DAG-Based Distributed Ledgers

Research Seminar in Cryptology

Janno Siim

June 1, 2018

Project advisor: Ahto Truu

1 Introduction

In 2008 Bitcoin [Nak09] was introduced as the first decentralized digital currency. Its core underlying technology is the *blockchain* which is essentially a distributed append-only database. In particular, blockchain solves the key issue in decentralized digital currencies – *the double spending problem* – which asks: *"if there is no central authority, what stops a malicious party from spending the same unit of currency multiple times"*. Blockchain solves this problem by keeping track of each transaction that has been ever made while being robust against adversarial modifications.

Besides digital currencies, blockchain has been proposed as a solution for many other problems. In short, it is potentially useful for any application where it should not be possible to modify data once it is stored (append-only storage systems). Some projects, such as Ethereum [1], go even a step further from plain data storage and allow to run so-called *smart contracts*, which are programs running on top of blockchain. In this case blockchain guarantees that a program is executed correctly in a distributed and untrusted setting.

In more detail, Bitcoin blockchain works as follows. Transaction are stored in *blocks* that are linked in a chain through hashing, that is, each block contains a hash of the previous block. Blocks themselves are generated by *miners* that collect and verify users' transactions. Transaction fees and block discovery rewards act as incentive for miners to participate.

Security of the Bitcoin blockchain comes from *proof-of-work (PoW)* puzzle solving – block is considered to be valid if it contains a solution to a computationally difficult puzzle. More precisely, miners needs to include a *nonce* (a small bitstring) to the block such that the hash of the block begins with a certain number of zeros. Finding such nonces is believed to be computationally difficult task.

Honest nodes always extend the longest valid chain; if an attacker wishes to modify a block, then it needs to construct a longer valid chain in order to convince honest nodes. Garay *et al.* [GKL17] show that Bitcoin blockchain realizes security definition of a *robust transaction ledger* given that adversary controls less than majority of the hashing power. In the following, we will also informally refer to blockchain and other analogues systems as *transaction ledgers*.

Problems with Bitcoin. Although Bitcoin blockchain is sufficient to implement a distributed digital currency, still a number of problems inhibit its wide-scale adoption. We mention a few of the more prominent issues:

[1] `https://www.ethereum.org/`

1. Scalability – As of 2018, Bitcoin has an average throughput of 7 transaction per second, whereas, for example, Visa credit card system processes 2000 transaction per second [CDE+16]. This means that Bitcoin cannot handle massive world-wide adoption.
2. Transaction confirmation – Mining a Bitcoin block (and hence getting a transaction confirmation) takes approximately 10 minutes [2]. For many real-life transactions, such as buying food from a supermarket, this is too long.
3. Transaction fees – Bitcoin transaction fees are high enough that it is unreasonable to make micro-payments.
4. Energy consumption – It has been estimated that the energy consumption of Bitcoin is comparable to energy consumption of Ireland [OM14].
5. Privacy – Although Bitcoin accounts are not necessarily associated with the owner's real-life identity, since every transaction is public, it is often quite easy to deanonymize accounts through association (e.g., same account bought coffee at place X at time Y and later also made a transaction to a known drug dealer).
6. Centralization – Bitcoin is not as decentralized as was originally hoped. Large groups of miners have decided to work together in *mining pools* which allows them to share the rewards and thus guarantees a more steady income. Moreover, as of now most of the hashing power has concentrated to China [3] opening up the possibility that Chinese government can significantly influence the behavior of miners.
7. Usability – It is difficult for an average user to store a secret key in convenient and secure manner. If the key is lost or stolen, then, distinct from traditional banking systems, there is no central authority that can help the user by closing the account or resetting the key.

Alternative ledgers. A myriad of alternative blockchains and cryptocurrencies have been proposed that try to address those issues. For example, problem of energy consumption is tackled by Cardano[4] cryptocurrency [KRDO17] by using a proof-of-stake type blockchain rather than the PoW blockchain used in Bitcoin; Zcash[5] [MGGR13] solves the privacy problem by introducing zero-knowledge proofs to blockchain.

In this report we review two distributed ledgers that claim to solve many of the above mentioned issues. These systems are IOTA[6] and Swirlds[7]. Common feature among them is the use of *directed acyclic graphs (DAGs)* as part of the consensus mechanism, although in quite different ways. We use white papers [Pop17, Bai16] and public documentation [IOT] as the main source of information.

For sake of completeness we also mention some other DAG-like distributed ledgers:

- Sompolinsky and Zohar et al. have studied tree-like and DAG-like blockchains in a series of papers [SZ15, LSZ15, SLZ16, SZ18].
- ByteBall [8] is a cryptocurrency that uses DAGs in quite similar way to IOTA.
- Dagcoin[9] is an Estonian DAG-based cryptocurrency. Some sources claim it to be a fraud[10].

[2] To be confident that transaction is included, several more blocks should be mined. Hence, it might actually take an hour.
[3] `https://www.buybitcoinworldwide.com/mining/pools/`
[4] `https://www.cardano.org/en/home/`
[5] `https://z.cash/`
[6] `https://iota.org/`
[7] `https://www.swirlds.com/`
[8] `https://byteball.org/`
[9] `https://dagcoin.org/`
[10] See `https://geenius.ee/uudis/tuhjad-lubadused-eesti-oma-kruptorahaga-kaasa-lainud-investorid-vihaseks-ajanud/`, `https://ethanvanderbuilt.com/2018/01/15/dagcoin-scam-yes-opinion/`

2 IOTA

IOTA is a DAG-based cryptocurrency designed for *Internet-of-Things (IoT)* devices and it claims to solve many of the problems in existing distributed ledgers [IOT]:

1. Scalability – It allows many nodes to verify transactions in parallel.
2. Transaction fees – It has none.
3. Decentralization – All users validate transaction on equal grounds and there is no mining. Hence, we should not expect to see the same issue of concentration of power as in Bitcoin.
4. Quantum resistance – They claim that IOTA is secure against quantum adversaries.

Instead of a linear blockchain, this system uses a DAG data structure called *tangle* where vertices that are called *sites* correspond to transactions and edges correspond to transaction approvals. More precisely, if user issues a transaction, as a new site u, then it must pick two existing sites v, w and approve transactions at those sites. This corresponds to adding directed edges (u, v) and (u, w) to the tangle. Finally, when the node has approved two transaction, then it will also solve a small PoW puzzle (3^8 hash function evaluations on average) similar to Bitcoin. The precise approval procedure seems to be unspecified in the white paper. Initial vertex in the tangle is called *genesis* and it contains all the currency in the system.

Yet to be proved sites in the tangle are called *tips*. Nodes are allowed to choose any sites for verification, but honest nodes follow the tip selection algorithm which roughly works as follows: node does a weighted random walk, starting from the genesis, until it reaches a tip. Weight of the site depends on the number of approvals that it previously received.

Currently the system contains a *coordinator* node, run by IOTA Foundation, that confirms the transactions. More precisely, the coordinator issues a *milestone* transaction every few minutes and all the transaction it references are considered to be confirmed. Transactions that are not referenced by any milestone are not considered to be confirmed. However, coordinator is supposed to be only a temporary solution and eventually IOTA should work in a fully distributed manner.

White paper [Pop17] describes a Markov-chain Monte-Carlo algorithm which can probabilistically check the transaction confirmation (without a coordinator). It works by running the tip selection algorithm n times while checking how many of the selected tips reference a given transaction. If κ out of n tips reference the transaction, then it is confirmed with confidence κ/n.

To create an IOTA account the user generates a private seed from which the private key and public address are derived. Since IOTA uses, supposedly quantum-secure, Winternitz one-time signatures [Mer90], then each private key (and address) can only be used once. Transactions in IOTA are called *bundles* that (among other details) contain signed input addresses and output address of the recipient.

It should be also mentioned that IOTA uses ternary system rather than binary. To the best of our knowledge, there is no official IOTA documentation explaining this design choice. However, one of the founders of IOTA has claimed on Reddit[11] that processors can run more efficiently on ternary system. Of course, since currently no such processors are widely available, then there has to be conversion to binary which likely makes it less efficient than a native binary system, not to mention the added complexity of designing everything in a new numeral system.

IOTA is open-source [12] except for the coordinator node that is kept closed-source, supposedly for copy-protection reasons. IOTA is not patented.

[11] `https://www.reddit.com/r/ethereum/comments/696iln/when_is_ethereum_going_to_run_in_to_serious/dh4jtgy/`
[12] `https://github.com/iotaledger`

2.1 Security

Next we try to understand the security guarantees of the tangle by following the IOTA white paper [Pop17].

Firstly, the white paper does not state any kind of security definition that the tangle should satisfy. In Section 3 they do identify rate of tip growth as an important problem to study. However, this analysis is done under assumption that do not seem realistic in practice.

Here are some of the assumptions made in [Pop17]:

- Transactions are issued by a large number of roughly independent entities.
- [Transaction] rate remains constant in time.
- Any node, at the moment when it issues a transaction, observes not the actual state of the tangle, but the one exactly h time units ago.
- All devices have approximately the same computing power.

Taking the last assumption as an example: it seems unjustified to assume that all devices have the same computing power. Since the main application of IOTA are IoT devices, then most of the devices will have low computing power. Then it seems quite easy for an attacker to find a computing device with a significantly higher computing power and connect it to the permissionless IOTA network.

Section 4 of the white paper considers some concrete attacks and the ways to overcome them. However, solutions to those attacks seem unconvincing.

In particular, it remains unclear what stops a computationally powerful adversary from conducting a double spending attack as follows: adversary can use its coin and wait until it receives the goods, then it can make a second transaction with the same coin and confirm it by making many small transaction, perhaps under multiple different accounts. Attacks of this type where adversary floods the systems with new accounts are sometimes called Sybil attacks.

Other consensus algorithms have a clear answer to this problem. In PoW blockchains this is impossible since adversary should have more computational power than rest of the system. Similarly in proof of stake blockchains adversary should obtain majority of stake (currency) to perform this attack. Permissioned consensus protocols simply restrict the adversary from adding new nodes to the system.

To the best of our understanding, IOTA team claims that it is the small amount of PoW (3^8 hashings on average) that stops such an attack[13].

2.2 Criticism

IOTA has attracted criticism from several different directions.

Curl vulnerability. In 2017, Heilman *et al.* [HNDV17] discovered a vulnerability of the in-house hash function *Curl* used in IOTA. Using differential cryptanalysis they were able to find hash collisions on commodity hardware under just a few minutes[14]. Under specific circumstances this attack even allowed to steal funds. Since then IOTA team has updated the hash function to *Kerl*[15] which is based on *Keccak* hash function (SHA-3). Currently no similar attacks are known.

[13] `http://www.tangleblog.com/2017/07/10/is-double-spending-possible-with-iota/` actually leaves the impression that double spending is theoretically possible, but just technically difficult to do in practice.

[14] `https://medium.com/@neha/cryptographic-vulnerabilities-in-iota-9a6a9ddc4367`

[15] `https://github.com/iotaledger/kerl/blob/master/IOTA-Kerl-spec.md`

Coordinator. As of beginning of 2018, IOTA relies on centralized coordinator, making it (at minimum) a single point of failure. IOTA team claims that as the networks grows, they are able to permanently turn off the coordinator. Others have expressed doubt that IOTA will ever reach that phase. (See [Wal17] and comments below for one such debate.) Source code of the coordinator node is not publicly available, as was mentioned above.

Replay vulnerability. Recently a new potential vulnerability was discovered by Joseph Rebstock [Reb17]. After receiving funds through a transaction, adversary can resend the same transaction multiple times and drain the address. However, this attack relies on more funds being on the address than is needed for a single transaction. Using the same address for multiple transactions is discouraged by the IOTA team and therefore under normal circumstances the address would be empty once a transaction is made. As such, IOTA team does not consider this to be a legitimate vulnerability and has no intentions of changing the protocol behavior[16].

3 Swirlds

Swirlds is a platform meant for creating distributed applications in a potentially malicious environment. Core component of Swirlds is the hashgraph consensus algorithm [Bai16]. This allows nodes to collectively agree on the order of an evergrowing sequence of transactions. Such an approach, which is called *state machine replication* [Sch90], can be used to implement cryptocurrencies or – more generally – robust distributed applications i.e. smart contracts.

By default, Swirlds is a permissioned distributed ledger in the sense that not anyone can freely join the system, although the white paper informally discusses extensions to the permissionless setting using proof of stake. Correspondingly, marketing of Swirlds is mainly oriented towards businesses, rather than individuals, which is more compatible with the permissioned setting.

Swirlds answers the problems of Bitcoin as follows:

1. Scalability – It removes proof of work.
2. Decentralization – Each node in the system has equal voting rights. Although, since it is foremost a permissioned ledger, then some amount of centralization is inherent.

We follow the white paper [Bai16] and only focus on the novel hashgraph consensus algorithm leaving aside other details of Swirlds.

Permissioned setting makes the hashgraph consensus more similar to classical Byzantine consensus protocols [BO83, DLS88, Lam98, CL99] than the recent blockchain protocols. Namely, such protocols rely on each node having knowledge of all other nodes in the system. The white paper does not directly say or analyze whether hashgraph consensus protocol allows to add nodes after the beginning of the protocol, but it seems to be implied that it is possible. Otherwise proof of stake approach would not be meaningful. However, as this setting is not analyzed in the white paper, then we leave it out from the protocol description.

We also note that there are a number of patents on the hashgraph consensus algorithm[17], potentially limiting the use for outsiders. At the time of writing, source-code for hashgraph protocol is not available, but they have promised to make it available in the future [18].

[16] `https://www.reddit.com/r/CryptoCurrency/comments/7yw5py/replay_attacks_in_iota_new_vulnerability_report/dujpzk5/`

[17] `https://www.swirlds.com/ip/`

[18] `https://www.hederahashgraph.com/faq#is-the-source-code-closed-source`

3.1 Setting and Assumptions

Hashgraph consensus protocol is executed by a fixed set of N nodes $\mathcal{P} := \{\mathsf{P}_1, \ldots, \mathsf{P}_N\}$. Security of the protocol requires the following assumptions:

- Less than $N/3$ of the nodes are malicious.[19]
- There is an existentially unforgeable signature scheme and a collision resistant hash function.
- Public key infrastructure to bind each node to a specific public key of the signature scheme.
- Asynchronous communication channel meaning that all messages will be eventually delivered to the recipient, but this might take unspecified amount of time which is controlled by the adversary. Note that this is a very weak requirement from the channel. Many protocols require that each message is delivered in a known time-bound Δ.

Nodes in the system can issue transactions and the hashgraph consensus protocol will guarantee that eventually all honest nodes **assign** and **agree** on a fixed position for each transaction in the sequence of all transactions.

Additionally, they claim the property of *fairness* which roughly means that if a transaction tx_A reaches at least $2N/3$ of nodes before some transaction tx_B, then tx_A gets a lower position in sequence of transaction than tx_B. This should make it difficult for malicious nodes to order transactions in their favour. For instance, in an auction application it might be crucial to get a transaction accepted first.

3.2 Protocol Description

Hashgraph consensus algorithm stores transactions in *(gossip) events*. Let h be a collision resistant hash function. An event created by node P_i is defined as a tuple $E = (\mathsf{tx}, \mathsf{ts}, h(E_c), h(E_r))$ where tx is transaction data, ts is a timestamp, E_c is the previous event that P_i created, and E_r is the previous event that P_i received. We allow E_r (resp. E_c) to be a special $\perp$ value if so far P_i has not received (resp. created) any events. Later the $\perp$ symbol is also used for some other variables to indicate that the value is unspecified. Gossip event is signed by its creator P_i.

We call a set of events $\mathcal{H}$ a *hashgraph* if for each $E := (\mathsf{tx}, \mathsf{ts}, h_c, h_r) \in \mathcal{H}$ exists $E_c, E_r \in \mathcal{H} \cup \{\perp\}$ such that $h_c = h(E_c)$ and $h_r = h(E_r)$. Connecting gossip events with directed edges, that is, there is an edge from E_c and E_r to E, forms a directed graph as can be seen in Figure 1.

The basis of the Hashgraph consensus algorithm is a simple *gossip* protocol where each node $\mathsf{P}_i \in \mathcal{P}$ picks a random party P_j from $\mathcal{P} \setminus \{\mathsf{P}_i\}$ and sends to P_j all the events that it knows. This process is constantly repeated. Upon receiving a gossip data, node records it by creating a new event E that additionally contains new transaction data.

Intuitively hashgraph data structure has some very convenient properties for the consensus algorithm. For example, if two honest nodes both have an event $E \in \mathcal{H}$, then properties of hash function guarantee that both nodes agree on the ancestors of E.

Basic terminology. In order to describe the consensus protocol, we introduce some basic notions related to events and hashgraphs.

We say that an event E^* is a *parent* of event E if there is a directed edge from E^* to E. More generally, an event E^* is an *ancestor* of event E if there is a directed path from E^* to E. By convention each event is considered to be its own ancestor. Event E^* is a *self-ancestor* of event E if E^* is an ancestor of E and

[19] Clearly, in a permissionless setting adversary could cross the $N/3$ bound.

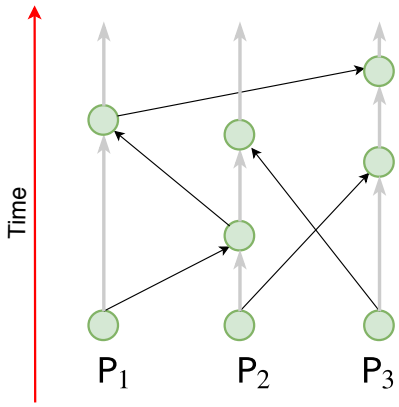

Figure 1: Hashgraph between parties P_1, P_2, and P_3

both E^* and E are created by the same node. We denote the set of parents, ancestors, and self-ancestors respectively by $\delta^-(E)$, $\delta^-_a(E)$, and $\delta^-_{sa}(E)$.

Event E can *see* E^* created by some node P if

- E^* is an ancestor of E, and
- for any ancestor E' of E, created by P, either E' is an ancestor of E^* or E^* is an ancestor of E'.

In other words, E cannot see ancestors created by P if P has created two ancestors such that neither is each others ancestor, i.e., P has created a *fork*.

Event E can *strongly see* E^* if E can see more than $2N/3$ events from distinct nodes such that each of those events can see E^*.

We note that all of the above definitions hold respect to some fixed hashgraph $\mathcal{H}$. Correct would be to say that (for example) E can *see* E^* in hashgraph $\mathcal{H}$ and so on, but we left that out for simplicity.

As will be explained later in more detail, consensus algorithm divides all events into *rounds* and the first event that a party creates in each round is called a *witness*. Moreover, witness can be *famous* if it gets many votes from witnesses of the next round. Final order of events is indicated by *round received* number, not to be confused by round number. We introduce notation for these and other notions associated to an event E (i) the round number $E.\mathsf{rnd}$, (ii) the boolean $E.\mathsf{w}$ indicating whether E is a witness, (iii) the boolean $E.\mathsf{f}$ indicating whether E is famous, (iv) the boolean vote $E.\mathsf{v}_{E^*}$ of E for E^*, (v) the self-parent $E.\mathsf{sp}$, (vi) the signature $E.\mathsf{sig}$, (vii) the round received number $E.\mathsf{rndr}$, (viii) the timestamp ts. Default value for all of the above variables is $\perp$.

Subprocedures. Consensus algorithm uses a number of subprocedures:

- $\mathsf{filter}(\mathcal{H})$ – filters out events from hashgraph $\mathcal{H}$ that do not have valid signatures or have hashes linking to unknown events.
- $\mathsf{gossip}(\mathcal{H})$ – sends a set of events $\mathcal{H}$ to a random peer as was described above.
- $\mathsf{topSort}(\mathcal{H})$ – outputs a *topological order* of the set of events $\mathcal{H}$, i.e., orders the events so that all directed edges point to the right.

- $\mathsf{see}(E, E', \mathcal{H})$ and $\mathsf{strongSee}(E, E', \mathcal{H})$ – outputs True if E can (strongly) see E' in hashgraph $\mathcal{H}$ and False otherwise.
- $\mathsf{uFam}(\mathcal{H}, r)$ – returns a set of famous witnesses of round r such that each node has at most one famous witness (unique famous witness). We explain the meaning of rounds and being famous in the next paragraph.

Consensus algorithm. Consensus protocol in full detail can be seen in Figure 2. It contains algorithms main, decideOrder, divideRounds, and decideFame that we refer to in the following.

Each node executes the main algorithm which runs two loops in parallel: first loop constantly gossips the local hashgraph $\mathcal{H}$ to random nodes and second loop deals with accepting gossips and achieving consensus.

We describe second loop in more detail. Firstly, node waits until it receives a gossip $\mathcal{H}'$ and includes correctly formed events to its local hashgraph $\mathcal{H}$. Then it creates a new event E which can include some new transaction data and adds it to $\mathcal{H}$. After that, node runs three subprotocols to achieve consensus on the order of events.

- $\mathsf{divideRounds}(\mathcal{H})$ – assigns each event in the hashgraph a round number and marks events that are witnesses, i.e., first events of a round for a given node. More precisely, there can be N witnesses in each round, one for each node. Events without ancestors get round number 1 and otherwise events get round number r that is the maximum of their parents' or $r+1$ if event can strongly see $2N/3$ of round r witnesses.
- $\mathsf{decideFame}(\mathcal{H})$ – runs a voting algorithm (locally) based on the knowledge of events that other nodes are aware of to decide which witnesses are famous. Essentially a witness of round r is elected to be famous if more than $2N/3$ of the witnesses of round $r+1$ can see it. Actual voting process is a bit more complicated as can be seen on Figure 2.
- $\mathsf{decideOrder}(\mathcal{H})$ – assigns a round received number (which is different from round number) to each event. Event E gets a round received number r if r is the smallest such round number that (i) witnesses up to round r have their fame decided and E is ancestor of all (unique) famous round r witnesses. Round received number is finally used to order the events. Ties are broken based on timestamps and still remaining ties are broken based on whitened signatures. Whitened signature is the signature of the event XORed with the signatures of all the unique famous witnesses in the received round.

3.3 Efficiency

Hashgraph white paper does not give precise complexity estimates. Clearly, the protocol in Figure 2 is not fully optimized. Most likely the subprocedures decideFame, divideRounds, and decideOrder can be be modified such that instead of constantly iterating over all the known events, algorithm updates the previous state by just considering the newly received events. Hence, computational complexity might not become a bottleneck.

Situation with communication complexity is less clear. Gossiping the full hashgraph each time (as in Figure 2) will become very inefficient as the number of events grows. Obvious optimization is that P_i gossips to P_j only the events it has not gossiped to P_j before. That would still mean that every node gossips each event to every other node, hence we would get $O(N^2)$ communication complexity.

Table 1: Comparison of tangle and hashgraph

	tangle	**hashgraph**
security	informal argument	mathematical proof
patent	no	yes
implementation	(mostly) open-source	closed-source (at the time of writing)
access	permissionless	permissioned

4 Conclusion

We reviewed two DAG-based distributed ledgers that claim to overcome many of the challenges faced by Bitcoin and other current blockchain protocols. Firstly we reviewed the IOTA DAG-based cryptocurrency. White paper of IOTA however lacks in cryptographic rigor and it remains unclear which assumptions are necessary and what precisely is the tangle (ledger of IOTA) achieving since there is no concrete security definition. Moreover, as of first half of 2018, IOTA is still relying on a centralized coordinator to confirm transaction. Secondly we looked at hashgraph consensus protocol of Swirlds distributed applications platform. They are much more precise with security assumptions and have a security proof. However, hashgraph consensus is foremost meant for the permissioned setting. Making it inconsistent with the ideology of cryptocurrencies that want to get rid of trusted parties. Hence, it is unlikely to be an acceptable substitute for Bitcoin, but it might be a reasonable solution for the corporate setting that they are currently aiming for. We briefly reiterate the main properties of both systems in Table 1.

main:

Each $\mathsf{P}_i \in \mathcal{P}$ does the following:

1. $\mathcal{H} \leftarrow \emptyset$; `//initialize hashgraph`
   `//(a) and (b) run in parallel`
2. (a) **while** True **do** gossip($\mathcal{H}$);
   (b) **while** True **do**
   i. receive gossip $\mathcal{H}'$;
   ii. $\mathcal{H} \leftarrow \mathsf{filter}(\mathcal{H} \cup \mathcal{H}')$;
   iii. construct new event E;
   iv. $\mathcal{H} \leftarrow \mathcal{H} \cup \{E\}$;
   v. divideRounds($\mathcal{H}$);
   vi. decideFame($\mathcal{H}$);
   vii. decideOrder($\mathcal{H}$);

decideOrder($\mathcal{H}$):

1. **For** $E \in \mathsf{topSort}(\mathcal{H})$ **do**
   (a) $r \leftarrow \min \left\{ \begin{array}{l} r' \in \mathbb{N} \mid \big(\forall E' \in \mathcal{H} : E'.\mathsf{rnd} > r' \\ \vee \neg E'.\mathsf{w} \vee E'.\mathsf{f} \neq \bot\big) \bigwedge \\ \big(\forall E' \in \mathsf{uFam}(\mathcal{H}, r') : E \in \delta_a^-(E')\big) \end{array} \right\}$;
   (b) **If** $r \neq \bot$ **do**
   i. $E.\mathsf{rndr} \leftarrow r$;
   ii. $S \leftarrow \left\{ \begin{array}{l} E' \in \mathcal{H} \mid \exists E_f \in \mathsf{uFam}(\mathcal{H}, r) : E' \in \delta_{sa}^-(E_f) \wedge \\ E \in \delta_a^-(E') \wedge E'.\mathsf{sp} \neq E \end{array} \right\}$;
   iii. $E.\mathsf{ts} \leftarrow \mathsf{median}(\{E'.\mathsf{ts} \mid E' \in S\})$;
2. **Return** $\mathcal{H}$ sorted first by rndr, remaining ties by ts, and still remaining ties by whitened signatures

divideRounds($\mathcal{H}$):
For $E \in \mathsf{topSort}(\mathcal{H})$ **do**

1. **If** $\delta^-(E) = \emptyset$ **do** $r \leftarrow 1$;
   Else $r \leftarrow \max_{E' \in \delta^-(E)} E'.\mathsf{rnd}$;
2. $S \leftarrow \left\{ \begin{array}{l} E' \in \mathcal{H} \mid E'.\mathsf{rnd} = r \wedge E'.\mathsf{w} \wedge \\ \mathsf{strongSee}(E, E') \end{array} \right\}$;
3. **If** $|S| \geq 2N/3$ **do** $E.\mathsf{rnd} \leftarrow r + 1$; **Else** $E.\mathsf{rnd} \leftarrow r$;
4. $E.\mathsf{w} \leftarrow (E.\mathsf{sp} = \bot) \vee (E.\mathsf{rnd} > E.\mathsf{sp}.\mathsf{rnd})$;

decideFame($\mathcal{H}$):
For $E \in \mathsf{topSort}(\mathcal{H})$ **do**

1. $E.\mathsf{f} \leftarrow \bot$;
2. **For** $E' \in \mathsf{topSort}(\mathcal{H})$ **do**
   If $E.\mathsf{w} \wedge E'.\mathsf{w} \wedge E'.\mathsf{rnd} > E.\mathsf{rnd}$ **do**
   (a) $d \leftarrow E'.\mathsf{rnd} - E.\mathsf{rnd}$;
   (b) $S \leftarrow \left\{ \begin{array}{l} E^* \in \mathcal{H} \mid E^*.\mathsf{rnd} = E'.\mathsf{rnd} - 1 \\ \wedge E^*.\mathsf{w} \wedge \mathsf{strongSee}(E', E^*) \end{array} \right\}$;
   (c) $t_0 \leftarrow |\{E^* \in S \mid E^*.\mathsf{v}_E = 0\}|$;
   (d) $t_1 \leftarrow |\{E^* \in S \mid E^*.\mathsf{v}_E = 1\}|$;
   (e) $v \leftarrow t_1 \geq t_0$; `//vote`
   (f) $t \leftarrow |\{E^* \in S \mid E^*.\mathsf{v}_E = v\}|$
   (g) **If** $d = 1$ **do** $E'.\mathsf{v}_E \leftarrow \mathsf{see}(E', E)$;
   Else
   i. **If** $d \mod c > 0$ **do**
   A. $E'.\mathsf{v}_E \leftarrow v$;
   B. **If** $t > 2N/3$ **do** $E.\mathsf{f} \leftarrow v$;
   Else
   A. **If** $t > 2N/3$ **do** $E'.\mathsf{v}_E \leftarrow v$;
   Else $E'.\mathsf{v}_E \leftarrow$ middle bit of $E'.\mathsf{sig}$;

Figure 2: Full description of the hashgraph consensus algorithm

References

[Bai16] Leemon Baird. The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance. `https://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf`, May 2016. Accessed: 31.03.2018. 1, 3, 3

[BO83] Michael Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing*, PODC '83, pages 27–30, New York, NY, USA, 1983. ACM. 3

[CDE+16] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains. In Jeremy Clark, Sarah Meiklejohn, Peter Y.A. Ryan, Dan

Wallach, Michael Brenner, and Kurt Rohloff, editors, *Financial Cryptography and Data Security*, pages 106–125, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. 1

[CL99] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, pages 173–186, Berkeley, CA, USA, 1999. USENIX Association. 3

[DLS88] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, April 1988. 3

[GKL17] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 291–323, 2017. 1

[HNDV17] Ethan Heilman, Neha Narula, Thaddeus Dryja, and Madars Virza. Iota vulnerability report: Cryptanalysis of the curl hash function enabling practical signature forgery attacks on the iota cryptocurrency. `https://github.com/mit-dci/tangled-curl/blob/master/vuln-iota.md`, 2017. Accessed: 31.03.2018. 2.2

[IOT] Iota documentation. `https://dev.iota.org`. Accessed: 25.03.2018. 1, 2

[KRDO17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 357–388, 2017. 1

[Lam98] Leslie Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998. 3

[LSZ15] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In Rainer Böhme and Tatsuaki Okamoto, editors, *Financial Cryptography and Data Security*, pages 528–547, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. 1

[Mer90] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 218–238, New York, NY, 1990. Springer New York. 2

[MGGR13] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 397–411, Washington, DC, USA, 2013. IEEE Computer Society. 1

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

[OM14] K. J. O'Dwyer and D. Malone. Bitcoin mining and its energy footprint. In *25th IET Irish Signals Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014)*, pages 280–285, June 2014. 4

[Pop17] Serguei Popov. The tangle. `https://iota.org/IOTA_Whitepaper.pdf`, 2017. Accessed: 31.03.2018. 1, 2, 2.1

[Reb17] Joseph Rebstock. Replay attacks in iota. `https://github.com/joseph14/iota-transaction-spammer-webapp/blob/master/replay%20attack.md`, 2017. Accessed: 31.03.2018. 2.2

[Sch90] Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, December 1990. 3

[SLZ16] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: A fast and scalable cryptocurrency protocol. *IACR Cryptology ePrint Archive*, 2016:1159, 2016. 1

[SZ15] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In Rainer Böhme and Tatsuaki Okamoto, editors, *Financial Cryptography and Data Security*, pages 507–527, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. 1

[SZ18] Yonatan Sompolinsky and Aviv Zohar. Phantom: A scalable blockdag protocol. Cryptology ePrint Archive, Report 2018/104, 2018. `https://eprint.iacr.org/2018/104`. 1

[Wal17] Eric Wall. Iota is centralized. `https://medium.com/@ercwl/iota-is-centralized-6289246e7b4d`, 2017. Accessed: 31.03.2018. 2.2