(i)$ to neighbor j
    foreach $j' \in N_i \setminus \{j\}$ **do**
      send opinion $X_{m-1}(j')$ to neighbor j

Model Suppose that there is a network composed of n nodes, and these nodes need to come to a consensus on the value of a bit. For clarity, we assume in the following that each node is directly connected to k neighbors, through the auto peering mechanism described in Section 4. The set of neighbors of node i is denoted by N_i. The auto peering mechanism is a key factor for the security of this approach: Indeed, a malicious node should not be able to select or influence its neighbors.

During each stage of the algorithm, each node holds an opinion on the value of the bit. The opinion can be either 0, 1 or $\perp$, depending on whether the node prefers 0, 1 or none at all. The opinion of node i in round m is denoted by $X_m(i) \in \{0, 1, \perp\}$. We further assume, that each node i has the initial opinion $X_0(i) \in \{0, 1\}$.

The protocol depends on the following parameters:

- $k \in \mathbb{N}$, number of (initial) neighbors of each node.
- $M \in \mathbb{N}$, maximum number of rounds.
- $\ell \in \mathbb{N}$, the number of consecutive rounds with the same opinion after which it becomes final.
- $p : \{0, \ldots, k\} \to \mathbb{R}_{\geq 0}$, monotonically increasing weight function that maps the number of neighbors to a weight. This penalizes nodes having fewer than k neighbors.

Algorithm 5: Cellular consensus

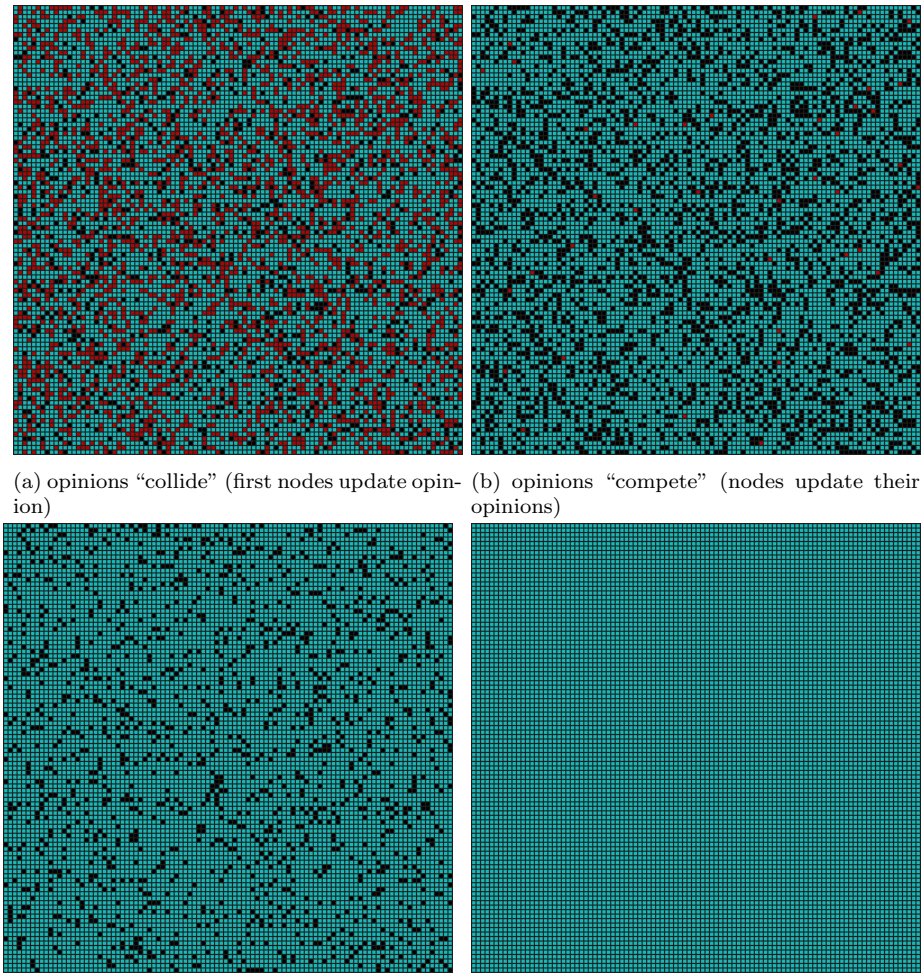
```
foreach node i do
    Send initial opinion X_0(i) to neighbors N_i

for m ← 1 to M do
    foreach node i do
        foreach neighbor j ∈ N_i do
            if X_{m-1}(j) is inconsistent wrt X_{m'}(j') for
              j' ∈ N_j, m' < m − 1 then
                // drop neighbor j
                N_i ← N_i \ {j}
        if node i finalized then
            X_m(i) ← X_{m-1}(i)
        else
            // adopt majority opinion
            total ← Σ_{j∈N_i} p(|N_j|)
            if Σ_{j∈N_i | X_{m-1}(j)=0} p(|N_j|) > total/2 then
                X_m(i) ← 0
            else if Σ_{j∈N_i | X_{m-1}(j)=1} p(|N_j|) > total/2 then
                X_m(i) ← 1
            else
                X_m(i) ← ⊥                          // cancel all
        heartbeat(i, m)
        if opinion X(i) did not change in the last ℓ rounds then
            mark node i finalized
```

Algorithm Each node i knows the opinions of its neighbors $j \in N_i$ as well as the opinions of all their neighbors N_j. This is assured by a broadcasting step where all the opinions are signed in such a way that the originating nodes as well as broadcasting node are unforgeable. This is formalized in Algorithm 4.

The consensus mechanism is a CA where a node uses the opinions of its neighbors to update its own state. When the majority of neighbors support either 0 or 1, the node adopts that opinion. If none of these opinions has a majority, it adopts $\perp$, i.e., none of them. As long as we assume that the set N_i is known at least for all of its neighbors i, any node can use these simple rules to validate whether the reported opinion of neighbor i is consistent with the opinions of all nodes in N_i. The overall consensus mechanism is more formally illustrated in Algorithm 5 and an illustration of the CA process for an example scenario can be found in Fig. 5.

7 Conclusion

In this paper, we outline our approach for the Coordicide project. In particular, we describe our main ideas around the consensus mechanism, security and protection against attacks, spam control and auto peering; all of which provide the building blocks that are cruicial for the Coordicide project. Although our proposal towards the path to Coordicide is now well-defined, we are currently evaluating various options and settings to implement and deploy these components together in a holistic framework.



(a) opinions "collide" (first nodes update opinion)

(b) opinions "compete" (nodes update their opinions)

(c) opinions "compete" (nodes update their opinions)

(d) a single opinion "survives" (network reaches consensus)

Figure 5: Visualization of the CA consensus process: Each square corresponds to a node connected to random neighbors. Initially, the two conflicting transactions are propagated through the network. Then, nodes are consistently adapting their opinions (0: red, 1: cyan, $\perp$: black) before eventually coming to consensus.

References

[1] Marcos K. Aguilera and Sam Toueg. The Correctness Proof of Ben-Or's Randomized Consensus Algorithm. *Distributed Computing*, pages 371–381, 2012.

[2] Sven Banisch, Tanya Araújo, and Jorge Louçã. Opinion dynamics and communication networks. *Advances in Complex Systems*, pages 95–111, 2010.

[3] Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Luca Trevisan. Stabilizing Consensus with Many Opinions. In *Symposium on Discrete algorithms*, pages 620–635. SIAM, 2016.

[4] Michael Ben-Or. Another Advantage of Free Choice (Extended Abstract): Completely Asynchronous Agreement Protocols. In *ACM Symposium on Principles of Distributed Computing*, pages 27–30, 1983.

[5] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable Delay Functions. In *Annual International Cryptology Conference*, pages 757–788. Springer, 2018.

[6] Allan Borodin and Ian Munro. *The Computational Complexity of Algebraic and Numeric Problems*. North-Holland, 1975.

[7] Gabriel Bracha. Asynchronous Byzantine Agreement Protocols. *Information and Computation*, pages 130 – 143, 1987.

[8] Quentin Bramas. The Stability and the Security of the Tangle. Preprint, 2018.

[9] Ignacio Cascudo and Bernardo David. SCRAPE: Scalable Randomness Attested by Public Entities. In *International Conference on Applied Cryptography and Network Security*, pages 537–556. Springer, 2017.

[10] Claudio Castellano, Santo Fortunato, and Vittorio Loreto. Statistical physics of social dynamics. *Reviews of Modern Physics*, page 591, 2009.

[11] Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Transactions on Computer Systems*, pages 398–461, 2002.

[12] Edgar F. Codd. *Cellular Automata*. Academic Press, 1968.

[13] Colin Cooper, Robert Elsässer, and Tomasz Radzik. The Power of Two Choices in Distributed Voting. In *International Colloquium on Automata, Languages, and Programming*, pages 435–446. Springer, 2014.

[14] Colin Cooper, Robert Elsässer, Tomasz Radzik, Nicolas Rivera, and Takeharu Shiraga. Fast Consensus for Voting on General Expander Graphs. In *International Symposium on Distributed Computing*, pages 248–262. Springer, 2015.

[15] Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. Verifiable Delay Functions from Supersingular Isogenies and Pairings. Cryptology ePrint Archive, Report 2019/166, 2019., 2019.

[16] John R. Douceur. The Sybil Attack. In *International Workshop on Peer-to-peer Systems*, pages 251–260. Springer, 2002.

[17] Cynthia Dwork and Moni Naor. Pricing via Processing or Combatting Junk Mail. In *Advances in Cryptology*, pages 139–147, 1993.

[18] Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Frederik Mallmann-Trenn, and Horst Trinker. Rapid Asynchronous Plurality Consensus. *arXiv preprint arXiv:1602.04667*, 2016.

[19] Giulia Fanti, Nina Holden, Yuval Peres, and Gireeja Ranade. Communication Cost of Consensus for Nodes with Limited Memory. *arXiv preprint arXiv:1901.01665*, 2019.

[20] Paul Feldman and Silvio Micali. An Optimal Probabilistic Algorithm for Synchronous Byzantine Agreement. In *Automata, Languages and Programming*, pages 341–378. Springer, 1989.

[21] Pietro Ferraro, Christopher K. King, and Robert Shorten. IOTA-Based Directed Acyclic Graphs without Orphans. *arXiv preprint arXiv:1901.07302*, 2019.

[22] Roy Friedman, Achour Mostéfaoui, and Michel Raynal. Simple and Efficient Oracle-Based Consensus Protocols for Asynchronous Byzantine Systems. *IEEE International Symposium on Reliable Distributed Systems*, pages 228–237, 2004.

[23] Ernst Ising. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Physik A Hadrons and Nuclei*, pages 253–258, 1925.

[24] Frank P. Kelly, Akhil K. Maulloo, and David K. H. Tan. Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability. *Journal of the Operational Research Society*, pages 237–252, 1998.

[25] Frederik Mallmann-Trenn. *Probabilistic Analysis of Distributed Processes with Focus on Consensus*. PhD thesis, PSL Research University, 2017.

[26] Hong-Li Niu and Jun Wang. Entropy and recurrence measures of a financial dynamic system by an interacting voter system. *Entropy*, pages 2590–2605, 2015.

[27] Krzysztof Pietrzak. Simple Verifiable Delay Functions. In *Innovations in Theoretical Computer Science Conference*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[28] Serguei Popov. On a Decentralized Trustless Pseudo-Random Number Generation Algorithm. *Journal of Mathematical Cryptology*, pages 37–43, 2017.

[29] Serguei Popov. Local Modifiers in the Tangle. 2018.

[30] Serguei Popov. The Tangle, 2018.

[31] Serguei Popov. IOTA: Feeless and Free. *IEEE Blockchain Technical Briefs*, 2019.

[32] Serguei Popov. On Fast Probabilistic Consensus in the Byzantine Setting. Preprint, 2019.

[33] Piotr Przybyła, Katarzyna Sznajd-Weron, and Maciej Tabiszewski. Exit probability in a one-dimensional nonlinear q-voter model. *Phys. Rev. E*, 2011.

[34] Michael O. Rabin. Randomized Byzantine Generals. In *Annual Symposium on Foundations of Computer Science*, pages 403–409. IEEE, 1983.

[35] Philipp Schindler, Nicholas Stifter, Aljosha Judmayer, and Edgar Weippl. HydRand: Practical Continuous Distributed Randomness. Cryptology ePrint Archive, Report 2018/319, 2018.

[36] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J. Fischer, and Bryan Ford. Scalable Bias-Resistant Distributed Randomness. In *IEEE Symposium on Security and Privacy*, pages 444–460, 2017.