SPECTRE:
A Fast and Scalable Cryptocurrency Protocol

Paper #174

Abstract

Bitcoin utilizes Nakamoto Consensus to achieve agreement on a consistent set of transactions, in the permissionless setting, where anyone can participate in the protocol anonymously. Since its rise, many other permissionless consensus protocols have been proposed. We present SPECTRE, a new protocol for the consensus core of cryptocurrencies that remains secure even under high throughput and fast confirmation times. At any throughput, SPECTRE is resilient to attackers with up to 50% of the computational power (reaching the limit defined by network congestion and bandwidth constraints). SPECTRE can operate at arbitrarily high block creation rates, which implies that its transactions confirm in mere seconds (limited mostly by the round-trip-time in the network).

SPECTRE's underlying model falls into the category of partial synchronous networks: its security depends on the existence of some bound on the delivery time of messages between honest participants, but the protocol itself does not contain any parameter that depends on this bound. Hence, while other protocols that do encode such parameters must operate with extreme safety margins, SPECTRE converges according to the actual network delay.

Key to SPECTRE's achievements is the fact that it satisfies weaker properties than classic consensus requires. In the conventional paradigm, the order between any two transactions must be decided and agreed upon by all non-corrupt nodes. In contrast, SPECTRE only satisfies this with respect to transactions performed by honest users. We observe that in the context of money, two conflicting payments that are published concurrently could only have been created by a dishonest user, hence we can afford to delay the acceptance of such transactions without harming the usability of the system. Our framework formalizes this weaker set of requirements for a cryptocurrency's distributed ledger. We then provide a formal proof that SPECTRE satisfies these requirements.

1. INTRODUCTION

Bitcoin, which was invented and later deployed by Satoshi Nakamoto [17], represents the first successful *permissionless* cryptocurrency system. In contrast to classic distributed systems, the set of participants in the protocol is dynamic, anonymous, and *a priori* unknown. Nakamoto utilized Proof-of-Work (PoW) [8], [12] to limit the behavior of pseudonymous participants in the system and to counter possible Sybil attacks.

The Bitcoin protocol organizes transactions made in the currency in its public ledger, *the blockchain*. Each *block* in the chain is a batch of transactions that were published by users of the currency. The system is designed according to the replicated state machine approach [24]: nodes have local replicas of the ledger, and they reach agreement on the full order of blocks (and by extension on the full order of transactions).

Unfortunately, recent research has shown that the Nakamoto consensus has severe scalability limitations [6], [25], [11], [18]. Increasing the system's throughput (either via an increase in block size or block creation rate) comes at the expense of security: Under high throughput, Nakamoto's original guarantee no longer holds, and attackers with less than 50% of the computational power are able to disrupt the system. To avoid this, Bitcoin was set to operate at extremely low rates. The protocol enforces a slow block creation rate, and small block sizes, extending the blockchain only once every 10 minutes (in expectation) with a block containing up to 1 MB (roughly 2,000 transactions). Users must thus wait a long while to receive approval for their transfers.

In this paper we present SPECTRE (Serialization of Proof-of-work Events: Confirming Transactions via Recursive Elections), a PoW-based protocol that can process a high throughput of transactions and maintain fast confirmation times while remaining secure from attackers with up to 50% of the computational power. SPECTRE relies on a data structure that generalizes Nakamoto's blockchain into a direct acyclic graph (a block DAG). By maintaining a full DAG of blocks, SPECTRE can allow miners to create blocks concurrently and much more frequently. This design is intended to avoid the need for nodes to reconcile their different world views regarding the identity of a selected chain at the time of block creation. We provide extensive theoretical analysis of the protocol, its security and scalability including simulation results that demonstrate its advanced capabilities.

Network parameters. Our model of the system falls into the paradigm of partially synchronous networks [7]. In SPECTRE, the mining operation is oblivious to the propagation delay of the network, and no knowledge of it is required when running a full node. An assumption about some upper bound on the propagation delay is only used in the analysis of the protocol. Users who wish to estimate this may apply different bounds, according to their beliefs about recent network conditions. In contrast, Nakamoto's consensus requires that the block creation rate would far exceed the delay in the system in order for it to be secure. Thus, in Bitcoin and its many variants, an assumption on the propagation delay is fixed within the protocol (via the hardcoded parameter regulating block creation). The decision regarding the value of this parameter must account for possible future fluctuations in network delay (which may also be due to attacks on the network) and must be extremely conservative. Bitcoin thus operates slowly even when network conditions could allow it to operate faster. (This is related to the *responsiveness* metric introduced recently in [20]).

Throughput. In SPECTRE, large and fast blocks can be created, even when others have not fully propagated to all nodes. This process of extending the block DAG can be accelerated up to the limit imposed by the bandwidth of nodes. An assumption on the available bandwidth is fixed within the protocol, via the parameter that regulates the difficulty of block creation (w.r.t. this parameter we adapt Bitcoin's retargeting scheme, see Appendix C). For instance, if full nodes are assumed to support a minimal bandwidth of 10 MBps, then the block creation rate can be adjusted to 10 blocks per second, with block size limit of 1 MB, resulting in a throughput of $\sim$20,000 transactions per second, rivaling the global transaction rate of Visa at peak times.

Confirmation times. As mentioned above, the waiting time until a transaction recipient can safely accept it depends on his belief regarding recent network conditions. Additionally, as in Bitcoin, the user must assume an upper bound on the relative computational power of potential attackers. Importantly, in SPECTRE, an attacker must actively engage with his victim and transfer funds to him before executing the attack and reversing the payment. Consequently, users who are not concerned by large miners defrauding them (e.g., merchants conducting transactions at a physical point of sale), can accept transactions in SPECTRE extremely fast. In contrast, in Bitcoin and its variants, a large attacker can disrupt the network and reverse multiple payments without engaging with his victims, by overriding the main chain. Practically, these considerations imply that transactions in SPECTRE can be accepted in mere seconds, under normal network conditions, compared to Bitcoin's tens of minutes.

Consensus properties. Nakamoto's protocol satisfies both *Liveness* and *Safety*: the order between any two transactions is decided, and this decision is not reversed, w.h.p. [11], [19]. When a total order over transactions exists, it is simple to select a consistent set of transactions: reject all but the first transaction (according to the order) of any conflicting set.

In contrast, SPECTRE satisfies a weaker version of Liveness, and guarantees only that the decision w.r.t. *some* pairs of transactions is made. If two conflicting transactions were published at about the same time, the identity of the prevailing transaction might remain undetermined for arbitrarily long periods of time (this may occur only if an attack is actively taking place). Our key insight is that, in the context of cryptocurrencies, transactions only conflict if they transfer the exact same funds to two different locations, which implies that they were both generated by the original owner of the funds (cryptographic signatures ensure that only the owner may move his funds). Hence, an honest participant will *never* create such conflicts, and we can afford to delay the decision regarding transactions that have been *visibly* double-spent. Thus, SPECTRE in its current form is not well suited for general computation where the total order of inputs is needed, e.g., in systems like Ethereum [1] (it is still applicable to other computations where the order of inputs does not matter – one such example is voting, where the order of entered votes does not affect the outcome). The main challenges that we address are first to provide a PoW system with the weaker properties, but also to extract a consistent set of (irreversible) transactions even when a total order does not exist.

We provide a formal framework and define precisely the properties that a consensus protocol for cryptocurrencies ought satisfy, namely, *Weak Liveness*, *Progress*, and *Safety*. Our relaxation of the traditional paradigm may be of independent interest.

Incentives and rewards. Every block grants its creator a financial reward, as a compensation for his computational resources, provided that it was created according to the updated difficulty of block creation (refer to Appendix C for more details). An additional benefit of SPECTRE's high block creation rate is that mining rewards have significantly lower variance compared to Bitcoin, which reduces the need for large mining pools and contributes to the system's decentralization. Importantly, SPECTRE is resilient to block withholding attacks (aka selfish mining) that aim to increase one's own share of the minted money by manipulating the main chain [10], [23]. Indeed, in SPECTRE every block is rewarded, and there is no notion of a selected chain.

Organization of the paper. The remainder of this paper is organized as follows: In the subsequent subsection we discuss related work. In Sec. 2 we formalize our network model, the desired properties of the protocol, and the main result. In Sec. 3 we describe the basic operation of SPECTRE. In Sec. 4 we provide the gist of the correctness proof, and we conclude the main body of the paper in Sec. 5. We complement the description of SPECTRE's procedures in Appendix A. We present simulation results and demonstrate SPECTRE's exponentially fast confirmation times in Appendix B, and discuss implementation details in Appendix C. In Appendix D we provide examples of attacks, and intuition for why they fail to disrupt the protocol. Appendix E contains the full proof of our main theorem, stating that SPECTRE satifies the desired properties.

A. Related work

Previous research has produced several suggestions for protocols that attempt to address the security-scalability challenge, but all protocols still provide a total order over blocks:

The GHOST protocol is an alternative chain selection rule that gradually chooses a tree of blocks until converging on a single chain [25]. It can be shown that the Liveness property of GHOST can be attacked in several ways, as was demonstrated by [13].

The use of block DAGs was proposed in the Inclusive protocol [15], in which throughput was increased by combining discarded blocks off-chain blocks into the ledger. Due to the reliance on a chain, Inclusive mitigates but does not avoid the security-scalability trade-off. The Inclusive paper further includes a game theoretic analysis of the incentives of nodes to embed different transactions in their blocks (without the ability to coordinate).[1]

Bitcoin-NG [9] provides a clever chain structure that is composed of two types of blocks: key blocks that require PoW but contain no transactions, and mini-blocks that do not require PoW but do contain transactions. Bitcoin-NG manages to obtain a significant scalability increase, but its key blocks are still generated slowly, hence, confirmation times remain high.

Another line of work bootstraps PoW to instantiate a committee that is later used to run classical Byzantine fault tollerant protocols. Examples from this line of research include Byzcoin [14], a work by Decker *et. al.* [5], Hybrid Consensus [21], and recently Solidus [2].

[1]We build on this argument, and indeed assume that nodes will maximize their profits by avoiding transaction "collisions" and will try to embed unique content in their blocks.

Protocols built in this manner are highly scalable, building upon work in consensus protocols, but lack some of the properties achieved by Bitcoin. They typically require large committees and require committee members to remain online for long periods of time, making them susceptible to network isolation and denial of service attacks. [14], [5], [2] additionally fail without recovering if the committee is ever composed of a high fraction of malicious entities (Bitcoin, on the other hand, is self-stabilizing). Moreover, they require forward secrecy. If the cryptographic keys of a sufficient fraction of the committee at any point in the past is compromised, the attacker can create an alternative equally acceptable version of events. In contrast, miners in SPECTRE are not directly involved in any explicit consensus protocol and moreover can operate with little regard of other nodes' synchronization status.

Honey Badger [16] is an atomic broadcast protocol that is oblivious to network parameters and does not require tuning under different network conditions (similarly to SPECTRE). It is set in the classical permissioned setting where identities of the participants are known.

2. Formal Statement of the Problem

A. Network model and assumptions

Transactions. While SPECTRE can be used in other use cases as well, we focus on transactions that are created for the purpose of sending money and are constructed similarly to Bitcoin transactions. A transaction is typically denoted tx. As several copies of the very same transaction may appear in the system, we denote the equivalence class of tx (which contains all of its copies) by $[tx]$. $inputs\,(tx)$ is the set of (equivalence classes of) transactions that must be accepted before tx can be accepted; these are the transactions that have provided the money that is being spent in tx. Two transactions tx_1 and tx_2 (that are not equivalent) conflict if they share a common input, i.e., they double spend the same money; we then write $tx_2 \in conflict\,(tx_1)$ (this is a symmetric relation).

Network and communication. The system is operated by nodes, connected by a P2P network $\mathcal{N}$. We denote by $honest \subset \mathcal{N}$ the set of nodes that always adhere to the protocol's instructions. We assume that the honest nodes make up a connected component in the network graph, and that messages that they send are forwarded to their peers via gossip algorithms. Nodes are instructed to propagate all messages they receive or create to their peers.

We assume the existence of a constant $D > 0$ such that any message (of size $\leq B$ KB) transmitted by an honest node arrives after at most D seconds at all honest nodes, regardless of any manipulation of dishonest nodes. In addition, any node $v \in \mathcal{N}$ has an upper bound on D, denoted by d^v, representing its own bound on the communication delay diameter. We do not require that nodes agree on this bound.

We denote by G_t^v the data structure repersenting the history of messages observed by node $v \in \mathcal{N}$ up to time t. We denote $G_t^{pub} := \cup_{v \in honest} G_t^v$.

B. Desired properties

In our framework, a protocol for transaction acceptence is specified by two procedures: $GetAcceptedTxs$(G), which given the current (local) state of all messages, G, outputs a consistent set of transactions; $CheckRobustAccept(G, tx, \epsilon, d, \alpha)$, which outputs ACCEPT if $tx \in GetAcceptedTxs(G)$ and the probability that it remains so forever is at least $1 - \epsilon$, and WAIT otherwise. Here, ϵ is the risk that the user is willing to tolerate, d is an upper bound on D, and α is an upper bound on the relative computational power held by dishonest nodes (both according to the user's belief). We say that node v ϵ-accepted a transaction at time t if $CheckRobustAccept(G_t^v, tx, \epsilon, d^v, \alpha) = ACCEPT$.

The following properties are desirable:

Property 1 (Consistency)**.** The accepted set is consistent: For any history G,

1) if $tx \in GetAcceptedTxs(G)$ and $tx_2 \in inputs\,(tx)$ then $tx_2 \in GetAcceptedTxs(G)$.
2) if $tx \in GetAcceptedTxs(G)$ and $tx_2 \in conflict\,(tx)$ then $tx_2 \notin GetAcceptedTxs(G)$.

Property 2 (Safety)**.** If some node $\epsilon-$accepts, then w.h.p. all other honest nodes will do so: For any $v \in honest$ and time t, if $CheckRobustAccept\,(tx, \epsilon, G_t^v, d^v, \alpha) = ACCEPT$ then, with probability of at least $1 - \epsilon$, there exists a $\tau \geq t$ such that $\forall u \in honest, \forall s \geq \tau$: $CheckRobustAccept\,(tx, \epsilon, G_s^u, d^u, \alpha) = ACCEPT$, and the expectation of $\tau - t$ is finite.

Property 3 (Progress)**.** Transactions become more robust as time passes: For any $v \in honest$ and time t, if $CheckRobustAccept\,(tx, \epsilon, G_t^v, d^v, \alpha) = ACCEPT$ then, with probability of $1-\epsilon$ at least, for any ϵ' there exists a ϕ such that $\forall s \geq \phi : CheckRobustAccept\,(tx, \epsilon, G_s^v, d^v, \alpha) = ACCEPT$, and the expectation of $\phi - t$ is finite.

Property 4 (Weak Liveness)**.** Transactions are $\epsilon-$accepted in finite time, provided that their inputs are $\epsilon-$accepted, and there are no conflicts visible: Let t be the current time, and assume that $tx \in G_t^{pub}$. Let $\psi \geq t$ be the earliest time after t at which an honest node ϵ-accepts tx. Then, conditioned on the event where $conflict\,(tx) \cap G_\psi^{pub} = \emptyset$ and on the event where for all $tx_2 \in inputs\,(tx)$, tx_2 remains ϵ-accepted forever (by some honest node), the expectation of $\psi - t$ is finite.

C. Main result

The highlight of this work is the SPECTRE protocol, which satisfies the above properties:

Theorem 1. *SPECTRE's $GetAcceptedTxs$ procedure (Algorithm 2) satisfies Property 1. In addition, for any $\alpha < 0.5$ and $d^v \geq D$, SPECTRE's $CheckRobustAccept$ procedure (Algorithm 6) satisfies Properties 2-4. Moreover, the expected waiting times $\mathbb{E}[\tau - t]$ and $\mathbb{E}[\psi - t]$ increase logarithmically in $1/\epsilon$, and similarly $\mathbb{E}[\psi - t]$ in $1/\epsilon'$.*

The full proof can be found in Appendix E. The expected values of $\tau - t$ and $\phi - t$ are in $\mathcal{O}\left(\frac{\ln(1/\epsilon)}{\lambda(1-2\alpha)} + \frac{d^v}{1-2\alpha}\right)$ (and similarly for $\psi - t$ w.r.t. ϵ'). λ will be defined in the next section.

3. The SPECTRE Protocol

A. The generation of the block DAG

As in Bitcoin, participating nodes (called *miners*) create blocks of transactions by solving PoW puzzles. A block specifies its direct predecessors by referencing their ID in its header (a block's ID is the cryptographic hash of its header); we will describe in the next subsection how these predecessors are chosen. This results in a structure of a direct acyclic graph (DAG) of blocks (as blocks can only reference blocks created before them), denoted typically $G = (C, E)$. Here, C represents blocks and E represents the hash references. We will frequently write $z \in G$ instead of $z \in C$.

$past\,(z, G) \subset C$ denotes the subset of blocks reachable from z, and similarly $future\,(z, G) \subset C$ denotes the subset of blocks from which z is reachable; these are blocks that were provably created before and after z, correspondingly. *Note that an edge in the DAG points back in time, from the new block to previously created blocks which it extends.* We denote by $cone\,(z, G)$ the set of blocks that the DAG directly orders with respect to z: $cone\,(z, G) := past\,(z, G) \cup \{z\} \cup future\,(z, G)$, and by $anticone\,(z)$ the complementary of $cone\,(z, G)$. The set $past\,(b, G)$ is fixed once and for all at the creation of b (in sharp contrast to $future\,(z, G)$ and $anticone\,(z, G)$ that may grow as blocks are added later to the DAG), hence we can simply write $past\,(b)$ without noting the context.

The unique block $genesis$ is the block created at the inception of the system, and every valid block must have it in its past set. In addition, we relate to a hypothetical block, $virtual\,(G)$. This block satisfies $past\,(virtual\,(G)) = G$. While its role is merely methodological, $virtual\,(G)$ can also be thought of as representing the next block that a node whose current observed DAG is G attempts to create.

G_t^v denotes the block DAG observed by node $v \in \mathcal{N}$ at time t. This DAG represents the history of all (valid) block-messages received by the node, instantiating the abstract data structure assumed in Sec. 2.

A block b is considered by a node as valid only if it received $past\,(b)$, the block is of size at most B KB (which in turn affects the parameter D discussed in Sec. 2), and the cryptographic hash of its header (including the required nonce, serving as PoW) is smaller than some predefined constant $TARGET$. The creation of blocks is well-approximated by a Poisson process. We denote by λ the rate of block creation in the system, when all nodes in the network participate. $TARGET$'s value is occasionally readjusted, so as to keep λ constant, as explained in Appendix C.

B. The miners' protocol

SPECTRE's instructions to miners are extermely simple:

1) *When creating or receiving a block, transmit the block to all peers.*
2) *When creating a block, embed in its header a list containing the hash of all leaf-blocks (blocks with in-degree 0) in the locally-observed DAG.*

Note that these instructions allow miners to operate concurrently irrespective of potential conflicts in the contents of their blocks.

C. The users' protocol

As the block DAG may contain conflicting transactions, we must provide a method for network nodes to interpret the DAG and extract from it the set of accepted transactions. Doing so in a way that will be agreed upon by all nodes (eventually) is the main challenge of SPECTRE. We now describe how this is done.

The topology of a block DAG G induces a natural precedence-relation over blocks: if x is reacheable from y (i.e., $x \in past\,(y)$) then x precedes y, as it was provably created before it. SPECTRE extends this relation into a complete relation over G's blocks, denoted $\prec$. This order is immediately translatable into an order over transactions in G: tx_1 precedes tx_2 if the block containing the former precedes that containing the latter. This relation, in turn, induces a natural subset of accepted transactions: tx is accepted if it precedes all of its conflicting transactions in G. The relation $\prec$ is generated by a pairwise vote procedure that occurs independently for every pair of blocks. The operation of this layer will be explained in the next subsections.

Although we may at times refer to $\prec$ as though it orders blocks, we stress that $\prec$ *is not necessarily a transitive relation*. It is possible to have a series of blocks that precede each other cyclically.[2] The lack of a total linear ordering over blocks is in fact the way SPECTRE utilizes the weaker consensus requirements of our framework, as a linear order is equivalent to solving the consensus problem [4] (which would lower the resilience thresold to 33% [21]).

The second layer of SPECTRE is responsible for correctly estimating how robust these relations are. Its main procedure, $CheckRobustAccept$, upper bounds the probability that a certain transaction, now considered accepted, will later be rejected. Such an event can occur only if the order between certain pairs of blocks is reversed, and so this challenge reduces to estimating the robustness of the pairwise ordering of these blocks. This layer of SPECTRE is highly involved, and we defer its description to Appendix A.

D. Pairwise ordering of blocks

The basic layer of SPECTRE involves deciding on a pairwise order over the block DAG. Fix two blocks $x, y \in G$. In order to decide if $x \prec y$ or $y \prec x$, we interpret the structure of the DAG as representing an abstract vote. Every block $z \in G$ is considered a voter with respect to the pair (x, y), and *its vote is inferred from the structure of the DAG*. We represent a vote by a number in $\{-1, 0, +1\}$, and we denote z's voting-profile on all pairs by $vote\,(z, G)$. $vote_{x,y}\,(z, G) = -1$ represents x preceding y ($x \prec y$), $vote_{x,y}\,(z, G) = +1$ represents y preceding x, and $vote_{x,y}\,(z, G) = 0$ represents a tie. Importantly, $vote\,(z, G)$ is an asymmetric relation: $vote_{y,x}\,(z, G) = -vote_{x,y}\,(z, G)$.

To simplify presentation, we associate a vote with $virtual\,(G)$ as well. Recall that the virtual block of G is a hypothetical block which satisfies $past\,(virtual\,(G)) = G$. The vote of $virtual\,(G)$ represents essentially the aggregated vote of the entire block DAG. The basic rules of z's vote, for any $z \in G \cup \{virtual\,(G)\}$, are as follows:

- if $z \in G$ is in $future\,(x)$ but not in $future\,(y)$ then it will vote in favour of x (i.e., for $x \prec y$).

[2]This is related to the Condorcet paradox in social choice [3].

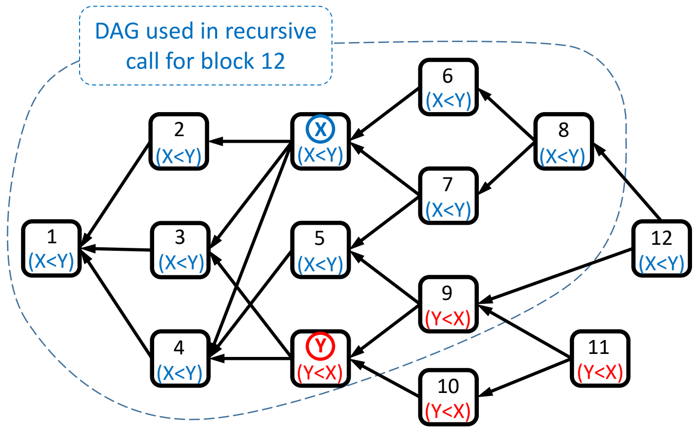

Fig. 1: An example of the voting procedure on a simple DAG. Block x and blocks 6-8 vote $x \prec y$ as they only see x in their past, and not y. Similarly, block y and blocks 9-11 vote $y \prec x$. Block 12 votes according to a recursive call on the DAG that does not contain blocks 10,11,12. Any block from 1-5 votes $x \prec y$, because it sees more $x \prec y$ voters in its future than $y \prec x$ voters.

- if $z \in G$ is in $future\left(x\right) \cap future\left(y\right)$ then z's vote will be determined recursively according to the DAG that is reduced to its past, i.e., it has the same vote as $virtual\left(past\left(z\right)\right)$. If the result of this vote is a tie, z breaks it arbitrarily.[3]
- if $z \in G$ is not in the future of either blocks then it will vote the same way as the vote of the majority of blocks in its own future.
- if z is the virtual block of G then it will vote the same way as the vote of the majority of blocks in G.
- finally, (for the case where z equals x or y), z votes for itself to succeed any block in $past\left(z\right)$ and to precede any block outside $past\left(z\right)$.

It is easy to see that all votes respect the DAG's topology: If x is reachable from y then all blocks vote unanimously $x \prec y$. Figure 1 illustrates the voting procedure with regards to a single pair of blocks (x,y). Additional examples along with intuition regarding this key algorithm are provided in Appendix D. The above procedure is implemented in Algorithm 1 below.

In the algorithm, $\widetilde{sgn}\left(n\right) = -1$ for $n < 0$, $\widetilde{sgn}\left(n\right) = +1$ for $n > 0$, and $\widetilde{sgn}\left(0\right) = 0$. To see that the recursion calls from line 4 halt, observe that they take as inputs DAGs strictly smaller than G (because $past\left(z\right) \subsetneq G$), and hence eventually all arrive at the base case $G = \emptyset$ and return. The algorithm is written in its naïve form, for the sake of readability, with a run time of $\mathcal{O}(|G|^3)$. A more sophisticated approach implements it in expected time of $\mathcal{O}(d \cdot \lambda)$.

The pairwise ordering of SPECTRE has the following highly valuable property: *Once a block is published, the set of blocks that precede it in the pairwise ordering closes fast—w.h.p. it only consists of blocks that were published before or right after its publication.*

The implications of this guarantee to the security of transactions is immediate, at least at the intuitive level: A user whose transaction is embedded in some published block x can guarantee its safety by waiting some time after x's publication before accepting it; he is then guaranteed

[3] We can use information encoded in z's header, e.g., explicit instructions for tie-breaking, or use the lexicographical ordering of (hashes of) tied blocks, etc.

Algorithm 1 $CalcVotes$

```
Input: $G$ – a block DAG
Output: $vote\left(virtual\left(G\right)\right)$ – a pairwise ordering of blocks in $G$
 1: if $G = \emptyset$ then
 2:     return an empty ordering
 3: for all $z \in G$ do
 4:     $vote\left(z, past\left(z\right)\right) \leftarrow CalcVotes\left(past\left(z\right)\right)$ and break ties arbitrarily
 5: for all $z \in G$ in some topological order (from leaves to root) do
 6:     for all $x, y \in G$ $(x \neq y)$ do
 7:         if $\left(x \in \overline{past}\left(z\right) \wedge y \notin past\left(z\right)\right) \vee \left(x \in past\left(z\right), y = z\right)$ then
 8:             $vote_{x,y}\left(z, G\right) \leftarrow -1$
 9:         else if $\left(y \in \overline{past}\left(z\right) \wedge x \notin past\left(z\right)\right) \vee \left(y \in past\left(z\right), x = z\right)$ then
10:             $vote_{x,y}\left(z, G\right) \leftarrow +1$
11:         else if $x, y \in past\left(z\right)$ then
12:             $vote_{x,y}\left(z, G\right) \leftarrow vote_{x,y}\left(z, past\left(z\right)\right)$
13:         else if $x, y \notin past\left(z\right)$ then
14:             $vote_{x,y}\left(z, G\right) \leftarrow \widetilde{sgn}\left(\sum_{z' \in future(z,G)} vote_{x,y}\left(z', G\right)\right)$
15: $vote\left(virtual\left(G\right), G\right) \leftarrow \widetilde{sgn}\left(\sum_{z \in G} vote\left(z, G\right)\right)$
16: return $vote\left(virtual\left(G\right), G\right)$
```

that any block published later on – and that might contain a conflicting transaction – will be preceded by x hence will not threaten the acceptance of his transaction. In Sec. 4 we will explain how this guarantee is achieved.

E. Accepting transactions

Equipped with the pairwise relation over blocks, we now turn to construct the set of accepted transactions. Basically, we mark a transaction as accepted iff all three conditions below hold true:

- all of its inputs have been accepted.
- all conflicting transactions that are in its anticone set (i.e., that are not related to it topologically) are contained in blocks that are preceded by the block containing the transaction.
- all conflicting transactions that are in its past set (i.e., that precede it in the DAG, topologically) have been rejected.

Algorithm 2 implements these rules, and outputs a set of accepted transactions. It operates recursively, and should be initially called with $GetAcceptedTxs(G, G)$ (we later denote this simply by $GetAcceptedTxs(G)$). In the algorithm, the notation $Z_G(tx)$ stands for all blocks in G that contain tx. Some complexity arises due to possible multiple copies of the same transactions, which requires dealing with equivalence classes of transactions $[tx]$.

Algorithm 2 $GetAcceptedTxs$

Input: G – a block DAG, $subG$ – a subDAG of G which is the past of a (possibly virtual) block

Output: TX – a hyper-set of valid transactions in G

```
1: $vote\left(virtual\left(G\right)\right) \leftarrow CalcVotes(G)$
2: $TX \leftarrow \emptyset$
3: for all $z_1 \in subG$ do
4:     for all $tx \in z_1$ do
5:         for all $tx_2 \in G \cap conflict\left(tx\right)$ do
6:             for all $z_2 \in Z_G(tx_2) \cap anticone\left(z_1, G\right)$ do
7:                 if $vote_{z_1,z_2}\left(virtual\left(G\right)\right) \geq 0$ then
8:                     break (to line 4 and pick next $tx$)
9:             if $[tx_2] \cap GetAcceptedTxs(G, past\left(z_1\right)) \neq \emptyset$ then
10:                break (to line 4 and pick next $tx$)
11:        for all $[tx_3] \in inputs\left(tx\right)$ do
12:            if $[tx_3] \cap GetAcceptedTxs\left(G, past\left(z_1\right)\right) = \emptyset$ then
13:                break (to line 4 and pick next $tx$)
14:        add $tx$ to $TX$
15: return $TX$
```

4. High-level Overview of the Proof

We now provide some intuition as to why SPECTRE's procedures indeed guarantee that transactions can be accepted safely, and that all transactions of honest users are quickly accepted. We aim at proving the main property of SPECTRE: *For a published block, the set of blocks that precede it in the order of the DAG consists only of blocks that were published before or right after its publication, w.h.p.* As mentioned at the end of Subsection 3.4, this property is easy to translate to the desired security properties of transactions (as we do formally in Appendix E). We thus wish to prove the following statement:

Proposition. *Let x, y be two blocks. Assume that x was published at time t_{pub} ($x \in G^{pub}_{t_{pub}}$), and y was not published before time t_{acc} ($y \notin G^{pub}_{t_{acc}}$).*[4] *Then, the probability that, at some point in the future, x will not precede y in the DAG observable by some honest node ($\Pr\left(\exists u \in honest, \exists s \geq t_{acc} : vote_{x,y}\left(virtual\left(G^u_s\right)\right) \geq 0\right)$) decreases exponentially in $(t_{acc} - t_{pub})$.*

Proof overview. Assume that the event in which y comes to precede x in some future DAG occurs. Let s be the earliest moment in time that such an event occurred at some node. Notice that y cannot be in the past of x or in its future (otherwise their order is determined by the topology and cannot be reversed). We thus assume henceforth $y \in anticone\left(x\right)$.

[4]Intuitively, t_{acc} represents the time at which some node accepted a transaction which appears in block x.

The block race after x is published. We first consider the votes of blocks created after the publication of block x:

- (Almost) all honest blocks created between t_{pub} and t_{acc} vote forever in favour of $x \prec y$, as they have x in their past but not y. Denote by n_1 the number of such blocks.
- All honest blocks created between t_{acc} and s vote in favour of $x \prec y$, as well, by the choice of s. Denote by n_2 the number of such blocks.
- Denote by m_1 and m_2 the number of blocks created by the attacker in the time intervals corresponding to n_1 and n_2. Honest nodes posses a fraction $1 - \alpha > \alpha$ of the computational power. Consequently, for any positive constant C, the probability that the relation $m_1 + m_2 + C - (n_1 + n_2) \geq 0$ will ever be satisfied decreases exponentially with n_1. This is typically analyzed as the probability that a biased random walk on the integers, beginning at C, returns to the origin (see [17], [22], [25]).

The term $m_1 + m_2 - (n_1 + n_2)$ represents the aggregate vote between x and y, considering only blocks created after x's publication. We now show that blocks that the attacker prepared in advance before x's publication, in a preparatory "pre-mining" stage, do not give him more than some constant advantage (which will be counted into C above).

The pre-mining stage. Honest blocks that were created before x was published are typically in its past (apart from a small set of blocks) and hence have their vote decided by the majority of votes in their future (as per Alg. 1). Their vote is thus possibly subject to change as the DAG grows, and as the attacker publishes blocks.

For every block z in the past of x we must therefore consider the number of blocks above it that vote in favour of x and those that vote against it. Denote by X_z the gap between the number of attacker blocks and honest blocks in the future of z, up to time t_{pub}. In Lemma 22 we show that the worst case gap X_z (over all blocks $z \in past(x)$) can be modeled as a reflecting random walk over the nonnegative integers, with bias towards the origin. Consequently, the best gap that the attacker can secretly gain over a block in $past(x)$ has an exponentially decaying tail, and, in particular, is bounded by a constant w.h.p.

All in all, as $t_{acc} - t_{pub}$ grows, the number n_1 of votes, or "confirmations", that x receives increases linearly, and the probability that the attacker will be able to reveal enough blocks so that some $z \in past(x)$ will have more $y \prec x$ votes in its future than $x \prec y$ votes, decreases exponentially in n_1. As this holds for all $z \in past(x)$ uniformly, it implies in particular that the $genesis$ block has more $x \prec y$ votes in its future than $y \prec x$ votes (unless an exponentially unlikely event occurred). The vote of the virtual block is determined by that of the $genesis$ block (this is easy to see, and is proven in Lemma 11), completing the argument. $\square$

The proposition above is the gist of Lemmas 12, and 13. In the above sketch, we abstracted out many additional subtleties and details. For instance, honest blocks that were created D seconds around t_{pub}, t_{acc}, or s may not have contributed votes in favour of x. In our formal analysis we count these as attacker blocks, accounting for the worst case, and add them to the aforementioned constant C. We additionally show how the user can measure n_1 correctly, even if the attacker publishes his blocks in an attempt to delay acceptance.

5. Conclusion

In this work we presented SPECTRE, a new protocol for fast and scalable distributed ledgers. We have shown SPECTRE's ability to process a high rate of block creation, as well as to handle large blocks securely. Our results demonstrate that SPECTRE can achieve incredibly low confirmation times, especially compared to the Nakamoto consensus. Further work to improve and tighten the acceptance policy we derived can lower confirmation times further.

SPECTRE thus shows the feasibility of creating open, permissionless distributed ledgers that will work successfully at large scale. It achieves these properties in the partial synchronous setup, due to its willingness to delay the decision regarding visibly double-spent transactions.

References

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

[2] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Alexander Spiegelman. Solidus: An incentive-compatible cryptocurrency based on permissionless byzantine consensus. *arXiv preprint arXiv:1612.02916*, 2016.

[3] Kenneth J Arrow, Amartya Sen, and Kotaro Suzumura. *Handbook of Social Choice & Welfare*, volume 2. Elsevier, 2010.

[4] Miguel Correia, Nuno Ferreira Neves, and Paulo Veríssimo. From consensus to atomic broadcast: Time-free byzantine-resistant protocols without signatures. *The Computer Journal*, 49(1):82–96, 2006.

[5] Christian Decker, Jochen Seidel, and Roger Wattenhofer. Bitcoin meets strong consistency. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, page 13. ACM, 2016.

[6] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *13th IEEE International Conference on Peer-to-Peer Computing (P2P), Trento, Italy*, September 2013.

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

[8] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*, pages 139–147. Springer, 1992.

[9] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 45–59, 2016.

[10] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[11] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 281–310. Springer, 2015.

[12] Ari Juels and John G Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *NDSS*, volume 99, pages 151–165, 1999.

[13] Aggelos Kiayias and Giorgos Panagiotakos. On trees, chains and fast transactions in the blockchain. Cryptology ePrint Archive, Report 2016/545, 2016.

[14] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 279–296, 2016.

[15] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *International Conference on Financial Cryptography and Data Security*, pages 528–547. Springer, 2015.

[16] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 31–42. ACM, 2016.

[17] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[18] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. *IACR Cryptology ePrint Archive*, 2016:454, 2016.
[19] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. *IACR Cryptology ePrint Archive*, 2016:454, 2016.
[20] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model, 2016.
[21] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. Cryptology ePrint Archive, Report 2016/917, 2016.
[22] Meni Rosenfeld. Analysis of hashrate-based double spending. *arXiv preprint arXiv:1402.2009*, 2014.
[23] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. *arXiv preprint arXiv:1507.06183*, 2015.
[24] Fred B Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)*, 22(4):299–319, 1990.
[25] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 507–527. Springer, 2015.

APPENDIX A
ADDITIONAL ALGORITHMS

In Sec. 3 we described the way SPECTRE pairwise orders blocks, and how this order is used to construct the subset of accepted transactions. In this section we describe the procedures of the second layer of SPECTRE, which allows users to measure the robustness of this order, and to translate this into the robustness of accepted transactions.

A. Robustness of the block pairwise ordering

Assume that in the order of the current observable DAG the block x precedes y. We need a method to measure how likely it is that this relation will persist forever. Algorithm 3 outputs an upper bound on the probability that an attacker will be able to reverse the relation $x \prec y$. When the argument y is unspecified, the interpretation of the algorithm's output is x's robustness against an unseen block (withheld by an attacker or yet to be created). In the algorithm, $gap\left(b, G\right)$ denotes the size of the set $\{z \in anticone\left(b, G\right) : vote_{z,b}\left(virtual\left(G\right)\right) \geq 0\}$. The notation $\langle G, z, K \rangle$ will be explained in the paragraphs that follow.

In line 13 the algorithm uses several functions whose precise definitions we defer to later sections. An explicit formula for f_{pre_pub} is given in (54), for f_{post_pub} is given in (50) and (52), and for f_{post_mine} is given in (5). Preceding Lemma 22, we provide a method to calculate f_{pre_mine} numerically.

Intuitively, the function f_{pre_mine} upper bounds the probability that the attacker has gained an advantage larger than l during the pre-mining phase (i.e., up until the creation of x). The function f_{post_mine} upper bounds the probability that the attacker will ever be able to create enough blocks so as to reverse the relation $x \prec y$. In essence, f_{post_mine} is an adaptation of a formula from [22]. According to our version of the formula, if during the interval $[time(x), t_{acc}]$ (where t_{acc} represents the current time) honest nodes created n blocks, then $\binom{n-1+m}{m} \cdot \alpha^m \cdot (1-\alpha)^n$ is the probability that the attacker has created during this interval m blocks. If g aggregates all the votes of blocks in $future\left(x, G\right)$, then the probability that the attacker will be able to reverse

Algorithm 3 $Risk$ (offline)

```
Input: $G$ – a block DAG, $x$ – a block in $G$, $y$ (optional) – a block in $anticone\left(x,G\right)$
Output: $risk$ – an upper bound on the probability of block $x$ not defeating $y$ at any point in the future
 1: if $time\_now < publication(x) + 2 \cdot d$ then
 2:     return 1
 3: $K \leftarrow \lceil \sqrt{\left|future\left(x,G\right)\right|} \rceil$
 4: if $NULL = y$ then
 5:     $g \leftarrow \left|\overline{future}\left(x,G\right)\right|$
 6:     $M \leftarrow 0$
 7: else
 8:     $g \leftarrow \sum_{z' \in \overline{future}(x,G)} vote_{y,x}\left(z',G\right)$
 9:     $M \leftarrow \left|\left\{z \in future\left(x,G\right) : vote_{x,y}\left(z,G\right) = +1 \wedge gap\left(z,\langle G,z,K\rangle\right) = 0\right\}\right|$
10: $n_x \leftarrow \left|\overline{future}\left(x,G\right)\right| - M$
11: $j \leftarrow gap\left(x,G\right) + K$
12: $l \leftarrow K$
13: $risk \leftarrow f_{pre\_mine}(l) + f_{pre\_pub}(K) + f_{post\_pub}\left(M\right) + f_{post\_mine}\left(n_x,g,j,l,M\right)$
14: return $risk$
```

the majority's vote is roughly $\left(\frac{\alpha}{1-\alpha}\right)^{\max\{g-m,0\}}$. The combined expressions produce an upper bound on the success-probability of an attack.[5]

The main challenge here is to correctly measure n. This is a difficult task, as Algorithm 3 uses only structural information (with the exception of making sure that x has been published for at least $2 \cdot d$ seconds) and does not rely on measurements of blocks' timings. Naïvely one would use $n \approx \left|future\left(x,G\right)\right|$ to upper bound blocks created after $publication(x)$. However, there are two main difficulties:

- The block x might have been created by a dishonest node and withheld by it. In this case, there might have passed a long time between its creation and its publication, which implies that $\left|future\left(x,G\right)\right|$ alone may be well below n. To avoid underestimating n, we upper bound the number of honest blocks in $anticone\left(x,G\right)$, by the variable j, and add it to our count (the addition is done inside $f_{post\_mine}\left(n_x,g,j,l,M\right)$).
  The function f_{pre_pub} upper bounds the probability that we have underestimated j.
- By publishing his attack blocks, the attacker can increase the size of $future\left(x,G\right)$ and cause us to overestimate n. This would result in an upper bound on the success-probability

[5]The calculations we use are quite more involved, as will be detailed later on. The reason why we aggregate in g votes from $future\left(x,G\right)$ alone – rather than votes from the entire DAG – was discussed in Sec. 4. Essentially, observe that counting all votes – including votes of blocks in $past\left(x\right)$ – is not meaningful, as such voters might reverse their vote as future events unfold. Rather, it is useful to measure how robustly voters in $past\left(x\right)$ support $x \prec y$, which is captured by our calculations.

of an attack that is not tight enough, which would allow an attacker with a large value of α to delay acceptance indefinitely. $Risk$ overcomes this problem, by recognizing attacker blocks and excluding them from the count of n. This is done as follows.

Let G be a block DAG, b a block in G, and K a whole number. The DAG $\langle G, b, K\rangle$ is obtained by creating a new chain $z_1, ..., z_K$ of K hypothetical blocks, connecting an edge from z_1 to b and replacing every edge $(z, b) \in G$ with $(z, z_K) \in G$. Essentially, this adds to the DAG K artificial voters which vote strongly $x \prec y$, against any $y \notin past\,(x, G)$. In line 9, the algorithm checks whether $gap\,(z, \langle G, z, K\rangle) = 0$, i.e., whether there exists a block in $anticone\,(z)$ that precedes z in the modified DAG $\langle G, z, K\rangle$. In the case of a negative answer, z is counted into n (in line 10).

The following property explains why this procedure is useful: *If we add K voters in favour of an honest block, for some small K, then no other block will precede it in the pairwise ordering (apart from its past set).* This is restated formally and proven in Lemma 27.

The function f_{post_pub} upper bounds the probability that we have underestimated the number of honest blocks in $future\,(x, G)$.

B. Robustness of transaction acceptance

The next step is to translate robustness of blocks (calculated by $Risk$) to robustness of transactions. This transition is implemented similarly to the transition from the (non-robust) ordering of blocks (Alg. 1) to the (non-robust) accepted set of transactions (Alg. 2).

The $RiskTxAccept$ procedure (Alg. 4) takes as input G and tx (and an additional argument) and returns an upper bound on the probability that some honest node will not ϵ-accept tx. The main task of $RiskTxAccept$ is to properly account and aggregate the error bounds that $Risk$ induces. As can be easily recognized, $RiskTxAccept$ and $RiskTxReject$ (Alg. 5) are mirror images of each other. While $RiskTxAccept$ upper bounds the probability that a given transaction will ever be *removed* from the accepted transaction subset, $RiskTxReject$ upper bounds the probability that a given transaction will ever be *included* in this subset. This is particularly vital for the case where two conflicting transactions are related topologically, i.e., $tx_2 \in y$ and $tx_1 \in x \in future\,(y)$, but tx_2 is not in the accepted set (due to some previous conflict). In this case, although the block containing tx_2 precedes that containing tx_1, we accept tx_1. It can be further ϵ-accepted if the rejection-status of tx_2 is robust, as calculated by $RiskTxReject$.

Algorithm 4 $RiskTxAccept$

```
Input: $G$ – a block DAG, $subG$ – a subDAG of $G$ which is the past of a (possibly virtual)
    block, $tx$ – a copy of the transaction to defend
Output: $risk$ – an upper bound on the probability that some honest node in some future point
    in time will accept no transaction in $[tx] \cap subG$.
 1: $minrisk \leftarrow 1$
 2: for all $z_1 \in Z_G([tx]) \cap subG$ do
 3:     $risk \leftarrow Risk(G, z_1, \emptyset)$
 4:     for all $tx_2 \in G \cap conflict(tx)$ do
 5:         for all $z_2 \in Z_G(tx_2) \cap anticone(z_1, G)$ do
 6:             $risk \leftarrow risk + Risk(G, z_1, z_2)$
 7:         $risk \leftarrow risk + RiskTxReject(G, [tx_2], past(z_1))$
 8:     for all $[tx_3] \in inputs(tx) \cap past(z_1)$ do
 9:         $risk \leftarrow risk + RiskTxAccept(G, [tx_3], past(z_1))$
10:     $minrisk \leftarrow \min\{minrisk, risk\}$
11: $risk \leftarrow minrisk$
12: return $risk$
```

Algorithm 5 $RiskTxReject$

```
Input: $G$ – a block DAG, $subG$ – a subDAG of $G$ which is the past of a (possibly virtual)
    block, $tx$ – a copy of the transaction to defend
Output: $risk$ – an upper bound on the probability that some honest node in some future point
    in time will accept a transaction in $[tx] \cap subG$.
 1: $risk \leftarrow 0$
 2: for all $z_1 \in Z_G([tx]) \cap subG$ do
 3:     $minrisk \leftarrow 1$
 4:     for all $tx_2 \in G \cap conflict(tx)$ do
 5:         for all $z_2 \in Z_G(tx_2) \cap anticone(z, G)$ do
 6:             $minrisk \leftarrow \min\{minrisk, Risk(G, z_2, z_1)\}$
 7:         $minrisk \leftarrow \min\{minrisk, RiskTxAccept(G, [tx_2], past(z_1))\}$
 8:     for all $[tx_3] \in inputs(tx)$ do
 9:         $minrisk \leftarrow \min\{minrisk, RiskTxReject(G, [tx_3], past(z_1))\}$
10:     $risk \leftarrow risk + minrisk$
11: return $risk$
```

Building on these procedures, we now present the *CheckRobustAccept* procedure of SPECTRE. The user should provide as input the entire DAG that it currently observes.

Algorithm 6 $CheckRobustAccept$

```
Input: $G$ – a block DAG representing the current DAG observed by the node running the policy, $tx$ – the transaction to defend, $\epsilon$ – the maximum risk the user is willing to tolerate, $\alpha$ – maximal size of attacker, $d$ – maximal delay diameter of network, $\lambda$ – the block creation rate
Output: ACCEPT– if the probability that (after some initial waiting) there will be an honest node that will not $\epsilon$-accept $tx$ is smaller than $\epsilon$; WAIT– otherwise
1: if $RiskTxAccept\left(G, [tx] \cap G\right) < \epsilon$ then
2:     return ACCEPT
3: else
4:     return WAIT
```

C. Online policy

We now present an alternative implementation of $Risk$, which requires that the user be online at the time when his block gains confirmations. This assumption is highly reasonable for many practical scenarios, e.g., a cashier serving a continuous line of customers. The main benefit of the online version is that it relies on a tighter analysis, and therefore accepts transactions slightly faster. We now confine ourselves to the case where there is no visible double-spend (i.e., $y = NULL$).

The fact that the user is online can be utilized in two ways: First, any block that the user receives after $received^v(b) + 2 \cdot d$ and does not belong to $future\left(x\right)$ can be marked by him as an attacker block. Second, the user can estimate the number of hidden attacker blocks by measuring the time that passed since the creation of x.

Below we describe the online version of $Risk$. The algorithm takes as input node v's DAG and the block x to defend, and returns an upper bound on the probability that some block $y \in G_\infty^{pub} \setminus G_t^{pub}$ will ever precede it.

Algorithm 7 *Risk* (online)

```
Input: $G_t^v$ – the block DAG that $v$ obesrves at time $t$, $x$ – a block in $G_t^v$
Output: $risk$ – an upper bound on the probability of block $x$ not preceding $y$ at any point in
    the future, for some $y \in G_\infty^{pub} \setminus G_t^{pub}$
 1: if $time\_now < publication(x) + d$ then
 2:     return 1
 3: $T \leftarrow time\_now - received^v(x)$
 4: $G_x \leftarrow G^v_{received^v(x)+2\cdot d} \cup future\left(x, G_x\right)$
 5: $g \leftarrow \min_{x' \in \overline{anticone}(x, G_x)} \left|future\left(x', G_x\right)\right|$
 6: $risk \leftarrow risk\_hidden(T, g)$
 7: if $risk < \epsilon$ then
 8:     return $ACCEPT$
 9: else
10:     return $WAIT$
```

The definition of $risk\_hidden$ appears in (45)-(46). In practice, as node v may have a partial view of $G_\infty^{pub} \setminus G_t^{pub}$, in order to use Alg. 7 the user must wait additional d seconds and verify that $conflict\left(tx\right) \cap G^v_{t+d} = \emptyset$, i.e., that the attacker did not publish a double-spend in the interval $[t-d, t]$. The correctness of the online policy modification is proven in Corollary 25.

Appendix B
Simulation Results

We implemented the SPECTRE protocol in Python along with an event-driven simulator of network dynamics. For each experiment we generated an Erdős-Rényi random network topology with 20 nodes. Each node forms 5 outgoing links, in expectation. The delay on each link was uniformly distributed and later scaled linearly so that the diameter of the graph is d (for the given d). Every point represents the average outcome over at least 500 experiments.

The main benefit of SPECTRE is fast transaction confirmation. The asymptotic waiting times derived from our formal analysis are in $\mathcal{O}\left(\frac{\ln(1/\epsilon)}{\lambda(1-2\alpha)} + \frac{d^v}{1-2\alpha}\right)$. In order to measure the actual waiting times, we utilized the online acceptance policy derived by Alg. 7. Accordingly, we stress that the merchant needs to wait additional d seconds in order to verify that no double-spend has been released in the past d seconds, as explained at the end of Appendix A.

How does the delay diameter affect acceptance times? Given that block creation rate is high, most of the waiting time for acceptance is dominated by the block propagation delay. Fig. 2 depicts the transaction acceptance times of SPECTRE, for various values of the delay diameter d, and for different security thresholds ϵ. Note that, unlike the Nakamoto Consensus, d affects the acceptance time of transactions but not their security.

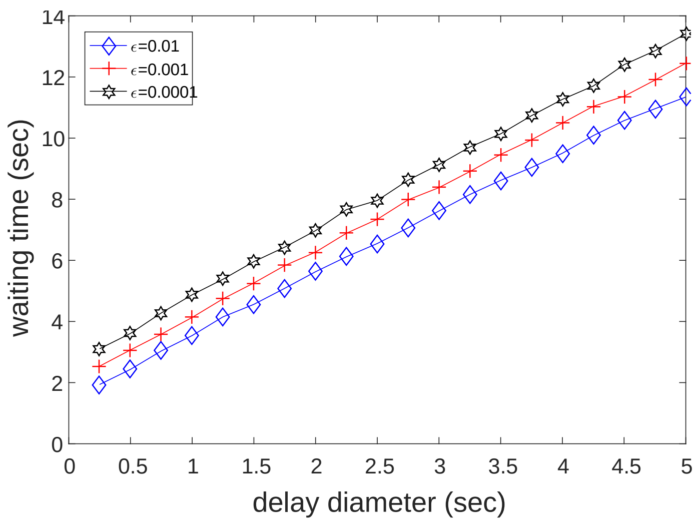

Fig. 2: The average time for $CheckRobustAccept$ to return ACCEPT, assuming there's no visible double-spend, for $\lambda = 10$ blocks per second and $\alpha = 0.25$.

How does the block creation rate affect acceptance times? Fig. 3 depicts the acceptance times for various values of the block creation rate λ, under a constant delay $d = 5$ seconds. The graph reaffirms the role of λ in our asymptotic bound: accelerating the block creation process allows for faster acceptance times. For comparison, Bitcoin's block creation rate of $1/600$ implies waiting times that are orders of magnitudes higher (not plotted).

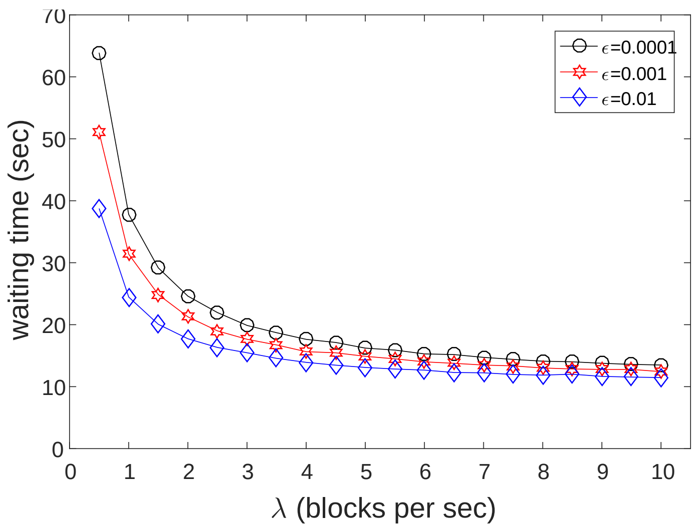

Fig. 3: The average time for $CheckRobustAccept$ to return ACCEPT, assuming there's no visible double-spend, for $d = 5$ seconds and $\alpha = 0.25$.

Can an attacker delay acceptance? We now turn to demonstrate the effect of censorship attacks in which some dishonest nodes publish blocks that do not reference other miners' blocks. Recall that the Weak Liveness property of SPECTRE (Proposition 4) guarantees fast acceptance of transactions that are not visibly double-spent, even in the presence of a censorship attack. However, such an attack still causes some delay in transaction acceptance, but this delay is minor for small attackers. In Fig. 4 we quantify this effect, by comparing the acceptance times in "peace days" to those under an active censorship attack. The parameters here are $d = 5$ seconds, $\lambda = 10$ blocks per second, and $\epsilon = 0.01$. The results display a modest effect of the attack, and they show that in order to delay transaction acceptance by more than 5 to 10 seconds an attacker must possess a significant share of the computational power in the network.

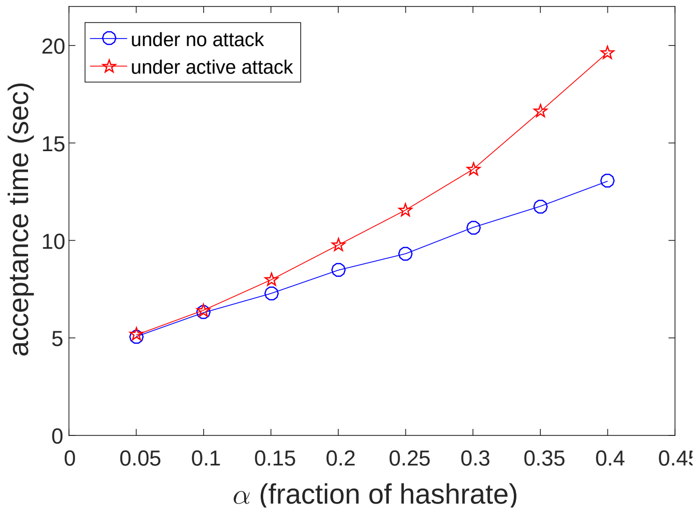

Fig. 4: The average time for $CheckRobustAccept$ to return ACCEPT, assuming there's no visible double-spend, for $d = 5$ seconds, $\lambda = 10$ blocks per second, and $\epsilon = 0.01$, in the presence and in the absence of a censorship attack.

How does ϵ decrease for various sizes of the attacker? Once an honest node ϵ-accepts a transaction, there's still a small risk (ϵ) that it would eventually be rejected. We show that the probability of this event vanishes quickly, even for an extremely capable attacker (e.g., with $\alpha = 0.4$ of the hashrate). This is illustrated in Fig. 5, assuming $d = 5$ seconds and $\lambda = 10$ blocks per second (notice that the y-axis is in log scale).

How tight is our security analysis? The analysis on which Alg. 3 relies makes several worst-case assumptions in order to bound the probability of a successful attack, e.g., that the attacker can broadcast blocks to and receive blocks from all nodes without any delay (see Appendix E, mainly Lemmas 12 and 18). Accordingly, the analysis is not tight, and in reality attacks are in fact less likely to succeed. In Fig. 6, we depict the comparison between the analytical bound and two different empirical simulations. In these simulations we explicitly

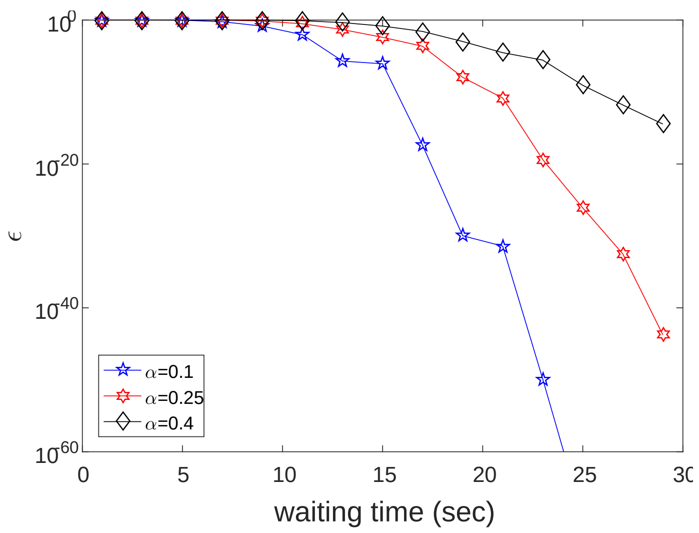

Fig. 5: The probability of a successful double-spending attack, as a function of the waiting time before acceptance, under $d = 5$ seconds and $\lambda = 10$ blocks per second, for $\alpha = 0.1$, 0.25, and 0.4. The probability here is the result of the calculation performed by Alg.3.

generate blocks for the attacker and simulate the optimal double-spending attack. We repeat the experiment 10,000 times for each point in the graph, and measure the empirical success rate. The simulations assume two types of attackers: a worst-case attacker that is able to transmit and receive blocks with no delays, and a more realistic attacker that is connected to other nodes with typical delays. We compared the fraction of successful attacks under these setups to the analytical risk calculated by SPECTRE's policy (Alg. 7).

The results show that the risk considered by SPECTRE's $CheckRobustAccept$ indeed upper bounds the actual risk, and that transactions are even safer than we guarantee formally.

Appendix C
Implementation Details

Minting. In SPECTRE, any block whose target meets the required value $TARGET$ – as will be defined below – receives the same minting reward. If its target is higher than $TARGET$ (i.e., it is solved with an easier difficulty) by a factor of $(1 + \delta)$ at most, then its reward is reduced by the same factor. The parameter δ represents the protocol's tolerance to blocks mined with an outdated difficulty. Thus, if for instance δ is chosen to equal 2, then blocks with a target value of $2 \cdot TARGET$ or $3 \cdot TARGET$ are valid, and their minting rewards are reduced by a factor of 2 or 3, respectively; blocks with a target higher than $3 \cdot TARGET$ are invalid and discarded. We now explain how $TARGET$ is defined and readjusted.

Retargeting. Similarly to Bitcoin and other PoW-based systems, the difficulty of block creation, represented by $TARGET$ (Subsection 3.1), must be occasionally adapted. Varying network

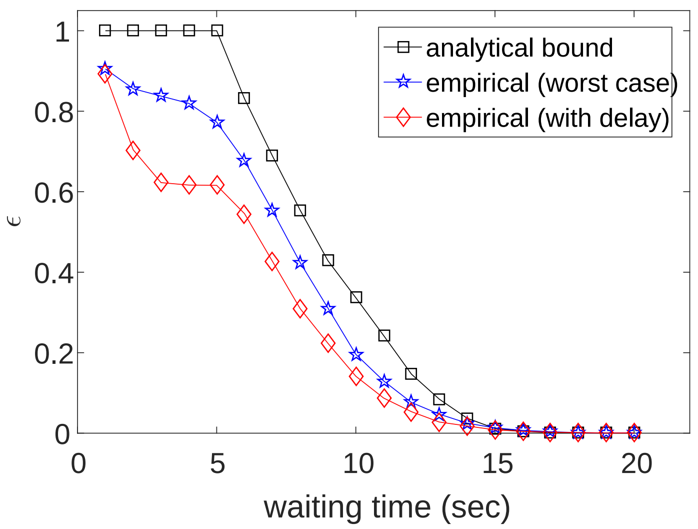

Fig. 6: The analytical vs. empirical probabilities of a successful double-spending attack, as a function of the waiting time before acceptance, under $d = 5$ seconds, $\lambda = 10$, and $\alpha = 0.25$.

conditions, and changes in the amount of computational resources invested in the system, require we limit the number of blocks created per second, to avoid network congestion. In Bitcoin this is done as follows: Every 2016 blocks, the next block – which we call the *reference block* – is mined according to an adjusted difficulty. The new difficulty is obtained by taking the time that elapsed since the previous reference block (using the timestamps written inside each block) and plugging it into the retargeting formula. The output of this formula is the new value of $TARGET$ that the new reference block should be mined with.

We adapt this scheme to SPECTRE: Let x_{n-1} be the previous reference block. Every new block x_n that has the property that $\left|past\,(x_n) \cap \overline{future}\,(x_{n-1})\right| = 2016$,[6] is a candidate to become the new reference block. In case additional candidates exist, we choose the one with the minimal $dist\_gap$, with some arbitrary tie-breaking, where $dist\_gap(b, G) := \min_{K\in\mathbb{N}} gap\,(b, \langle G, b, K\rangle) = 0$. The variable $dist\_gap(b, G)$ represents the minimal K such that adding K votes in favour of b makes its gap equal zero. This guarantees that among a set of candidates to become x_n (satisfying the above property) one and only block would be chosen as the reference block succeeding x_{n-1}. In particular, as explained in Appendix D, an attacker block that was withheld for a while will have a large $dist\_gap$ and will not be eligible as a reference block. Furthermore, an attacker block that was mined before x_{n-1} will not affect the next retargeting, as it cannot belong to $future\,(x_{n-1})$.

The new difficulty, with which the new reference block should be mined, is given again through the formula that uses the time that elapsed between x_{n-1} and x_n to update $TARGET$.

[6]Some new notation is used in these paragraphs: $\overline{future}\,(x) := future\,(x) \cup \{x\}$, and similarly for $\overline{past}\,(x)$ and $\overline{anticone}\,(x)$. In addition, $antipast\,(x) = future\,(x) \cup anticone\,(x)$, and similarly for $antifuture\,(x)$.

The formula should aim for a predefined λ for which nodes are believed to have sufficient bandwidth, e.g., 1 MB per second. This difficulty dictates the difficulty for every block in $antipast\,(x_n) \setminus \overline{anticone}\,(x_{n+1})$, where x_{n+1} is the next reference block. Every block in this set should be mined according to the same difficulty as x_n.

If block $b \in antipast\,(x_n) \setminus \overline{anticone}\,(x_{n+1})$ was solved with an easier difficulty than that dictated by the reference block x_n, then b is still considered valid, provided that its outdated target is at most $(1+\delta)$ of the target of x_n (i.e., a difficulty easier by at most $(1+\delta)$). The parameter δ is the protocol's tolerance threshold. The minting reward of b is reduced by the corresponding factor, as explained above. Blocks whose target exceeds the required one by a factor higher of $(1+\delta)$ are ignored and discarded.

Block headers. In order to incorporate all blocks into the DAG, every block embeds in its header pointers to the hash of previous blocks. No redundancies are permitted, hence only leaf-blocks of $past\,(b)$ should be pointed at by the header of b. The implication of this is that a block's header is of size $\approx 50 + d \cdot \lambda \cdot 32$ Byte. Therefore, there is a limit to the extent at which block size could be reduced and block creation rates increased – at extremely high rates, the overhead of the block header becomes significant relative to the number of included transactions. We note, additionally, that in case the current observable DAG has too many leaves (whether by a rare burst in block creations or by an attacker releasing many outdated blocks), the next block creator can cap the number of leaf-blocks it points at. Blocks left out by this block will later integrate into the DAG, as future blocks will have available space in their headers and will be able to point at these blocks and include them.

Efficient implementation. Our current implementation of SPECTRE uses naïve calculations which are usually inefficient, specifically, cascading the votes all the way to the $genesis$ block. Several efficient implementations are possible. Designing such an efficient implementation requires attention to CPU attacks, in which the attacker exposes peculiar structures of outdated blocks in order to cause other nodes to perform extensive computation. It can be shown that these attacks are highly costly to the attacker. We have an implementation of SPECTRE's procedures that works in $\mathcal{O}(d \cdot \lambda)$ in expectation, compared to the naïve implementation with $\mathcal{O}(|G|^3)$. We leave its full specification, and a proof of the cost of CPU attacks on it, to future work.

Transaction fees. The body of a transaction specifies the amount transferred from the payer to the payee. The transaction-fee specifies the payment from the payer to the miner whose block contains the transaction. We regard these two parts as separate transactions, in the following sense. Assume that $tx \in x$, and denote by $fee(tx, x)$ the transaction representing the fee-payment of tx to the creator of block x. Assume now that two copies of tx appear in two different blocks x, y. Then the body is considered simply as a copy of the same transaction (recall the notation $[tx]$ from Sec. 2), whereas the transactions $fee(tx, x)$ and $fee(tx, y)$ are considered a conflict, i.e., a double-spend. Accordingly, as in the ordinary scheme of SPECTRE, the fee is granted to (the creator of) block x iff $tx \in GetAcceptedTxs(G)$ and x defeats all other blocks that contain tx as well.

This rule can potentially harm miners, in the special case when the relation between x and y does not become robust (SPECTRE does not guarantee robustness if these blocks were published in time proximity and an active attack is taking place). We address this problem by introducing *settlement transactions*. A settlement transaction is a voluntary transaction which both the creators of x and y sign after they observe that their blocks conflict. We denote it $settlement(x, y)$. The interpretation of $settlement(x, y)$ is that the fees from all of (or part of, if the parties involved so choose) the transactions in $x \cap y$ should be divided evenly between blocks x and y. $settlement(x, y)$ essentially overrides $fee(tx, x)$ and $fee(tx, y)$. When $settlement(x, y)$ appears in some block z in the DAG G, it is considered accepted (i.e., a member of $GetAcceptedTxs(G)$) iff $x, y \in past(z)$ and z precedes every block that contains a transaction *spending* $fee(tx, x)$ or $fee(tx, y)$. Therefore, once one party has spent its fee before it belonged to it robustly, it won't be able to settle later (w.h.p.). Miners are therefore advised to wait for their transaction-fee rewards to become robust, or to initiate a settlement, before spending these rewards.

Note that this scheme can be used to settle conflicts between blocks of multiple parties simultaneously. Furthermore, the settlement scheme need not be confined to conflicts regarding fees, and can be applied to any double-spend.

Appendix D
Intuition and Examples

In this section we provide some basic explanations and intuitions regarding the operation of SPECTRE. We focus primarily on explaining the ideas underlying Alg. 1 that is at the core of the protocol. We later go on to present examples for simple attacks that shed some light on how resilience is achieved.

Intuition 1 (Vote in favour of visible blocks)**.** If a block x is known by honest participants, their blocks will include it in their past. Given that blocks vote in favour of blocks in their past (over other unknown blocks), and given that honest nodes publish their blocks quickly, hidden attacker blocks lose votes.

Intuition 2 (Majority amplification)**.** Given blocks x, y that contain potential conflicts, blocks that are generated by honest participants after their publication reference both of them in the DAG. According to Alg. 1, these new blocks adopt the vote of the sub-DAG in their past with regards to x and y. Thus, if block x precedes block y, additional votes that support this decision are added, and the attacker will find it more difficult to reverse the vote.

Intuition 3 (Referencing recent blocks is beneficial)**.** Blocks from the past vote according to their future. Thus if an attacker creates a block that does not reference recent blocks, it is at a disadvantage compared to other blocks that do (it loses votes from recent blocks it did not reference and did not "convince").

Intuition 4 (Votes from the past counter pre-mining attacks)**.** Consider an attacker that creates a block y, withholds it, and constructs many blocks on top of it over an extended period of time. After a long while, a conflicting transaction is released to the network, and eventually ends up in some block x. Block y has many blocks (built by the attacker) that reference it. Thus, if only votes from the future are counted, block y would prevail even if x is allowed to accumulate some votes. In SPECTRE, blocks that were created by honest nodes while y was withheld, look to their future for their votes. These will usually vote in favour of x and will usually outnumber the attacker blocks that were created when y was withheld (an example of pre-mining appears in Fig. 8).

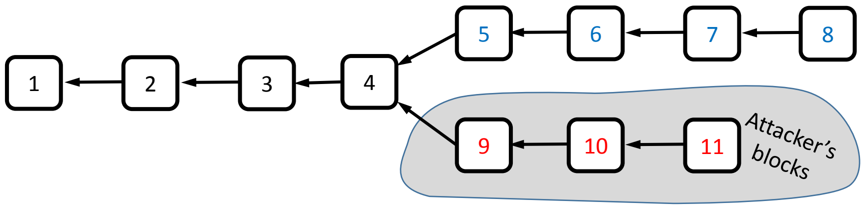

Fig. 7: SPECTRE coincides with the longest-chain rule when it is applied to “simple” chains of blocks. In the depicted DAG, the chain ending at block 8 is longer and would be selected in the longest chain protocol. In SPECTRE each one of the blocks 5,6,7,8 precedes each of the blocks in 9,10,11. Consider for instance blocks 6 and 10 and the pairwise vote that involves them. Blocks 6-8 vote strongly $6 \prec 10$, as they see block 6 in their past but not block 10. Block 5 is a weak voter, as it sees neither 6 nor 10 in its past, hence it votes as the majority of its future (thus voting $6 \prec 10$ as well). For similar reasons, blocks 9-11 all vote $10 \prec 6$. Block 4, at the fork of the two chains, is a weak voters as well, as it sees neither 6 nor 10 in its past; it therefore votes according to the majority of future blocks. As block 4 sees four votes in favour of $6 \prec 10$, and three votes in favour of $10 \prec 6$, it will vote in favour of $6 \prec 10$. Blocks 1-3 similarly vote according to their future, and see an increasing number of votes for $6 \prec 10$, adding their own vote to the result. Thus, the end result is that 6 precedes 10.

A. *Equivalence to longest-chain*

We first demonstrate how SPECTRE coincides with Bitcoin's longest-chain rule, in the case of a “simple” fork between two chains. Consider the DAG illustrated in Fig. 7. In Bitcoin, the longer chain would be selected. Similarly, in the pairwise ordering of SPECTRE, each of the blocks in the longest chain 5,6,7,8 would precede each of the blocks in the shorter one 9,10,11. To see why this is true refer to the caption of the figure.

We now turn to examine two different attack scenarios, which we name *double-spending*, and *censorship*. Recall the requirement from our miner protocol: each miner is required to *(i)* reference recent blocks, and to *(ii)* publish his blocks immediately. Each attack is basically a violation of one of these requirements. In the double-spending attack, the attacker delays the publication of a set of blocks (that includes a conflicting transaction), and in the censorship attack he publishes blocks but “ignores” a certain block and transactions inside it, hoping to convince nodes that it did not secure enough votes, and thus cannot be accepted.

B. Example of a double-spending attack

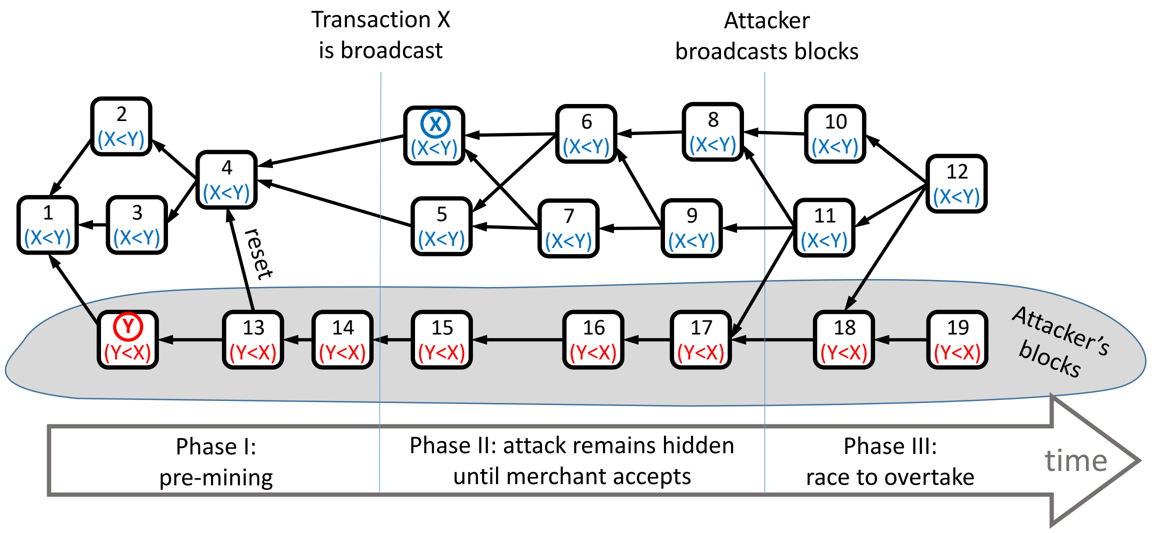

Fig. 8: An example of the voting procedure on a DAG in which a double-spending attack is (unsuccessfully) attempted. *Block x and blocks 6-8 vote strongly $x \prec y$ as they only see x in their past, and not y. Similarly, block y and blocks 13-19 vote strongly $y \prec x$. In the DAG which is the past of block 11, each of the blocks 1-5 sees more $x \prec y$ voters in its future than $y \prec x$ voters, hene each of them votes $x \prec y$. Block 11 votes (as the virtual block of its past votes), according to the majority in its past, thus it too votes $x \prec y$. A similar argument goes for the the vote of 11 and 12. Finally, aggregating the vote of all blocks in the DAG, x got more votes hence $x \prec y$.*

Fig. 8 depicts an (unsuccessful) double-spending attack. The attack is composed of three main phases:

Phase I: Pre-mining. In phase I, the attacker begins building blocks and withholding them from the network. The first block that is constructed (named block y) contains a transaction that will later conflict with the transaction sent to the honest nodes. Blocks built by the attacker ideally form a chain, and due to the voting rules in SPECTRE, will all vote $y \prec x$ (blocks y,13,14). Blocks built by the honest node are unaware of y (and also of x that is yet to be created), and will eventually vote according to the majority of future votes. During this phase, attacker blocks reference honest blocks that are built (in hopes of later convincing them to vote $y \prec x$). After some time, the attacker transmits the transaction to the network, and proceeds to phase II.

Notice that at the exact time that phase I ends, the attacker has more blocks above block 4 than honest nodes have, so it starts at an advantage: it will more easily sway the vote of block 4 towards $y \prec x$ (this advantage later disappears as honest nodes typically build blocks faster than the attacker).

Phase II: Waiting for acceptance. The attacker now continues to build blocks in secret. If he publishes his blocks, then his conflicting transaction will be visible to all, and the double-spend will be detected. Instead, he waits for block x to receive sufficient weight (in the form of blocks built on top of it) so that the recipient of the transaction in x accepts it, and provides the attacker with some service or product. During this phase, attacker blocks that are created (blocks 15-17) vote $y \prec x$, as the attacker is careful to have them reference only his secret chain, and never indirectly reference block x. Honest blocks created during this phase will typically vote $x \prec y$ since y is hidden from them. Some small number of blocks (created before x propagated to the whole network – block 5 in this example) do not reference x, and so will vote according to the result of future votes.

Phase III: Race to overtake. Once x was ϵ-accepted by the victim, the attacker wishes to publish his secret blocks in hopes of causing his conflicting transaction in y to precede x. In this case, the transaction in x will be considered rejected, and the payment will be canceled (leaving the attacker with an item he did not pay for). He publishes his secret chain (which from this point on is referenced by honest nodes), and continues to build upon it. Blocks that he builds, again do not reference x, and so they vote $y \prec x$, supporting his goal. New honest nodes are for the first time exposed to the conflicting transaction y, and thus vote according to the result in the sub-DAG in their past.

Why the attack fails. First, notice that the attacker in the above example creates fewer blocks in each phase than the honest nodes. This will usually be the case if attackers have less computational power than all honest nodes. "Poisson bursts" in block creation by the attacker are possible, and this will allow him to overtake the network, but these are less likely if the attack lasts for a long period of time. The defender can control the length of phase II by waiting a long while before accepting the transaction, which decreases the probability of such bursts. If phase II is long enough, x will have more votes in this period than y. Weak blocks in the past of x will then vote in favour of x, according to this majority. Such blocks that look at their future begin a cascade: each block further in the past adds a vote that agrees with the majority of future blocks and thus strengthens the decision. The greater the majority obtained in Phase II, the less likely it is that the attacker will be able to catch up from behind in Phase III. The attack therefore depends heavily on successfully swaying the votes of blocks that were created just before x (e.g., block 4).

It is important to note that an attacker that creates more blocks in expectation than the honest network will succeed in carrying out this attack. The blocks voting $y \prec x$ would outnumber those who vote to the contrary. Hence the 50% threshold in Theorem 1.

C. Example of a censorship attack

Fig. 9 depicts an (unsuccessful) censorship attack. The attack is composed of a single main phase during which an attacker creates his own blocks, publishes them instantly, but also ignores (and does not reference) recent blocks created by the honest network. The figure depicts (in stage I on the left side) the current state of the blockchain (where all blocks are published at this point). An honest participant that then observes the network and wishes to tell if a transaction in block

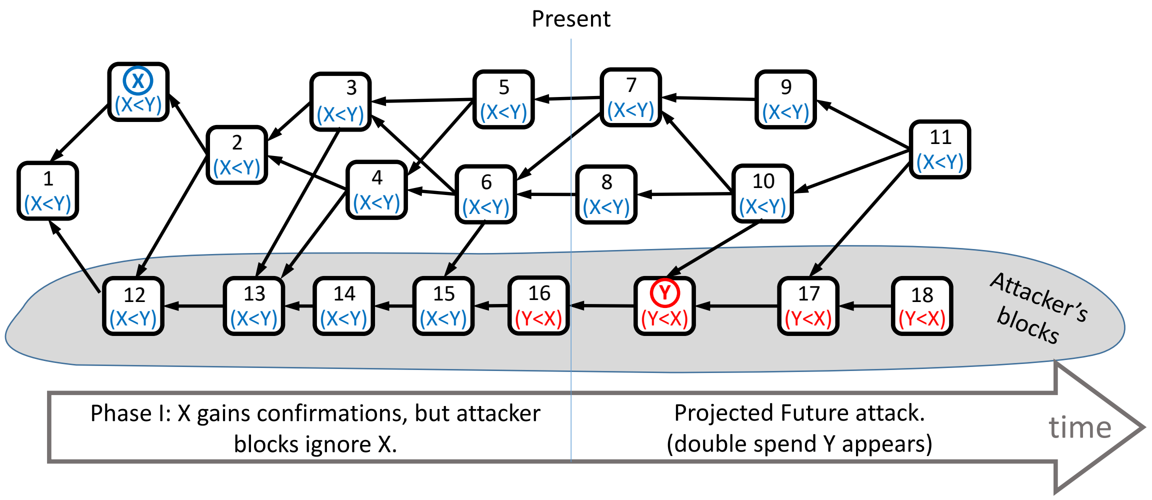
Present
X
(X<Y)
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
(Y<X)
Y
17
18
Attacker's blocks
Phase I: X gains confirmations, but attacker blocks ignore X.
Projected Future attack. (double spend Y appears)
time

Fig. 9: An example of the voting procedure on DAG in which an unsuccessful censorship attack is depicted. *The left side depicts the current state of the block DAG. The right-hand side depicts its likely future development. Blocks 12-16 do not add strong votes to x. Can they be convinced to vote for block y when it appears? Will they further sway other blocks in their past? The vote of each block in this projected future are depicted: Blocks 2-9 vote strongly for x as they see it in their past (but not y). Blocks 17-18 similarly vote strongly for y. Block 16 is indeed convinced to vote for y as more blocks in its future vote for y than for x. Blocks 1, 12-15 vote for x. They each see more votes in favour of x than votes in favour of y in their future. Blocks 10-11 see more $x \prec y$ voters in their past when they make a recursive call.*

x is secure, can see a large number of blocks that do not reference x. These blocks are not guaranteed to vote in favour of x. An attacker may later insert a conflicting transaction y and add blocks atop it (this projected attack is depicted on the right-hand side of the figure). These may potentially sway previously created attacker blocks to vote against x.

The main risk from the censorship attack is that merchants, upon seeing the attacker's blocks, will consider transactions in block x not sufficiently secure. This could potentially delay the acceptance of transactions forever. Our analysis of SPECTRE shows that even in this case the merchants accept transactions quickly (and securely).

Appendix E
The Complete Proof of Theorem 1

Theorem 1. *SPECTRE's $GetAcceptedTxs$ procedure (Algorithm 2) satisfies Property 1. In addition, for any $\alpha < 0.5$ and $d^v \geq D$, SPECTRE's $CheckRobustAccept$ procedure (Algorithm 6) satisfies Properties 2-4. Moreover, the expected waiting times $\mathbb{E}[\tau - t]$ and $\mathbb{E}[\psi - t]$ increase logarithmically in $1/\epsilon$, and similarly $\mathbb{E}[\psi - t]$ in $1/\epsilon'$.*

A. Additional notation

- $node(b) \in \mathcal{N}$ – the node that created block b, $time(b)$ – the time of its creation, $publication(b)$ – the time at which $node(b)$ *begun* the transmission of b to some other honest node, $received^v(b)$ – the time at which node v received b.
- $\overline{future}\,(x) := future\,(x) \cup \{x\}$, and similarly for $\overline{past}\,(x)$ and $\overline{anticone}\,(x)$. In addition, $antipast\,(x) = future\,(x) \cup anticone\,(x)$, and similarly for $antifuture\,(x)$
- $\widehat{\mathcal{E}}^u_s(x, y) :=$ the event where $vote_{y,x}\,(virtual\,(G^u_s)) = +1$.
- $\mathcal{E}^u_s(x, y, \epsilon) :=$ the event where $Risk\,(G^u_s, x, y) < \epsilon$.
- $\widehat{\mathcal{A}}^u_s(tx) :=$ the event where $tx \in GetAcceptedTxs(G^u_s)$.
- $\mathcal{A}^u_t(tx, \epsilon) :=$ the event where $CheckRobustAccept(G^u_t, tx, \epsilon) = ACCEPT$.
- $\mathcal{E}^{all}_{t\to\infty}(x, y, \epsilon) :=$ the event $\cap_{u\in honest} \cap_{s\in(t,\infty)} \mathcal{E}^u_s(x, y, \epsilon)$, and similarly for $\widehat{\mathcal{E}}^{all}_{t\to\infty}(x, y)$, $\mathcal{A}^{all}_{t\to\infty}(tx, \epsilon)$, and $\widehat{\mathcal{A}}^{all}_{t\to\infty}(tx)$.
- $past_h\,(z, G) := past\,(z, G) \cap honest$, and similarly for the future and anticone sets.
- $V_{x\prec y}(G) := \{z \in G | z$ is a strong voter w.r.t. (x, y) and $vote_{x,y}\,(z) = -1\}$ ($V_{x\prec y}(G)$ depends on x, y).
- $\mathcal{P}_{oiss}(\delta, j) := e^{-\delta} \cdot \frac{\delta^j}{j!}$.

B. Formal claims

We now take apart Theorem 1 and write a separate proposition for each of the security properties Safety, Progress, and Weak Liveness, and for Consistency. In order to prove Theorem 1, we need to prove the following four propositions:

Proposition 2 (Consistency). *The accepted set is consistent: For any history G,*

1) *if $tx \in GetAcceptedTxs(G)$ and $tx_2 \in inputs\,(tx)$ then $tx_2 \in GetAcceptedTxs(G)$.*
2) *if $tx \in GetAcceptedTxs(G)$ and $tx_2 \in conflict\,(tx)$ then $tx_2 \notin GetAcceptedTxs(G)$.*

Proposition 3 (Safety). *For any $v \in honest$ and time t, if $CheckRobustAccept\,(tx, \epsilon, G^v_t, d^v, \alpha) = ACCEPT$ then, with probability of at least $1 - \epsilon$, there exists a $\tau \geq t$ such that $\forall u \in honest, \forall s \geq \tau : CheckRobustAccept\,(tx, \epsilon, G^u_s, d^u, \alpha) = ACCEPT$, and the expectation of $\tau - t$ is finite.*

Proposition 4 (Progress). *For any $v \in honest$ and time t, if $CheckRobustAccept\,(tx, \epsilon, G^v_t, d^v, \alpha) = ACCEPT$ then, with probability of $1 - \epsilon$ at least, for any ϵ' there exists a ϕ such that $\forall s \geq \phi : CheckRobustAccept\,(tx, \epsilon, G^v_s, d^v, \alpha) = ACCEPT$, and the expectation of $\phi - t$ is finite.*

Proposition 5 (Weak Liveness). *Let t be the current time, and assume that $tx \in G^{pub}_t$. Let $\psi \geq t$ be the earliest time after t at which an honest node ϵ-accepts tx. Then, conditioned on the event where $conflict\,(tx) \cap G^{pub}_\psi = \emptyset$ and on the event where for all $tx_2 \in inputs\,(tx)$, tx_2 remains ϵ-accepted forever (by some honest node), the expectation of $\psi - t$ is finite.*

We will prove Consistency in a later subsection. To each of the last three propositions we write a matching one which regards robustness of blocks (rather than that of transactions).

Proposition 6 (Safety (blocks)). *For any $v \in honest$, if $Risk\left(x, y, G_t^v\right) < \epsilon$ then, with probability of $1-\epsilon$ at least, there exists a τ such that $\forall u \in honest, \forall s \geq \tau : Risk\left(x, y, G_s^u\right) < \epsilon$, and $\mathbb{E}[\tau - t] < \infty$.*

Proposition 7 (Progress (blocks)). *For any $v \in honest$, if $Risk\left(x, y, G_t^v\right) < \epsilon$ then, with probability of $1-\epsilon$ at least, for any $\epsilon' < \epsilon$ there exists a ϕ such that $\forall s \geq \phi : Risk\left(x, y, G_s^u\right) < \epsilon$, and $\mathbb{E}[\phi - t] < \infty$.*

Proposition 8 (Weak Liveness (blocks)). *Let t be the current time, and assume that $x \in G_t^{pub}$. Let ψ be the first time s at which for some honest node v: $Risk\left(x, y, G_s^v\right) < \epsilon$. Then, conditioned on the event where $y \notin G_\psi^{pub}$, the expectation of $\psi - t$ is finite.*

We prove the correctness of these propositions in separate subsections below. But first, we begin with three simple lemmas.

C. Basic properties

The two following lemmas are immediate from lines 7-14 of Algorithm 1.

Lemma 9. *Topological relations are unanimously agreed: If $G = (C, E)$ is a block DAG, and $(y, x) \in E$, then $\forall z \in G : vote_{x,y}\left(z, G\right) = -1$.*

Lemma 10. *A block's vote regarding block(s) in its past depends only on its past, hence remains fixed forever: Let G_1 and G_2 be two block DAGs, and assume $x, y, z \in G_1 \cap G_2$. If $\{x, y\} \cap \overline{past}\left(z\right) \neq \emptyset$ then $vote_{x,y}\left(z, G_1\right) = vote_{x,y}\left(z, G_2\right)$.*

Accordingly, we say that z is a strong voter w.r.t the pair (x, y) if $z \in \overline{future}\left(x\right) \cup \overline{future}\left(y\right)$, and otherwise it is a weak voter.

The following Lemma shows that the vote of the $genesis$ coincides with the vote of the virtual block. Intuitively, the $genesis$ votes according to the majority vote in the DAG excluding itself, and amplifies this majority, which in turn dictates the virtual block's vote.

Lemma 11. *$genesis$'s vote is the final vote: $vote\left(virtual\left(G\right)\right) = vote\left(genesis, G\right)$.*

Proof. Suffice it show that if $vote_{x,y}\left(genesis, G\right) \geq 0$ then $vote_{x,y}\left(virtual\left(G\right)\right) \geq 0$. If (x, y) are related topologically then by Lemma 9 all votes agree unanimously on their ordering, and in particular $vote_{x,y}\left(virtual\left(G\right)\right) = vote_{x,y}\left(genesis, G\right) \geq 0$. Otherwise, it cannot be the case that x or y are the $genesis$ block, hence the $genesis$ is a weak voter, and by line 14 we obtain

$$vote_{x,y}\left(virtual\left(G\right)\right)=\widetilde{sgn}\left(\sum_{z\in G}vote_{x,y}\left(z,G\right)\right)= \tag{1}$$

$$\widetilde{sgn}\left(vote_{x,y}\left(genesis,G\right)+\sum_{z\in future(genesis,G)}vote_{x,y}\left(z,G\right)\right)\geq \tag{2}$$

$$\widetilde{sgn}\left(\sum_{z\in future(genesis,G)}vote_{x,y}\left(z,G\right)\right)=vote_{x,y}\left(genesis,G\right)\geq 0, \tag{3}$$

hence $vote_{x,y}\left(virtual\left(G\right)\right)\geq 0$. □

D. Overview of the proof of Safety (blocks)

Proposition 6 claims essentially that all nodes will forever agree on the (robustness of) the order $x\prec y$, provided that it was sufficiently robust in the DAG observed by some honest node. This is the main and most involved part of the proof. The rest of the propositions follow from it, and their proofs are rather self explanatory. Since its proof is involved and occasionally technical, we begin with an overview of its structure.

In order to simplify the analysis, we need to make some worst case assumptions regarding the behaviour of the attacker. Lemma 18 proves that these are indeed worst case assumptions, namely, that they indeed represent the optimal attack. The vote of each block under our modification is denoted $p\_vote\left(\right)$, a notion which we describe formally in Subsection E.6.

In the next central lemma we show that, provided that the aggregate vote in $future\left(x\right)$ is sufficiently biased in favour of $x\prec y$, the $genesis$ block – hence the virtual block (by Lemma 11) – will vote $x\prec y$. This proves that, roughly speaking, *the vote of recent weak voters cascades through the DAG and convinces older weak blocks, forming thus the genesis's vote*. The way we prove this is by choosing a specific weak voter z_{late}(in case x is an honest block, $z_{late}=x$), and making sure that its vote is sufficiently robust so as to guarantee that (i) it will not be reversed, and (ii) it will cascade all the way to the $genesis$. Consequently, a successful attack (namely, a reversal of $x\prec y$ in the DAG observed by some honest node) requires that the attacker add more blocks to $future\left(z_{late}\right)$ than the honest network adds (up to some additive term), in some time interval.

The following lemma formalizes these observations. It uses some parameters (h, j, etc.) that only an oracle can have full knowledge of. We will later show how in reality a node can infer the robustness of block relations without having access to these parameters.

Lemma 12. *Let $t\geq publication(x)+2\cdot d$. Let z_{late} be the latest block in $\overline{past_h}\left(x\right)$. Denote:*

- $h:=\left|anticone_h\left(z_{late},G_t^{oracle}\right)\right|$
- $j:=\left|future_h\left(z_{late},G_t^{oracle}\right)\setminus future_h\left(x,G_t^{v}\right)\right|$
- $m:=\left|future_a\left(z_{late},G_t^{oracle}\right)\setminus future_a\left(x,G_t^{v}\right)\right|$

- $k_1 := \left|G^{oracle}_{[t-d,t]} \cap honest\right|$
- $l := \max_{z \in G^{oracle}_t \cap honest} \left\{\left|future_a\left(z, G^u_{time(z_{late})}\right)\right| - \left|future_h\left(z, G^u_{time(z_{late})}\right)\right|\right\}$
- $g := \sum_{z \in \overline{future}(x, G^v_t)} vote_{y,x}\left(z, G^v_t\right)$

Then,

$$\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)^{\complement} \subseteq \left\{\exists s \geq t, \exists u \in honest \text{ s.t. } \left|G^u_{[t,s]} \cap attacker\right| \geq \left|G^u_{[t,s]} \cap honest\right| + g - 2\cdot h - j - k_1 - l - m\right\}.$$

Given the result of the previous lemma, we can upper bound the probability that the order of $x \prec y$ will be reversed. This result resembles the conventional analysis of Bitcoin's security: The greater number of blocks currently pointing at x (and in SPECTRE: voting for $x \prec y$), the less likely it is that the attacker will be able to win the block-count race and reverse the decision.

Lemma 13. *Given the parameters of Lemma 12,*

$$\Pr\left(\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)^{\complement}\right) \leq \sum_{h'=0}^{\infty} \mathcal{P}_{oiss}(d\cdot(1-\alpha)\cdot\lambda, h')\cdot \left(\frac{\alpha}{1-\alpha}\right)^{(g-2\cdot h-j-k_1-l-m-h')^+}.$$

An ordinary node does not typically know for sure the values of the parameters assumed in Lemma 12. The next corollary shows that the result of that lemma (and the one that follows) applies when replacing these parameters with proper bounds thereof. We will later discuss how a node can obtain such bounds.

Corollary 14. *If*

- $j \geq \left|anticone_h\left(x, G^{oracle}_t\right)\right|$
- $l \geq \max_{z \in G^{oracle}_t \cap honest} \left\{\left|future_a\left(z, G^u_{time(z_{late})}\right)\right| - \left|future_h\left(z, G^u_{time(z_{late})}\right)\right|\right\}$
- $n_x \geq future_h\left(x, G^{oracle}_t\right)$
- $g \leq \sum_{z \in \overline{future}(x, G^v_t)} vote_{y,x}\left(z, G^v_t\right)$.

Then

$$\Pr\left(\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)\right) \leq \tag{4}$$
$$\sum_{k=0}^{\infty}\mathcal{P}_{oiss}((2-\alpha)\cdot d\cdot\lambda,k)\cdot\sum_{h=0}^{\infty}\mathcal{P}_{oiss}(d\cdot(1-\alpha)\cdot\lambda,h)\cdot$$
$$\sum_{m=0}^{\infty}\binom{n_x+j+h+m-1}{m}\cdot(1-\alpha)^{n_x+j+h}\cdot\alpha^{m}\cdot$$
$$\left(\frac{\alpha}{1-\alpha}\right)^{(g-2\cdot h-k-j-l-m)^+}$$

We adjust the above results to the case where some blocks in $future\,(x,G)$ are known to belong to the attacker. Here we assume that this knowledge is granted to us by a hypothetical oracle. Later on, we will see how attacker blocks are recognized by Algorithm 3, w.h.p.

Corollary 15. *If in addition to the assumptions of Corollary 14 we assume that* $M \leq |future_a\,(x,G^v_t)|$, *then*

$$\Pr\left(\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)\mid |future_a\,(x,G^v_t)|\geq M\right)\leq$$
$$\sum_{k=0}^{\infty}\mathcal{P}_{oiss}((2-\alpha)\cdot d\cdot\lambda,k)\cdot\sum_{h=0}^{\infty}\mathcal{P}_{oiss}(d\cdot(1-\alpha)\cdot\lambda,h)\cdot \tag{5}$$
$$\left(\sum_{m'=M}^{\infty}\binom{n_x+j+h+m'-1}{m'}\cdot(1-\alpha)^{n_x+j+h}\cdot\alpha^{m'}\right)^{-1}\cdot$$
$$\sum_{m=M}^{\infty}\binom{n_x+j+h+m-1}{m}\cdot(1-\alpha)^{n_x+j+h}\cdot\alpha^{m}\cdot$$
$$\left(\frac{\alpha}{1-\alpha}\right)^{(g-2\cdot h-k-j-l-(m-M))^+}.$$

We denote the RHS of this inequality by $f_{post\_mine}\,(n_x,g,j,l,M)$.

So far, our analysis assumed that we are given some proper bounds over the parameters from Lemma 12. Lemmas 22, 27, and 29 show how to appropriately bound these parameters. For each of these parameters, a separate error function is defined, which upper bounds the probability that it does not serve as a correct bound. These error functions deteriorate exponentially fast, by Lemmas 23, 28, and 30. Algorithm 3 aggregates these error functions into the total risk that it outputs.

The parameters are:

- l – the pre-mining lead that the attacker obtained before the publication of x, with error function $f_{pre\_mine}(l(G^v_t))$, calculated numerically in Subsection E.6.1
- n_x – the number of honest blocks in $future\,(x,G^v_t)$, with error function $f_{post\_pub}\,(|future\,(x,G^v_t)|)$, defined in Corollary 27 (Inequality (52)), and

- j – the number of honest blocks created after $time(x)$, with error function $f_{pre\_pub}\left(n_j(G_t^v)\right)$, defined in Lemma 29 (Inequality (54)).

While we have previously shown that n_x properly counts all honest blocks, we now show that it does successfully exclude almost all attack blocks. Without such a guarantee, weaker attackers would have been able to publish their blocks and delay acceptance indefinitely.

Lemma 16. *Conditioned on the event $\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)$, there exists a time $\tau \in [t,\infty)$ such that $\forall s \geq \tau$: $M(oracle^u, s) \geq \left|future_a\left(x, G_s^{oracle^u}\right) \cap G^{oracle}_{[t,s]} \setminus V_{x\prec y}(G_s^{oracle^u})\right| - m^*$, for some m^* that remains fixed after τ (and with $\mathbb{E}[m^*]$ determined by the events up to time t).*

The above analysis (particularly Lemma 12) has upper bounded the probability that the attacker would be able to reverse the relation $x \prec y$. We now show that, conditioned on the order remaining $x \prec y$, the error function f_{post_mine} (which upper bounds the probability of this order ever reversing) vanishes as well, which in turn implies that their order would be considered robust by all honest nodes.

Lemma 17. *There exists a $\psi \in [t,\infty)$ such that $\Pr\left(\mathcal{E}^{all}_{t\to\infty}(x,y,\epsilon)^{\complement} \mid \mathcal{E}_t^v(x,y,\epsilon)\right) < \epsilon$. Moreover, $\mathbb{E}\left[\psi - t\right] < \epsilon$.*

We have thus shown that if the output of Algorithm 3, as run by some honest node, was smaller than ϵ then with probability of at least $1-\epsilon$, any honest node running Algorithm 3 (after some time) will get a result smaller than ϵ.[7] This completes the proof of Safety w.r.t. blocks.

E. Proof of Consistency

Proof. ***Part I:*** We first prove that for any DAG G, and for any $tx_1, tx_2 \in T$: if $tx_2 \in inputs\,(tx_1)$ and $[tx_1] \cap GetAcceptedTxs(G) \neq \emptyset$ then $[tx_2] \cap GetAcceptedTxs(G) \neq \emptyset$.

Assume $tx_2 \in inputs\,(tx_1)$ and $[tx_1] \cap GetAcceptedTxs(G,G) \neq \emptyset$ and let $tx_1 \in [tx_1] \cap GetAcceptedTxs(G,G)$. Consider the iteration of the second loop (line 4) over $tx = tx_1$. As $tx \in GetAcceptedTxs(G,G)$ it must be the case that during this iteration the algorithm has reached line 14. This means that for any $[tx_3] \in inputs\,(tx_1)$ it hasn't visited line 13; in particular for $[tx_3] = [tx_2]$, the condition $[tx_2] \cap GetAcceptedTxs\,(G, past\,(z_1)) = \emptyset$ has failed, i.e., $[tx_2] \cap GetAcceptedTxs\,(G, past\,(z_1)) \neq \emptyset$. To see that $GetAcceptedTxs\,(G, past\,(z_1)) \subseteq GetAcceptedTxs\,(G,G)$ observe that (i) during the run of the algorithm no transaction is ever removed from TX, and that (ii) for any $z_1 \in G \cap subG$, the operations (in lines 4-14) of $GetAcceptedTxs\,(G, subG)$ and $GetAcceptedTxs\,(G,G)$ are identical; thus any addition of a transaction in line 14 in $GetAcceptedTxs\,(G, subG)$ occurs in $GetAcceptedTxs\,(G,G)$ as well. In particular, $[tx_2] \cap GetAcceptedTxs\,(G,G) \neq \emptyset$.

Part II: We now prove that for any DAG G, and for any $tx_1, tx_2 \in T$: if $tx_2 \in conflict\,(tx_1)$ and $[tx_1] \cap GetAcceptedTxs(G,G) \neq \emptyset$ then $[tx_2] \cap GetAcceptedTxs(G) = \emptyset$.

[7] ϵ here simply represents a value greater than $f_{pre\_mine} + f_{pre\_pub} + f_{post\_pub} + f_{post\_mine}$.

Assume that $tx_2 \in conflict(tx_1)$ and $[tx_1] \cap GetAcceptedTxs(G, G) \neq \emptyset$ and let tx_1 be an element in the latter intersection. Assume by way of negation that there exists a $tx_2 \in [tx_2] \cap GetAcceptedTxs(G, G)$. Then during the iteration of the first loop (line 3) over some instantiation z_1^1 of z_1 such that $tx_1 \in z_1^1$, and of the second loop (line 4) over tx_1, the algorithm has reached line 14. In particular, it did not reach line 10, hence $z_1^2 \notin past\left(z_1^1\right)$. For the symmetrical argument, $z_1^1 \notin past\left(z_1^2\right)$, which implies that $z_1^2 \in anticone\left(z_1^1, G\right)$ (and $z_1^2 \in anticone\left(z_1^1, G\right)$). Now, either $vote_{z_1^1, z_1^2}(virtual(G)) \geq 0$ or $vote_{z_1^2, z_1^1}(virtual(G)) \geq 0$. Either way, line 8 was reached by either the run on tx_1 or the run on tx_2, which contradicts the assumption that both runs reached line 14. □

F. Proof of Safety (blocks)

In order to simplify the analysis, we need to make some worst case assumptions regarding the behaviour of the attacker, namely, that it publishes all of its blocks immediately after time t (which represents the time at which some honest node accepted the transaction), and that before $time(x)$ its blocks point at all available blocks. These assumptions essentially modify the DAG (in case the attacker does not carry out the optimal attack scheme). We need to prove that these modifications indeed represent the worst case. To this end we use the notion of a *pseudo-vote*. A pseudo-vote begins by first explicitly defining and *fixing* the pseudo-vote of some blocks, which we call the *initial pseudo-voters*. Then we define the pseudo-vote of the rest of the blocks as in Algorithm 1. In more detail, we replace in Algorithm 1 the $vote()$ notation by the $p\_vote()$ notation, and whenever the algorithm references $p\_vote(c)$ of an initial pseudo-voter c, we refer to its fixed predetermined value. Thus, the pseudo-vote of an initial pseudo-voter might change the pseudo-vote of other blocks.

Lemma 18. *Let $x, y \in G = (C, E)$ such that $G_t^v \subseteq G$. Let $G' = (C, E')$ be the DAG resulting from adding the following edges to E:*

1) $\forall z_1 \in G \cap before(time(x)) \cap attacker$, $\forall z_2 \in G \cap before(time(z_1)) \setminus \{z_1\}$: add (z_1, z_2) to E.

2) $\forall z_1 \in G \cap attacker \setminus G_t^v$, $\forall z_2 \in G^{oracle}_{[publication(z_1), \infty)} \cap honest$: add (z_2, z_1) to E.

Let $p\_vote()$ be defined by specifying the following initial pseudo-voters (and their votes):

3) $\forall z \in (G \cap attacker \setminus G_t^v) \cup G \cap before(time(x)) \cap attacker$: $p\_vote_{x,y}(z, G) = +1$.

*Then $vote_{x,y}(virtual(G), G) \leq$
$p\_vote_{x,y}(virtual(G'), G')$.*[8]

Importantly, we assume here that blocks in $G \cap attacker$ break ties in favour of $y \prec x$.

Proof. ***Part I:*** Assume by way of negation that $vote_{x,y}(virtual(G), G) = +1$ yet nonetheless $p\_vote_{x,y}(virtual(G), G') = -1$ (observe that this is the only case in which the claim can fail, by definition, as a virtual vote cannot take the value of 0).

[8]Note that $virtual(G) = virtual(G')$, as they share the same vertex-set.

Let b be a block in $\overline{future}\left(x,G\right)\cup\left\{virtual\left(G\right)\right\}$ such that $p\_vote_{x,y}\left(b,G'\right)=-1$. b cannot belong to $G\setminus G_t^v$ or to $G\cap before(time(x))\cap attacker$, because blocks in these sets have a pseudo-vote of $+1$. Let z be a block in $past\left(b,G\right)$. Since $b\notin\left(G\setminus G_t^v\right)\cup\left(G\cap before(time(x))\cap attacker\right)$, there exists in G' a path from b to z that passes through an edge (z_2,z_1) satisfying the conditions of the second modification to G, and through an edge (z_1',z_2') satisfying those of the first modification. In particular, $time(z_2)\geq publication(z_1)\geq t-d$, and $time(x)\geq time(z_2')\geq time(z_1')$. As b and z are the end-vertices of this path, $time(b)\geq time(z_2)\geq t-d\geq publication(x)+d\geq time(x)+d\geq time(z_1')+d\geq time(z)+d$. Since z_2 is honest, $z\in past\left(z_2,G\right)$, hence $z\in past\left(b,G\right)$. Combined with $E\subset E'$ we obtain: $past\left(b,G'\right)=past\left(b,G\right)$.

Part II: Let b be the earliest block in $\overline{future}\left(x,G\right)\cup\left\{virtual\left(G\right)\right\}$ for which $vote_{x,y}\left(b,G\right)=+1$ but $p\_vote_{x,y}\left(b,G'\right)=-1$, and let z be the latest block in $antifuture\left(x,G'\right)$ for which $vote_{x,y}\left(z,past\left(b,G\right)\right)>p\_vote_{x,y}\left(z,past\left(b,G'\right)\right)$. If such a z exists then, similarly to the previous part, we know that $past\left(z,G'\right)=past\left(z,G\right)$; this proves that z is a weak voter both in G and in G', hence that its pseudo-vote is the sign of the sum of pseudo-votes in its future.[9]

To see that such a z indeed exists, observe that the $genesis$ satisfies these conditions: By Lemma 11 $vote_{x,y}\left(b,G\right)=vote_{x,y}\left(virtual\left(past\left(b,G\right)\right)\right)=+1$ implies $vote_{x,y}\left(genesis,past\left(b,G\right)\right)\geq0$, and in a similar way $p\_vote_{x,y}\left(b,G'\right)=p\_vote_{x,y}\left(virtual\left(past\left(b,G'\right)\right)\right)=-1$ implies that $p\_vote_{x,y}\left(genesis,past\left(b,G'\right)\right)=-1$.[10]

Part III: By the choice of z, if $z'\in future\left(z,past\left(b,G'\right)\right)$ is weak with respect to (x,y) then $vote_{x,y}\left(z',past\left(b,G\right)\right)\leq p\_vote_{x,y}\left(z',past\left(b,G'\right)\right)$. Moreover, by the choice of b, if $z'\in past\left(b,G'\right)$ is strong w.r.t. (x,y) and $vote_{x,y}\left(z',past\left(b,G\right)\right)=+1$ then $p\_vote_{x,y}\left(z',past\left(b,G'\right)\right)=+1$. All in all, we have that for all $z'\in future\left(z,past\left(b,G'\right)\right)$,

[9]It cannot be the case that $y\in past\left(z,G'\right)$, because we know that z is not an initial pseudo-voter (as its pseudo-vote is -1), and therefore the pseudo-vote procedure would have assigned its pseudo-vote to be $+1$, because y is in its past but x is not in it past, in the same way the ordinary procedure does.

[10]The equality $p\_vote_{x,y}\left(b,G'\right)=p\_vote_{x,y}\left(virtual\left(past\left(b,G'\right)\right)\right)$ holds because b is either a strong voter w.r.t. (x,y) or the virtual voter.

$vote_{x,y}(z', past(b,G)) \le p\_vote_{x,y}(z', past(b,G'))$. Therefore:

$$\sum_{z' \in future(z, past(b,G))} vote_{x,y}\left(z', past(b,G)\right) \le \tag{6}$$

$$\sum_{z' \in future(z, past(b,G))} p\_vote_{x,y}\left(z', past\left(b,G'\right)\right) \le \tag{7}$$

$$\sum_{z' \in future(z, past(b,G))} p\_vote_{x,y}\left(z', past\left(b,G'\right)\right) +$$

$$\sum_{z' \in future(z, past(b,G') \setminus past(b,G))} p\_vote_{x,y}\left(z', past\left(b,G'\right)\right) = \tag{8}$$

$$\sum_{z' \in future(z, past(b,G'))} p\_vote_{x,y}\left(z', past\left(b,G'\right)\right). \tag{9}$$

The last equality follows from $future(z, past(b,G)) \subseteq future(z, past(b,G'))$, which holds because $E \subseteq E'$. The inequality in (7) holds because if some z' has been added (by transforming G into G') to the future of some honest block, then z' must belong to the attacker, hence $p\_vote_{x,y}(z', past(b,G')) = +1 > 0$.

Part IV: Consequently, since z is a weak voter with respect to (x,y), (6)-(9) imply that $vote_{x,y}(z, past(b,G)) \le p\_vote_{x,y}(z, past(b,G'))$, which contradicts the choice of z. □

Lemma 12. *Let $t \ge publication(x) + 2 \cdot d$. Let z_{late} be the latest block in $\overline{past_h}(x)$. Denote:*

- $h := \left|anticone_h\left(z_{late}, G_t^{oracle}\right)\right|$
- $j := \left|future_h\left(z_{late}, G_t^{oracle}\right) \setminus future_h\left(x, G_t^v\right)\right|$
- $m := \left|future_a\left(z_{late}, G_t^{oracle}\right) \setminus future_a\left(x, G_t^v\right)\right|$
- $k_1 := \left|G_{[t-d,t]}^{oracle} \cap honest\right|$
- $l := \max_{z \in G_t^{oracle} \cap honest} \left\{\left|future_a\left(z, G_{time(z_{late})}^u\right)\right| - \left|future_h\left(z, G_{time(z_{late})}^u\right)\right|\right\}$
- $g := \sum_{z \in \overline{future}(x, G_t^v)} vote_{y,x}\left(z, G_t^v\right)$

Then,

$$\widehat{\mathcal{E}}_{t\to\infty}^{all}(x,y)^{\complement} \subseteq \Big\{\exists s \ge t, \exists u \in honest \text{ s.t. } \left|G_{[t,s]}^u \cap attacker\right| \ge \left|G_{[t,s]}^u \cap honest\right| + g - 2 \cdot h - j - k_1 - l - m\Big\}. \tag{10}$$

Proof. ***Part I:*** In the proof below we make the following assumption: Any attacker-block z created before $time(x)$ always votes in favour of $y \prec x$ (even if it is supposed to vote otherwise according to Algorithm 1). We further assume that any such z satisfies $\overline{past}(z) = G_{time(z)}^{oracle}$, i.e., it points at all blocks available at the time of its creation. Finally, we assume that the attacker releases all of his blocks to all nodes in $honest \setminus \{v\}$ precisely at time t and onward. The previous lemma implies that these are indeed worst case assumptions: Take G to be any G_s^u. Then, what the lemma shows is that as long as $p\_vote_{x,y}\left(virtual\left(G_s^u\right)\right) = -1$, also $vote_{x,y}\left(virtual\left(G_s^u\right)\right) = -1$ (under the worst case assumption that ties are always broken in favour of y).[11] The analysis below applies, formally, to $p\_vote\left(\right)$ as formalized in the prevoius lemma (specifically in (3)). Nevertheless, now that the argument has been formally made, we omit this notation henceforth.

Part II: Let us look at the following chain of implications:

$$\begin{aligned} &vote_{x,y}\left(virtual\left(G_s^u\right)\right) \geq 0 \Rightarrow vote_{x,y}\left(genesis, G_s^u\right) \geq 0 \Rightarrow \\ &\sum_{z' \in future\left(genesis, G_s^u\right)} vote_{x,y}\left(z', G_s^u\right) \geq 0 \end{aligned} \tag{11}$$

The first implication follows from Lemma 11. The second one follows from the definition of $genesis$'s vote.[12] Thus,

$$\begin{aligned} &\widehat{\mathcal{E}}_{t\to\infty}^{all}(x,y)^{\complement} = \cup_{u\in honest, s\in[t,\infty)} \widehat{\mathcal{E}}_s^u(x,y)^{\complement} = \\ &\left\{\exists u \in honest, \exists s \geq t : vote_{x,y}\left(virtual\left(G_s^u\right)\right) \geq 0\right\} \end{aligned}$$

However, if there exists such an s as the latter event requires, then we can look at the first such s. With respect to it, between t and s all honest votes were in favour of $x \prec y$; this is because for any *honest* block z' with $time(z') \in [t,s)$, $past\left(z'\right) = G_{time(z')}^{node(z')}$, hence $vote\left(z'\right) = vote\left(virtual\left(G_{time(z')}^{node(z')}\right)\right)$, and by the choice of s as the earliest time for which an honest node's DAG's virtual block votes in favour of $y \preceq x$, we know that $vote_{x,y}\left(z', G_s^u\right) = -1$.

Part III: *Below, the notation* $G_{[t_1,t_2]}^u$ *stands for* $G_s^u \cap before\left(t_2\right) \setminus before\left(t_1\right)$.

We claim that for all $z \in \overline{past_h}\left(x\right)$:

$$\begin{aligned} &vote_{x,y}\left(z, G_s^u\right) \leq \widetilde{sgn}\left(\left|G_{[t,s]}^u \cap attacker\right| - \left|G_{[t,s]}^u \cap honest\right|\right. \\ &\left. +2\cdot h + l + k_1 + j + m - g\right). \end{aligned}$$

We prove the claim by a complete induction on $D(z) := \left|future\left(z, \overline{past_h}\left(x\right)\right)\right|$. Assume we have proved the claim for any z with $D(z) < D$. We now prove it for z with $D(z) = D$. If $z = x$ then $vote_{x,y}\left(z, G_s^u\right) = -1$ hence the above inequality is satisfied trivially. Otherwise, z is a weak voter, and $vote_{x,y}\left(z, G_s^u\right)$ is given by the sign of the sum of votes in its future. We decompose these voters into three subsets: members of $future\left(z, G_{time(z_{late})}^u\right)$, members of $future\left(z, G_{[time(z_{late}),t]}^u\right)$, and members of $future\left(z, G_{[t,s]}^u\right)$.

[11]In fact, we use the lemma with a slight modification: The second modification does not apply to all such (z_2, z_1) satisfying the specified conditions, rather to a subset thereof, since blocks created by node v between t and $t+d$ need not point at all attacker blocks in $G \setminus G_s^v$. It is easy to see, however, that the proof of the lemma remains intact (and it remains so when applying the second modification to *any* subset of $(G \cap attacker \setminus G_t^v) \times (G_{[publication(z_1),\infty)}^{oracle} \cap honest)$).

[12]Here we implicitly assume that x and y are not related topologically, which rules out the option that $x = genesis$ or $y = genesis$, hence $genesis$ is weak w.r.t. (x,y). If they are related topologically, the result is trivial, for all votes are then forever unanimous in the same direction (Lemma 9).

1) Members of $future\left(z, G^u_{time(z_{late})}\right)$: By the induction hypothesis we know that all blocks in $future_h\left(z, \overline{past_h}\left(z_{late}\right)\right)$ vote in favour of $x \prec y$, and by the choice of z_{late} we have that $future_h\left(z, G^u_{time(z_{late})}\right) \setminus \overline{past}\left(z_{late}\right) = anticone_h\left(z, G^u_{time(z_{late})}\right)$. Thus, $\sum_{z' \in future_h\left(z, G^u_{time(z_{late})}\right)} vote_{x,y}\left(z', G^u_s\right) \leq 2 \cdot \left|anticone_h\left(z_{late}, G^u_{time(z_{late})}\right)\right| - \left|future_h\left(z, G^u_{time(z_{late})}\right)\right|$. We obtain:

$$\sum_{z' \in future\left(z, G^u_{time(z_{late})}\right)} vote_{x,y}\left(z', G^u_s\right) \leq$$
$$2 \cdot \left|anticone_h\left(z_{late}, G^u_{time(z_{late})}\right)\right|$$
$$- \left|future_h\left(z, G^u_{time(z_{late})}\right)\right| + \left|future_a\left(z, G^u_{time(z_{late})}\right)\right|.$$

2) Members of $future\left(z, G^u_{[time(z_{late}),t]}\right)$:

a) Honest blocks: By *Part I* we have that $future_a\left(z, G^u_{[time(z_{late}),t]}\right) \setminus future_a\left(z_{late}, G^u_{[time(z_{late}),t]}\right) = \emptyset$. This implies that $anticone_h\left(z_{late}, G^u_{[time(z_{late}),time(z_{late})+d]}\right) \supseteq future\left(z, G^u_{[time(z_{late}),t]}\right) \setminus future\left(z_{late}, G^u_t\right)$. We obtain:

$$\sum_{z' \in future_h\left(\boldsymbol{z}, G^u_{[time(z_{late}),t]}\right)} vote_{x,y}\left(z', G^u_s\right) \leq$$
$$\sum_{z' \in future_h\left(\boldsymbol{z_{late}}, G^u_t\right)} vote_{x,y}\left(z', G^u_s\right) +$$
$$\left|anticone_h\left(z_{late}, G^u_{[time(z_{late}),time(z_{late})+d]}\right)\right| \leq$$
$$\sum_{z' \in future_h\left(z_{late}, G^v_t\right)} vote_{x,y}\left(z', G^u_s\right) +$$
$$\left|anticone_h\left(z_{late}, G^u_{[time(z_{late}),time(z_{late})+d]}\right)\right| +$$
$$\left|G^{oracle}_{[t-d,t]} \cap honest\right| \leq$$
$$\sum_{z' \in future_h\left(\boldsymbol{x}, G^v_t\right)} vote_{x,y}\left(z', G^u_s\right) +$$
$$\left|anticone_h\left(z_{late}, G^u_{[time(z_{late}),time(z_{late})+d]}\right)\right| +$$
$$\left|G^{oracle}_{[t-d,t]} \cap honest\right| +$$
$$\left|future_h\left(z_{late}, G^v_t\right) \setminus future_h\left(x, G^v_t\right)\right|.$$

b) Attacker blocks: We utilize our worst case assumptions described in *Part I* to obtain:

$$\sum_{z' \in future_a\left(\boldsymbol{z}, G^u_{[time(z_{late}),t]}\right)} vote_{x,y}\left(z', G^u_s\right) =$$
$$\sum_{z' \in future_a\left(\boldsymbol{z_{late}}, G^u_{[time(z_{late}),t]}\right)} vote_{x,y}\left(z', G^u_s\right) \leq$$
$$\sum_{z' \in future_a(\boldsymbol{x}, G^v_t)} vote_{x,y}\left(z', G^u_s\right) +$$
$$\left|future_a\left(z_{late}, G^u_t\right) \setminus future_a\left(x, G^v_t\right)\right|.$$

c) All blocks: We combine the honest and attacker blocks in $future\left(z, G^u_{[time(z_{late}),t]}\right)$ to obtain:

$$\sum_{z' \in future\left(z, G^u_{[time(z_{late}),t]}\right)} vote_{x,y}\left(z', G^u_s\right) \leq$$
$$\sum_{z' \in future_h(x, G^v_t)} vote_{x,y}\left(z', G^u_s\right) +$$
$$\left|anticone_h\left(z_{late}, G^u_{[time(z_{late}),time(z_{late})+d]}\right)\right| +$$
$$\left|G^{oracle}_{[t-d,t]} \cap honest\right| + \left|future_h\left(z_{late}, G^v_t\right) \setminus future_h\left(x, G^v_t\right)\right|$$
$$+ \sum_{z' \in future_a(\boldsymbol{x}, G^v_t)} vote_{x,y}\left(z', G^u_s\right) +$$
$$\left|future_a\left(z_{late}, G^u_t\right) \setminus future_a\left(x, G^v_t\right)\right| = \tag{12}$$

$$g + \left|anticone_h\left(z_{late}, G^u_{[time(z_{late}),time(z_{late})+d]}\right)\right| + \tag{13}$$
$$\left|G^{oracle}_{[t-d,t]} \cap honest\right| + \left|future_h\left(z_{late}, G^v_t\right) \setminus future_h\left(x, G^v_t\right)\right|$$
$$+ \left|future_a\left(z_{late}, G^u_t\right) \setminus future_a\left(x, G^v_t\right)\right|.$$

3) Members of $future\left(z, G^u_{[t,s]}\right)$: Finally, by the choice of s, all honest blocks created between t and s vote in favour of $x \prec y$, hence

$$\sum_{z' \in future\left(z, G^u_{[t,s]}\right)} vote_{x,y}\left(z', G^u_s\right) \leq$$
$$-\left|future_h\left(z, G^u_{[t,s]}\right)\right| + \left|future_a\left(z, G^u_{[t,s]}\right)\right| \leq$$
$$-\left|G^u_{[t,s]} \cap honest\right| + \left|G^u_{[t,s]} \cap attacker\right|,$$

where we used again the fact that $t \geq publication(x) + d \geq publication(z) + d$.

4) Combining all the above results we obtain:

$$\sum_{z' \in future(z, G_s^u)} vote_{x,y}\left(z', G_s^u\right) \leq \tag{14}$$

$$2 \cdot \left| anticone_h \left(z_{late}, G^u_{time(z_{late})} \right) \right|$$

$$- \left| future_h \left(z, G^u_{time(z_{late})} \right) \right| + \left| future_a \left(z, G^u_{time(z_{late})} \right) \right| \tag{15}$$

$$+ g + \left| anticone_h \left(z_{late}, G^u_{[time(z_{late}), time(z_{late})+d]} \right) \right| +$$

$$\left| G^{oracle}_{[t-d,t]} \cap honest \right| +$$

$$\left| future_h \left(z_{late}, G_t^v \right) \setminus future_h \left(x, G_t^v \right) \right| +$$

$$\left| future_a \left(z_{late}, G_t^u \right) \setminus future_a \left(x, G_t^v \right) \right| + \tag{16}$$

$$- \left| G^u_{[t,s]} \cap honest \right| + \left| G^u_{[t,s]} \cap attacker \right| \leq \tag{17}$$

$$\tag{18}$$

$$2 \cdot h + l + k_1 - g + \left| G^u_{[t,s]} \cap attacker \right| - \left| G^u_{[t,s]} \cap honest \right|$$

$$+ \left| future_h \left(z_{late}, G_t^v \right) \setminus future_h \left(x, G_t^v \right) \right| +$$

$$\left| future_a \left(z_{late}, G_t^u \right) \setminus future_a \left(x, G_t^v \right) \right| \leq$$

$$2 \cdot h + l + k_1 - g + \left| G^u_{[t,s]} \cap attacker \right| - \left| G^u_{[t,s]} \cap honest \right|$$

$$+ \left| future_h \left(z_{late}, G_t^{oracle} \right) \setminus future_h \left(x, G_t^v \right) \right| + \tag{19}$$

$$\left| future_a \left(z_{late}, G_t^{oracle} \right) \setminus future_a \left(x, G_t^v \right) \right| = \tag{20}$$

$$2 \cdot h + l + k_1 - g + j + m + \left| G^u_{[t,s]} \cap attacker \right| \tag{21}$$

$$- \left| G^u_{[t,s]} \cap honest \right| .$$

As z is a weak voter, we conclude that $vote_{x,y}\left(z, G_s^u\right) \leq \widetilde{sgn}\left(\left| G^u_{[t,s]} \cap attacker \right| - \left| G^u_{[t,s]} \cap honest \right| + 2 \cdot h + l + k_1 + g + j + m \right)$.

Part IV: In particular, for $z = genesis$, the event $vote_{x,y}\left(genesis, G_s^u\right) \geq 0$ is contained in the event where $\left| G^u_{[t,s]} \cap attacker \right| \geq \left| G^u_{[t,s]} \cap honest \right| - 2 \cdot h - l - k_1 - g - j - m$. By (11), this event contains also $\widehat{\mathcal{E}}_u^s(x, y)^{\complement}$, for all $u \in honest$ and $s \geq t$, hence it contains also their union $\widehat{\mathcal{E}}^{all}_{t \to \infty}(x, y)^{\complement}$. $\square$

Lemma 13. *Given the parameters of Lemma 12,*

$$\Pr\left(\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)^{\complement}\right) \leq \sum_{h'=0}^{\infty} \mathcal{P}_{oiss}(d\cdot\alpha\cdot\lambda, h')\cdot \left(\frac{\alpha}{1-\alpha}\right)^{(g-2\cdot h-j-k_1-l-m-h')^+}. \tag{22}$$

Proof. Since all nodes $u \in honest$ receive honest blocks with a delay of d seconds at most, we have that $\left|G^u_{[t,s]} \cap attacker\right| - \left|G^u_{[t,s]} \cap honest\right| \leq \left|future_a\left(z_{late}, G^{oracle}_{[t,s]}\right)\right| - \left|future_h\left(z_{late}, G^{oracle}_{[t,\max\{s-d,t\}]}\right)\right|$. We further upper bound $\left|future_a\left(x, G^{oracle}_{[s',s]}\right)\right|$ by $\left|future_a\left(x, G^{oracle}_{[s-d,s]}\right)\right|$, and observe that the latter follows a Poisson distribution with parameter $\alpha\cdot d\cdot\lambda$; we denote this variable by h'. For any given value of h', the variable $\left|future_a\left(x, G^{oracle}_{s'} \setminus G^{oracle}_{t}\right)\right| - \left|future_h\left(x, G^{oracle}_{s'} \setminus G^{oracle}_{t}\right)\right| + h'$ can be modeled as a random walk X_i (where the ith step is the creation-time of the ith block after time t), with $X_0 = h'$, and with a drift of α towards positive infinity. The probability that X_i would ever reach the interval $[-h-j-k_1-h'-l-m+g, +\infty)$ is $\left(\frac{\alpha}{1-\alpha}\right)^{g-2\cdot h-j-k_1-l-m-h'}$, if $g > h+j+k_1+l+m+h'$, and 1 otherwise (see [25], [22]). □

Corollary 14. *If*

- $j \geq \left|anticone_h\left(x, G^{oracle}_t\right)\right|$
- $l \geq \max_{z\in G^{oracle}_t \cap honest}\left\{A^z_{time(x)} - H^z_{time(x)}\right\}$
- $n_x \geq future_h\left(x, G^{oracle}_t\right)$
- $g \leq \sum_{z\in \overline{future}(x,G^v_t)} vote_{y,x}\left(z, G^v_t\right)$.

Then

$$\Pr\left(\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)\right) \leq \sum_{k=0}^{\infty}\mathcal{P}_{oiss}((2-\alpha)\cdot d\cdot\lambda, k)\cdot\sum_{h=0}^{\infty}\mathcal{P}_{oiss}(d\cdot(1-\alpha)\cdot\lambda, h)\cdot \sum_{m=0}^{\infty}\binom{n_x+j+h+m-1}{m}\cdot(1-\alpha)^{n_x+j+h}\cdot\alpha^m\cdot \left(\frac{\alpha}{1-\alpha}\right)^{(g-2\cdot h-k-j-l-m)^+} \tag{23}$$

Proof. We build on the results of previous lemmas. The proof of Lemma 12, which is deterministic, remains intact when the corresponding parameters serve as bounds; see (14)-(21).

The variables k_1, $\left|anticone_h\left(z_{late}, G^u_{[time(z_{late}),time(z_{late})+d]}\right)\right|$, and h' are the sum of independent Poisson processes; the parameter of the first two is $d \cdot (1-\alpha) \cdot \lambda$, and the parameter of h' is $d \cdot \alpha \cdot \lambda$. Thus, their sum is a new Poisson variable k with parameter $(2 \cdot (1-\alpha) + \alpha) \cdot d \cdot \lambda = (2-\alpha) \cdot d \cdot \lambda$. The variable $\left|anticone_h\left(z_{late}, G^u_{time(z_{late})}\right)\right|$ is an additional Poisson variable with parameter $d \cdot (1-\alpha) \cdot \lambda$. We denote it by h (thereby overriding its original meaning in Lemma 12).

Lemma 12 uses the variable $m = \left|future_a\left(z_{late}, G^{oracle}_t\right) \setminus future_a\left(x, G^v_t\right)\right|$ which is upper bounded by $\left|future_a\left(z_{late}, G^{oracle}_t\right)\right|$. Provided that the honest network has created precisely n blocks since the creation of z_{late}, the number of blocks created by the attacker at the same time follows a negative binomial distribution (see [22]), i.e., it takes the value m with probability $\binom{n+m-1}{m} \cdot (1-\alpha)^n \cdot \alpha^m$. In the worst case, all of these blocks belong to $future\left(z_{late}, G^{oracle}_t\right)$. Here, again, it is sufficient to upper bound n, since increasing the parameter n results in a distribution over m that stochastically dominates (in first order) the original one. The number of honest blocks created after $time(z_{late})$ (up to time t) is upper bounded by $antipast_h\left(z_{late}, G^{oracle}_t\right)$, since blocks in $past_h\left(z_{late}\right)$. We thus have: $n \leq \left|anticone_h\left(z_{late}, G^v_t\right)\right| + \left|future_h\left(z_{late}, G^v_t\right) \setminus future_h\left(x, G^v_t\right)\right| + \left|future_h\left(x, G^v_t\right)\right| \leq h + j + n_x$.

Finally, as l and j are upper bounds and g is a lower bound to the corresponding variables from Lemma 12, one could simply turn all equalities in its proof (and in the proof of Lemma 13) into "$\leq$" inequalities and the proof remains intact. □

Below we revisit previous results, regarding the case where x is known to be an honest block, and to the case where one needs to defend a group of blocks rather than an individual block.

Lemma 19. *Assume that $node(x) \in \underline{honest}$ and that $publication(y) \geq publication(x) + \underline{d}$. Let z_{late} be the latest block in $\overline{anticone_h}\left(x, G^v_t\right)$ and let z_{early} be the earliest block in $\overline{anticone_h}\left(x, G^v_t\right)$. Furthermore, assume:*

- $l := \max_{z \in G^{oracle}_t \cap honest}\left\{\left|future_a\left(z, G^u_{time(z_{early})}\right)\right| - \left|future_h\left(z, G^u_{time(z_{early})}\right)\right|\right\}$
- $n_x \geq \max_{x' \in \overline{anticone_h}(x)}\left\{\left|future_h\left(x', G^{oracle}_t\right)\right|\right\}$
- $g \leq \max_{x' \in \overline{anticone_h}(x, G^v_t)}\left\{z \in \overline{future}\left(x', G^v_t\right) : vote_{y,x}\left(z, G^v_t\right) = -1\right\} - \min_{x' \in \overline{anticone_h}(x, G^v_t)}\left\{z \in \overline{future}\left(x', G^v_t\right) : vote_{y,x}\left(z, G^v_t\right) = +1\right\}$.

Then,

$$\Pr\left(\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)^{\complement}\right) \leq \sum_{h=0}^{\infty} \mathcal{P}_{oiss}(d \cdot \lambda, h) \cdot \tag{24}$$

$$\sum_{m=0}^{\infty} \binom{n_x + m - 1}{m} \cdot (1-\alpha)^{n_x} \cdot \alpha^m \cdot \tag{25}$$

$$\left(\frac{\alpha}{1-\alpha}\right)^{(g-h-l-m)^+}.$$

Proof. Let $k_1 \geq \left|G^{oracle}_{[t-d,t]} \cap honest\right|$, and let $m := \left|future_a\left(z_{early}, G^{oracle}_t\right) \setminus future_a\left(z_{late}, G^v_t\right)\right|$. We adjust the analysis from the proof of Lemma 12. We claim that for all $z \in \overline{anticone_h}\left(x, G^v_t\right)$:

$$\begin{aligned}&vote_{x,y}\left(z, G^u_s\right) \leq \widetilde{sgn}\left(\left|G^u_{[t,s]} \cap attacker\right| - \left|G^u_{[t,s]} \cap honest\right|\right.\\&\left.l + k_1 + m - g\right).\end{aligned}$$

We prove the claim by a complete induction on $D(z) := \left|future\left(z, \overline{anticone_h}\left(x\right)\right)\right|$. Assume we have proved the claim for any z with $D(z) < D$. We now prove it for z with $D(z) = D$. If $z = x$ then $vote_{x,y}\left(z, G^u_s\right) = -1$ hence the above inequality is satisfied trivially. Otherwise, z is a weak voter, as $y \notin \overline{past}\left(z\right)$ by the assumption on $publication(y)$, therefore $vote_{x,y}\left(z, G^u_s\right)$ is given by the sign of the sum of votes in its future. We decompose these voters into three subsets: members of $future\left(z, G^u_{time(z_{late})}\right)$, members of $future\left(z, G^u_{[time(z_{late}),t]}\right)$, and members of $future\left(z, G^u_{[t,s]}\right)$.

1) Members of $future\left(z, G^u_{time(z_{late})}\right)$: By the induction hypothesis we know that all blocks in $future_h\left(z, \overline{anticone_h}\left(z_{late}\right)\right)$ vote in favour of $x \prec y$, hence We obtain:

$$\begin{aligned}&\sum_{z' \in future\left(z, G^u_{time(z_{late})}\right)} vote_{x,y}\left(z', G^u_s\right) \leq\\&-\left|future_h\left(z, G^u_{time(z_{late})}\right)\right| + \left|future_a\left(z, G^u_{time(z_{late})}\right)\right|.\end{aligned}$$

2) Members of $future\left(z, G^u_{[time(z_{late}),t]}\right)$: Every z' in this set belongs to $future\left(x'\right)$ for some $x' \in \overline{anticone_h}\left(x, G^v_t\right)$, therefore, by the definition of g:

$$\begin{aligned}&\sum_{z' \in future\left(z, G^u_{[time(z_{late}),t]}\right)} vote_{x,y}\left(z', G^u_s\right) \leq\\&-g + \left|future_a\left(z_{early}, G^{oracle}_t\right) \setminus future_a\left(z_{late}, G^v_t\right)\right| +\\&\left|future_h\left(z, G^u_t\right) \setminus future_h\left(z, G^v_t\right)\right| \leq\\&-g + \left|future_a\left(z_{early}, G^{oracle}_t\right) \setminus future_a\left(z_{late}, G^v_t\right)\right| + \left|G^{oracle}_{[t-d,t]} \cap honest\right| =\\&-g + m + k_1.\end{aligned} \tag{26}$$

3) Members of $future\left(z, G^u_{[t,s]}\right)$: By the choice of s, all honest blocks created between t and

s vote in favour of $x \prec y$, hence

$$\sum_{z' \in future\left(z, G^u_{[t,s]}\right)} vote_{x,y}\left(z', G^u_s\right) \leq$$
$$-\left|future_h\left(z, G^u_{[t,s]}\right)\right| + \left|future_a\left(z, G^u_{[t,s]}\right)\right| \leq$$
$$-\left|G^u_{[t,s]} \cap honest\right| + \left|G^u_{[t,s]} \cap attacker\right|,$$

where we used the fact that $t \geq publication(x) + d \geq publication(z) + d$.

4) All in all,

$$\sum_{z' \in future(z, G^u_s)} vote_{x,y}\left(z', G^u_s\right) \leq \tag{27}$$
$$l + k_1 + m - g - \left|G^u_{[t,s]} \cap honest\right| + \left|G^u_{[t,s]} \cap attacker\right| \leq \tag{28}$$
$$l + k_1 + m - g - \left|future_h\left(z_{late}, G^{oracle}_{[t,\max\{s-d,t\}]}\right)\right| +$$
$$\left|future_a\left(z_{late}, G^{oracle}_{[t,s]}\right)\right| \tag{29}$$

Therefore, the event where for some s and some u, $vote_{x,y}\left(virtual\left(G^u_s\right)\right) \geq 0$ is contained in the event where (29) is non-negative. As in the proof of Lemma 13, the probability of the latter event is upper bounded by $\left(\frac{\alpha}{1-\alpha}\right)^{(l+k_1+h'+m-g)^+}$, where h' equals $\left|future_a\left(x, G^{oracle}_{[s',s]}\right)\right|$. We then combine k_1 and h' into one Poisson variable h with parameter $\alpha \cdot d \cdot \lambda + (1-\alpha) \cdot d \cdot \lambda = d \cdot \lambda$, to obtain:

$$\Pr\left(\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)^{\mathsf{C}}\right) \leq \sum_{h=0}^{\infty} \mathcal{P}_{oiss}(d \cdot \lambda, h) \cdot \tag{30}$$
$$\sum_{m=0}^{\infty} \binom{n_x + m - 1}{m} \cdot (1-\alpha)^{n_x} \cdot \alpha^m \cdot \tag{31}$$
$$\left(\frac{\alpha}{1-\alpha}\right)^{(g-h-l-m)^+}.$$

□

Corollary 20. *If in addition to Lemma 19's assumptions we know that* $publication(y) \geq t$, *then*

$$\Pr\left(\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)^{\mathsf{C}}\right) \leq \sum_{h=0}^{\infty} \mathcal{P}_{oiss}(d \cdot \alpha \cdot \lambda, h) \cdot \tag{32}$$
$$\sum_{m=0}^{\infty} \binom{n_x + m - 1}{m} \cdot (1-\alpha)^{n_x} \cdot \alpha^m \cdot \tag{33}$$
$$\left(\frac{\alpha}{1-\alpha}\right)^{(g-h-l-m)^+}.$$

Proof. Given that y wasn't published until time t, we know that all honest blocks in $future\left(x, G^{oracle}_{[t-d,t]}\right)$ vote in favour of x, hence the reduction of $k_1 = \left|G^{oracle}_{[t-d,t]} \cap honest\right|$ in (26) is superfluous, and we thus only need to reduce h', the Poisson variable with parameter $d \cdot \alpha \cdot \lambda$ from Lemma 13. □

Corollary 21. *Let $X \subseteq G^v_t \cap honest$ and $Y \subseteq G^{oracle}_t \setminus G^v_t$. Assume further that elements in X do not relate topologically to one another (i.e., $\forall x_1, x_2 \in X, x_1 \in anticone\left(x_2, G^v_t\right)$). Let z_{late} be the latest block in X, let z_{early} be the earliest block in X.*

Then,

$$\Pr\left(\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)^{\complement}\right) \leq \sum_{h=0}^{\infty} \mathcal{P}_{oiss}((3-2\cdot\alpha)\cdot d\cdot\lambda, h)\cdot$$
$$\sum_{m=0}^{\infty} \binom{n_x+m-1}{m} \cdot (1-\alpha)^{n_x} \cdot \alpha^m \cdot \left(\frac{\alpha}{1-\alpha}\right)^{(n_x-h-l-m)^+}.$$

Proof. We adjust the result of Lemma 19. The main modification is that now g must relate to all pairs (x,y). Define: $g := \max_{x_1,x_2\in X}\left\{z \in \overline{future}\left(x_1, G^v_t\right) : vote_{y,x_2}\left(z, G^v_t\right) = -1\right\} - \min_{x_1,x_2\in X}\left\{z \in \overline{future}\left(x_1, G^v_t\right) : vote_{y,x_2}\left(z, G^v_t\right) = +1\right\}$. Observe that in the interval $[time(z_{early}) + 2\cdot d, t]$ all honest blocks belong to $\cap_{x\in X}\overline{future}\left(x, G^v_t\right)$. In particular, if we denote $h' := n_x - g$ we have that h' is upper bounded by a Poisson variable with parameter $2\cdot d\cdot\lambda$. We then apply the analysis done in the proof of Lemma 19, with s being the first time at which for some $(x,y) \in X\times Y$, $vote_{x,y}\left(virtual\left(G^u_s\right)\right) \geq 0$. Combining the result of that lemma with the probability distribution over h' we conclude that the probability of the event $\cup_{(x,y)\in X\times Y}\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)$ is at most

$$\Pr\left(\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)^{\complement}\right) \leq \sum_{h'=0}^{\infty} \mathcal{P}_{oiss}(2\cdot d\cdot(1-\alpha)\cdot\lambda, h')\cdot\sum_{h=0}^{\infty}\mathcal{P}_{oiss}(d\cdot\lambda, h)\cdot$$
$$\sum_{m=0}^{\infty} \binom{n_x+m-1}{m} \cdot (1-\alpha)^{n_x} \cdot \alpha^m \cdot \left(\frac{\alpha}{1-\alpha}\right)^{(n_x-h'-h-l-m)^+} =$$
$$\sum_{h=0}^{\infty} \mathcal{P}_{oiss}((3-2\cdot\alpha)\cdot d\cdot\lambda, h)\cdot$$
$$\sum_{m=0}^{\infty} \binom{n_x+m-1}{m} \cdot (1-\alpha)^{n_x} \cdot \alpha^m \cdot \left(\frac{\alpha}{1-\alpha}\right)^{(n_x-h-l-m)^+}.$$

□

Corollary 15. *If in addition to the assumptions of Corollary 14 we assume that* $M \leq |future_a(x, G_t^v)|$, *then*

$$\Pr\left(\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y) \mid |future_a(x,G_t^v)| \geq M\right) \leq \tag{34}$$

$$\sum_{k=0}^{\infty}\mathcal{P}_{oiss}((2-\alpha)\cdot d\cdot\lambda,k)\cdot\sum_{h=0}^{\infty}\mathcal{P}_{oiss}(d\cdot(1-\alpha)\cdot\lambda,h)\cdot \tag{35}$$

$$\left(\sum_{m'=M}^{\infty}\binom{n_x+j+h+m'-1}{m'}\cdot(1-\alpha)^{n_x+j+h}\cdot\alpha^{m'}\right)^{-1}\cdot$$

$$\sum_{m=M}^{\infty}\binom{n_x+j+h+m-1}{m}\cdot(1-\alpha)^{n_x+j+h}\cdot\alpha^{m}\cdot$$

$$\left(\frac{\alpha}{1-\alpha}\right)^{(g-2\cdot h-k-j-l-(m-M))^+}$$

We denote the RHS of this inequality by $f_{post\_mine}(n_x, g, j, l, M)$. We note that from Lemmas 23, 30, and 28 it follows that, in order to compute f_{post_mine}, one can truncate these sums and suffer an exponentially low error.

Proof. Under the assumption on M we have $\left|future_a\left(z_{late}, G_t^{oracle}\right) \setminus future_a(x, G_t^v)\right| = \left|future_a\left(z_{late}, G_t^{oracle}\right)\right| - |future_a(x, G_t^v)| \geq \left|future_a\left(z_{late}, G_t^{oracle}\right)\right| - M$.

We then adjust the result of Corollary 14 and adjust (4) to account for the above updated definition of m. Thus, in the exponent, we substitute $m - M$ for m and write: $\left(\frac{\alpha}{1-\alpha}\right)^{(g-2\cdot h-j-k-l-(m-M))^+}$. Next, the updated probability distribution over $m - M$ can be obtained by conditioning the negative binomial distribution (described in the proof of Corollary 14) on its being larger than or equal to M; indeed, the M blocks of $future_a(x, G_t^v)$ were created after z_{late} (and before time t), and $future_a(x, G_t^v) \subseteq future_a\left(z_{late}, G_t^{oracle}\right)$. Consequently, the probability distribution over $m - M$ is given by

$$\Pr(m-M) = \left(\sum_{m'=M}^{\infty}\binom{n_x+j+h+m'-1}{m'}\cdot(1-\alpha)^{n_x+j+h}\cdot\alpha^{m'}\right)^{-1}\cdot$$

$$\binom{n_x+j+h+m-1}{m}\cdot(1-\alpha)^{n_x+j+h}\cdot\alpha^{m},$$

and we arrive at the desired term. The rest of the arguments in the proof of Corollary 14 remain unaffected. □

1) Numerical method to calculate f_{pre_mine}:

- Put $\delta := \alpha\cdot\lambda\cdot d$. Pick some $N \gg 1$,[13] and define a matrix $T \in \mathbb{R}_{N\times N}$ as follows. For all $1 \leq l < N-1$, $T_{l-1,l} = 1-\alpha$, $T_{l+1,l} = \alpha$, and for $l = N-1$: $T_{l-1,l} = 1-\alpha$, $T_{l,l} = \alpha$.

[13]By Lemma 23, to achieve an error of at most $\widehat{\epsilon}$ it suffices to choose N such that $\left(\frac{\alpha}{1-\alpha}\right)^{N-1} < \widehat{\epsilon}/2$ and $e^{-d\cdot\alpha\cdot\lambda}\cdot\frac{(d\cdot\alpha\cdot\lambda)^N}{N!} < \widehat{\epsilon}/2$. In particular, N is logarithmic in $\widehat{\epsilon}$.

The first column of the matrix is defined by: $T_{0,0} := (1-\alpha)\cdot e^{-\delta}$, $T_{1,0} = e^{-\delta}\cdot\alpha + e^{-\delta}\cdot\delta$, for $1 < l < N-1$: $T_{l,0} = e^{-\delta}\cdot\frac{\delta^l}{l!}$, and for $l = N-1$: $T_{l,0} = 1 - \sum_{l=0}^{N-2} e^{-\delta}\cdot\frac{\delta^l}{l!}$.

- Find the eigenvector of T corresponding to the eigenvalue 1, and denote it π. Define $\Pi(l) := \sum_{l'=0}^{l}\pi(l')$, and, finally, define $f_{pre\_mine}(l) := 1 - \Pi((l-1)^+)$.

The matrix T is the transition probability matrix of a special reflecting random walk (X_k) over the nonnegative integers: $T_{i,j} := \Pr(X_{k+1} = i \mid X_k = j)$. At every position (apart from the edges 0 and $N-1$) the walk takes a step towards negative infinity w.p.$(1-\alpha)$ and towards positive infinity w.p.α. Whenever it reaches the origin, it jumps to its next position in $\{0,1,...,N-1\}$ according to a (modified) Poisson distribution. It is easy to see that this random walk induces an ergodic Markov chain, hence it has a unique stationary distribution, which we denoted π; Π is the cumulative probability function of π.

Lemma 22. *For all $\hat{r} \geq r$ and for all For all $l \in \mathbb{N}$:*

$$\Pr\left(\max_{z\in G_{\hat{r}}^{oracle}\cap honest}\left\{\left|future_a\left(z, G_r^{oracle}\right)\right| - \left|future_h\left(z, G_r^{oracle}\right)\right|\right\} > l\right) \leq f_{pre\_mine}(l). \tag{36}$$

Proof. ***Part I:*** We prove the result assuming the maximum is taken over all $z \in G_{\boldsymbol{r}}^{oracle} \cap honest$; taking then the maximum over all $z \in G_{\hat{\boldsymbol{r}}}^{oracle} \cap honest$ does not change the result, because the variable $\left\{\left|future_a\left(z, G_r^{oracle}\right)\right| - \left|future_h\left(z, G_r^{oracle}\right)\right|\right\}$ is nonnegative (as will be shown below), and its value for $z \in G_{\hat{r}}^{oracle} \setminus G_r^{oracle}i$ is zero.

We show that the variable $\max_{z\in G_s^{oracle}\cap honest}\left\{\left|future_a\left(z, G_s^{oracle}\right)\right| - \left|future_h\left(z, G_s^{oracle}\right)\right|\right\}$ can be modeled as a reflecting random walk (with some special behaviour when the walk visits the origin, due to the honest network's inner delay d).

Intuitively, observe that whenever a new honest block b is created, $future_h\left(z, G_{time(b)}^{oracle}\right)$ increases by 1 for all z's in its past. For b itself, the value of this variable is 0. Thus, the value of $\max_{z\in G_s^{oracle}\cap honest}\left\{future_a\left(z, G_s^{oracle}\right) - future_h\left(z, G_s^{oracle}\right)\right\}$ is lower bounded by 0. On the other hand, whenever a new attack block is created, the value of $future_a\left(z, G_s^{oracle}\right)$ increases by 1 for all honest blocks available to it at the time (following the worst case assumptions specified in Lemma 12, *Part I*). Therefore, the attacker's maximal advance over the honest network can be modeled as a reflecting random walk. Note that, since the creation of an honest block b increases $future_h\left(z, G_s^{oracle}\right)$ only for blocks in $past_h(b)$, which might be a proper subset of $before(time(b))$ (when $d > 0$), there are certain situations where honest blocks do not "work against" attack blocks to decrease the value of $\max\left\{\left|future_a\left(z, G_s^{oracle}\right)\right| - \left|future_h\left(z, G_s^{oracle}\right)\right|\right\}$. We take this into account by skewing the behviour of the walk whenever the origin is visited (and proving that in all other states the honest network's inner delay has no effect).

In the following analysis, we assume the worst case scenario, namely, that if z_1 and z_2 are honest blocks such that $|time(z_1) - time(z_2)| < d$ then $z_1 \in anticone(z_2)$. That this is a worst case follows simply from the fact that omitting some edges between honest blocks can only decrease $future_h\left(z, G_s^{oracle}\right)$ hence increase the value of $\left|future_a\left(z, G_s^{oracle}\right)\right| - \left|future_h\left(z, G_s^{oracle}\right)\right|$.

If the attacker is creating blocks in secret, it needs to decide upon a strategy regarding which blocks should its new block point at, for every new block it creates. Consider the following strategy: The attacker's new block b, created at $time(b)$, points at $G_{time(b)}^{oracle}$ (except itself, of course). While we have already argued why this is a worst case assumption (Lemma 12, *Part I*), it is here easy to see that this strategy maximizes $\max_{z \in G_s^{oracle} \cap honest}\left\{\left|future_a\left(z, G_s^{oracle}\right)\right| - \left|future_h\left(z, G_s^{oracle}\right)\right|\right\}$.

Part II: Denote by t_i the creation time of the ith block in G_r^{oracle}. Denote by z_s the variable $\arg\max_{z \in G_s^{oracle} \cap honest}\left\{\left|future_a\left(z, G_s^{oracle}\right)\right| - \left|future_h\left(z, G_s^{oracle}\right)\right|\right\}$. Define further $A_s^z := \left|future_a\left(z, G_s^{oracle}\right)\right|$, and $H_s^z := \left|future_h\left(z, G_s^{oracle}\right)\right|$. Abbreviate $A_s := \left|future_a\left(z_s, G_s^{oracle}\right)\right|$, and $H_s := \left|future_h\left(z_s, G_s^{oracle}\right)\right|$.

We define a subseries $(s_k) \subseteq (t_i)$ recursively: $s_0 = 0$, and for all $k > 0$: $s_{k+1} = \min_i \{t_i : t_i \geq time\left(z_{s_k}\right) + d\}$. We claim that $(A_{s_k} - H_{s_k})$ has the same probability distribution as X_k. Assume this claim holds true, and let s_k be the earliest s_k with $s_k \geq r$. Then $(A_r - H_r) \leq 1 + (A_{s_k} - H_{s_k})$.[14] Consequently,

$$\begin{aligned}
&\Pr\left(\max_{z \in G_r^u \cap honest}\{|future_a\left(z, G_r^u\right)| - \right.\\
&\left.|future_h\left(z, G_r^u\right)|\} > l\right) = \\
&\Pr\left(A_r - H_r > l\right) \leq \Pr\left(A_{s_k} - H_{s_k} > l - 1\right) = \\
&\Pr\left(X_k > l - 1\right) = 1 - \Pi((l-1)^+).
\end{aligned}$$

Part III: To complete the proof we prove our claim, by induction on k. For $k = 0$, $s_0 = 0$. At time 0, following the creation of the $genesis$ block, the value of $(A_0 - H_0)$ is 0, as $future\left(genesis\right) \cap G_0^{oracle} = \emptyset$, and likewise $X_0 = 0$. Assume we have proved this for k, and we now prove it for $k+1$. Assume first that $(A_{s_k} - H_{s_k}) > 0$. Assume by way of negation that $s_k < time\left(z_{s_k}\right) + d$. Then, by the construction of (s_k), $s_k = time\left(z_{s_k}\right)$. This implies that the honest network created z_{s_k} in time s_k. Thus $(A_{s_k} - H_{s_k}) = 0$, because z_{s_k} was created at time s_k. As z_{s_k} is supposed to be in $\arg\max_{z \in G_r^{oracle} \cap honest}\{A_{s_k} - H_{s_k}\}$, this contradicts our assumption that $A_{s_k} - H_{s_k} > 0$. Thus, $\left(A_{s_k}^z - H_{s_k}^z\right) > 0$ implies $s_k \geq time\left(z_{s_k}\right) + d$.

Consequently, if $(A_{s_k} - H_{s_k}) > 0$, we are guaranteed that the entire honest network has learnt about the block z_{s_k}. Thence, the honest network adds blocks to $future\left(z_{s_k}\right)$ at a rate of $(1 - \alpha \cdot \lambda)$, while the attacker adds them at a rate of α. Every block of the honest network then contributes 1 to $\left|future_h\left(z, G_s^{oracle}\right)\right|$, whereas an attacker block contributes 1 to $\left|future_a\left(z, G_s^{oracle}\right)\right|$. Thus, $(A_{s_k} - H_{s_k})$ increases by 1 by the addition of an attacker's block, that is, w.p.α, and decreases by 1 w.p.$(1-\alpha)$. Indeed, conditioned on $X_k > 0$, X_{k+1}'s distribution behaves the same: $\Pr\left(X_{k+1} = X_k + 1 \mid X_k > 0\right) = 1 - \Pr\left(X_{k+1} = X_k - 1 \mid X_k > 0\right) = \alpha$.

[14]Indeed, if $r = s_k$ then this holds trivially. Otherwise, in the interval (r, s_k) the honest network could have contributed at most one block to $future\left(z_{s_{k-1}}\right)$, because (X_i) can decrease by at most 1 at every step, according to its transition matrix, thus in the interval $(r, s_k) \subseteq (s_{k-1}, s_k)$ the honest network created at most 1 block.

Assume now that $(A_{s_k} - H_{s_k}) = 0$. It cannot be the case that the block that was created in time s_k belongs to the attacker, since that would imply that the attacker has an advantage of at least 1 over the last block that was created by the honest network (up to time s_k). Therefore, it belongs to the honest network. By the definition of z_{s_k}, it is precisely the block that was created in time s_k. Consequently, in the interval $(s_k, s_k + d)$, the honest network does not add blocks to $future\,(z_{s_k})$ (recall we are assuming that the worst case scenario is realized, i.e., a propagation time of d seconds per honest block). During this interval, the attacker creates blocks at following a Poisson process with parameter $\alpha \cdot \lambda$. Thus, $\left(A^{z_{s_k}}_{s_k+d} - H^{z_{s_k}}_{s_k+d}\right) = i$ w.p. $\mathcal{P}_{oiss}(\alpha \cdot \lambda \cdot d, i)$. Upon which, the next block in the system, created after $s_k + d$, is the attacker's w.p.α, in case which the total gap increases by $j+1$, i.e., $\left(A^{z_{s_{k+1}}}_{s_{k+1}} - H^{z_{s_{k+1}}}_{s_{k+1}}\right) = \left(A^{z_{s_k}}_{s_k+d} - H^{z_{s_k}}_{s_k+d}\right) + 1$; alternatively, the next block after $s_k + d$ is the honest network's, w.p.$(1-\alpha)$, in case which $\left(A^{z_{s_{k+1}}}_{s_{k+1}} - H^{z_{s_{k+1}}}_{s_{k+1}}\right) = \max\left\{\left(A^{z_{s_k}}_{s_k+d} - H^{z_{s_k}}_{s_k+d}\right) - 1, 0\right\}$. By comparing this to $\Pr\left(X_{k+1} \mid X_k = 0\right)$, we see that also in this case the variable X_{k+1} behaves the same as $\left(A^{z_{s_{k+1}}}_{s_{k+1}} - H^{z_{s_{k+1}}}_{s_{k+1}}\right)$. $\square$

Lemma 23. *$f_{pre\_mine}(l) \leq C_l \cdot e^{-B_l \cdot future_a(x, G^v_t)}$ for some positive constants B_l, C_l.*

To get the intuition of this result, notice that when $d = 0$ the stationary distribution of the reflecting random walk is known to be proportionate to $\left(\frac{\alpha}{1-\alpha}\right)^l$, and if $d > 0$, this relation still holds for $l \gg d \cdot \lambda$.

Proof. For $n > 1$, the stationary distribution π satisfies the relation $\pi(n) = (1-\alpha) \cdot \pi(n+1) + \alpha \cdot \pi(n-1) + e^{-\delta} \cdot \frac{\delta^n}{n!} \cdot \pi(0)$. Let us write $\pi(n) = C_n \cdot \left(\frac{\alpha}{1-\alpha}\right)^n$ for $n \geq 0$. We have:

$$C_n \cdot \left(\frac{\alpha}{1-\alpha}\right)^n = (1-\alpha) \cdot C_{n+1} \cdot \left(\frac{\alpha}{1-\alpha}\right)^{n+1} + \tag{37}$$

$$\alpha \cdot C_{n-1} \cdot \left(\frac{\alpha}{1-\alpha}\right)^{n-1} + e^{-\delta} \cdot \frac{\delta^n}{n!} \cdot \pi(0) \Longrightarrow \tag{38}$$

$$C_n = C_{n+1} \cdot \alpha + C_{n-1} \cdot (1-\alpha)^{-1} + e^{-\delta} \cdot \frac{\left(\delta \cdot \frac{(1-\alpha)}{\alpha}\right)^n}{n!} \cdot \pi(0). \tag{39}$$

For large enough n's, the last summand in the above expression is negligible. Thus, when we write $\forall n : C_n \approx C$, the above relation will be satisfied for large n's (up to the negligible error of the last summand). Thus, for some constant C, $\pi(n) \leq C \cdot \left(\frac{\alpha}{1-\alpha}\right)^n$, hence $1 - \Pi((n-1)^+) = \sum_{k=n}^{\infty} \pi(k) \leq B_l \cdot e^{-C_l \cdot n}$, for some large enough n, and some constants $B_l, C_l > 0$. $\square$

The following Corollary is immediate from Lemma 22.

Corollary 24. *In Lemma 19, if l is not known, then*

$$\Pr\left(\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)^{\complement}\right) \leq \sum_{l=0}^{\infty}\pi(l)\cdot\sum_{h=0}^{\infty}\mathcal{P}_{oiss}(d\cdot\lambda,h)\cdot \tag{40}$$

$$\sum_{m=0}^{\infty}\binom{n_x+m-1}{m}\cdot(1-\alpha)^{n_x}\cdot\alpha^{m}\cdot \tag{41}$$

$$\left(\frac{\alpha}{1-\alpha}\right)^{(g-h-l-m)^{+}}.$$

Similarly, in Corollary 20,

$$\Pr\left(\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)^{\complement}\right) \leq \sum_{l=0}^{\infty}\pi(l)\cdot\sum_{h=0}^{\infty}\mathcal{P}_{oiss}(d\cdot\alpha\lambda,h)\cdot \tag{42}$$

$$\sum_{m=0}^{\infty}\binom{n_x+m-1}{m}\cdot(1-\alpha)^{n_x}\cdot\alpha^{m}\cdot \tag{43}$$

$$\left(\frac{\alpha}{1-\alpha}\right)^{(g-h-l-m)^{+}}. \tag{44}$$

Finally, in Corollary 21,

$$\Pr\left(\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)^{\complement}\right) \leq \sum_{l=0}^{\infty}\pi(l)\cdot\sum_{h=0}^{\infty}\mathcal{P}_{oiss}((3-2\cdot\alpha)\cdot d\cdot\lambda,h)\cdot$$
$$\sum_{m=0}^{\infty}\binom{n_x+m-1}{m}\cdot(1-\alpha)^{n_x}\cdot\alpha^{m}\cdot\left(\frac{\alpha}{1-\alpha}\right)^{(n_x-h-l-m)^{+}}.$$

Using this corollary we can prove the bound that is used by the online policy described in Algorithm 7. Denote:

$$risk\_hidden(T,g) := \sum_{l=0}^{\infty}\pi(l)\cdot\sum_{m=0}^{\infty}\mathcal{P}_{oiss}((T+2\cdot d)\cdot\alpha\cdot\lambda)\cdot \tag{45}$$

$$\left(\frac{\alpha}{1-\alpha}\right)^{(g-l-m)^{+}} \tag{46}$$

Corollary 25. *If Algorithm 7 returns a value less than ϵ then* $\Pr\left(\cup_{y\in G^{pub}_{\infty}\setminus G^{pub}_{t}}\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)^{\complement}\right) < \epsilon$.

Proof. First, observe that the variable g used in Corollary 20 could be replaced (here and in that corollary) with $\min_{x'\in\overline{anticone_h}(x,G^v_t)}|future(x',G^v_t)|$, because all blocks in $future(x,G^v_t)$ vote in favour of x, by the assumption on y. The value assigned to g, in line 5 of Algorithm 7, is upper bounded by $\min_{x'\in\overline{anticone}(x,G_x)}|future(x',G_x)|$, because G_x includes all honest blocks in G^v_t. Next, T is assigned the value $time\_now - received^v(x) = t - received^v(x)$, in line 3. Observe that m is distributed according to $\mathcal{P}_{oiss}(m,(t-time(x))\cdot\alpha\cdot\lambda)$.[15] As $time(x)\geq received^v(x)+d$, we can upper bound this by a Poisson variable with parameter $(T+d)\cdot\alpha\cdot\lambda$. We then adjust the result of the second term in Corollary 24; we combine the distrbituions over h and m (where h is taken from (42), to conclude that

$$\Pr\left(\cup_{y\in G^{pub}_\infty\setminus G^{pub}_t}\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)^{\complement}\right)\leq \tag{47}$$

$$\sum_{l=0}^{\infty}\pi(l)\cdot\sum_{m=0}^{\infty}\mathcal{P}_{oiss}((T+2\cdot d)\cdot\alpha\cdot\lambda)\cdot\left(\frac{\alpha}{1-\alpha}\right)^{(g-l-m)^+}= \tag{48}$$

$$risk\_hidden(T,g). \tag{49}$$

Note that we do not need to apply here a union bound over the different y's in $G^{pub}_\infty\setminus G^{pub}_t$, because our analysis assumes that in the worst case all of the attacker blocks vote strongly in favour of $y\prec x$, for all y in this set, and, additionally, all honest blocks in G^v_t will always vote strongly in favour of $x\prec y$, for all y's in this set (as they do not see y in their past). Thus, under our worst case analysis, the event where for some y in $G^{pub}_\infty\setminus G^{pub}_t$ the attacker manages to reverse the relation $x\prec y$ is equivalent to the event where it manages to do so for a given y. In conclusion, if Algorithm 7 returned a value less than ϵ, we know that $\Pr\left(\cup_{y\in G^{pub}_\infty\setminus G^{pub}_t}\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)^{\complement}\right)<\epsilon$. □

Denote by $dist\_gap(b,G)$ the minimal k for which $gap\left(b,\langle G,b,K\rangle\right)=0$.

Lemma 26. *Let b be an honest block. Then,*

$$\Pr\left(\cup_{u\in honest,s\in[time(b),\infty)}dist\_gap\left(b,G^u_s\right)>K\right)\leq \tag{50}$$

$$\sum_{l=0}^{\infty}\pi(l)\cdot\sum_{h=0}^{\infty}\mathcal{P}_{oiss}(d\cdot\lambda,h)\cdot$$

$$\sum_{m=0}^{\infty}\binom{n_x+m-1}{m}\cdot(1-\alpha)^{n_x}\cdot\alpha^m\cdot \tag{51}$$

$$\left(\frac{\alpha}{1-\alpha}\right)^{(K-h-l-m)^+}.$$

We denote the RHS of (50) by $f_{distgap}(K)$.

Proof. By its definition, the event where $dist\_gap\left(b,G^u_s\right)>K$ is equivalent to the event where some block in $anticone\left(b,G^{node(b)}_{time(b)}\right)\cup\left(G^u_s\setminus G^{node(b)}_{time(b)}\right)$ precedes b (or achieves a tie with it) according to $vote\left(virtual\left(\langle G^u_s,b,K\rangle\right)\right)$. In $\langle G^u_s,b,K\rangle$, b has additional K blocks $b_1,...,b_K$ that vote in favour of it against any other $y\notin\overline{past}\left(b\right)$; indeed, for any $y\notin\overline{past}\left(b\right)$, $y\notin\overline{past}\left(b_i\right)$. Consequently, at $G^{node(b)}_{time(b)}$, K blocks in $future\left(b\right)$ vote in its favour against any block in its anticone. We can thus apply the first part of Corollary 24 with $n_x=K$, $X=\{b\}$, and $Y=$

[15]Our previous analysis measured m using n_x, as it was structure-based and had no access to T.

$G_{\infty}^{oracle} \setminus \overline{past}\,(b)$ to conclude that (50) is an upper bound on the probability that a block in $anticone\left(b, G_{time(b)}^{node(b)}\right)$ will ever precede b (or obtain a tie with it) in the pairwise order of $\langle G_s^u, b, k\rangle$ for any $s \geq time(b)$ in the future. $\square$

Lemma 27. *For all $n_x \in \mathbb{N}$,*

$$\Pr\left(|future_h\left(x, G_t^v\right)| > n_x\right) \leq \tag{52}$$

$$|future\left(x, G_t^v\right)| \cdot f_{distgap}\left(\sqrt{|future\left(x, G_t^v\right)|}\right). \tag{53}$$

The RHS of the last inequality is denoted $f_{post\_pub}\left(|future\left(x, G_t^v\right)|\right)$.

Proof. If $y = NULL$ there is nothing to prove, since then $n_x = future\left(x, G_t^v\right) \geq future_h\left(x, G_t^v\right)$. Assume $y \neq NULL$.

Denote $K := \sqrt{|future\left(x, G_t^v\right)|}$. n_x is obtained in Algorithm 3 by subtracting M, the number of blocks with $dist\_gap > K$, from $future\left(x, G_t^v\right)$. Let b be an honest block in $future\left(x, G\right)$. By Lemma 26, the probability that $dist\_gap\left(b, G_t^v\right)$ will be larger than K is at most $f_{distgap}\left(K\right)$. By the union bound, the probability that for some b in $future_h\left(x, G_t^v\right)$, $dist\_gap\left(b, G_t^v\right) > K$, is at most $|future_h\left(x, G_t^v\right)| \cdot f_{distgap}\left(K\right) \leq |future\left(x, G_t^v\right)| \cdot f_{distgap}\left(K\right) = f_{post\_pub}\left(|future\left(x, G_t^v\right)|\right)$. $\square$

The RHS of Inequality (50) implies:

Lemma 28. $f_{post\_pub}\left(|future\left(x, G_t^v\right)|\right) \leq C_c \cdot e^{-B_c \cdot future_a\left(x, G_t^v\right)}$ *for some positive constants B_c, C_c.*

Lemma 29. *For $n_j \in \mathbb{N}$ put $j := gap\left(x, G\right) + n_j$.*

$$\Pr\left(\left\{\left|anticone_h\left(x, G_t^{oracle}\right)\right| > j\right\}\right) \leq \tag{54}$$

$$f_{pre\_mine}\left(\sqrt{n_j}\right) + \sum_{h'=0}^{\infty} \mathcal{P}_{oiss}((1-\alpha)\cdot\lambda\cdot d, h')\cdot \tag{55}$$

$$f_{post\_mine}\left(n_j, n_j - h' + 1, \sqrt{n_j}\right) \tag{56}$$

We denote the RHS of this inequality by $f_{pre\_pub}(n_j)$. To understand the intuition behind this resul recall that w.h.p. a block defeats only blocks that were published close to its publication or after it.

Proof. ***Part I:*** Let $t_x := publication(x)$. Define $L_n := \left\{z \in anticone_h\left(x, G_t^v\right) : future_h\left(z, anticone_h\left(x, G_{t_x}^v\right)\right) \geq n\right\}$. (Note the use of t_x in this definition). Denote by A_n the event $\{\exists z \in L_n : z \in X_{win}\left(x, G_t^v\right)\}$. Finally, let z_e be the earliest block in $L_{n'}^{\complement} \cap anticone_h\left(x, G_t^{oracle}\right)$ and put $n' := n_j - \left|\overline{anticone_h}\left(z_e, G_t^{oracle}\right)\right| + 1$ for $n_j :=$

$\sqrt{|future\left(x, G_t^v\right)|}$. Denote by $X_{win}\left(x, G\right)$ the set of blocks that x precedes (or obtains a tie with) in the pairwise order of G's virtual vote, and by $X_{lose}\left(x, G\right)$ the rest of blocks. Then:

$$\begin{aligned}
&\left\{\left|anticone_h\left(x, G_t^{oracle}\right)\right| > gap\left(x, G\right) + n_j\right\} = \\
&\left\{\left|X_{win}\left(x, G_t^{pub}\right) \cap anticone_h\left(x, G_t^{oracle}\right)\right| + \right. \\
&\left.\left|X_{lose}\left(x, G_t^{pub}\right) \cap anticone_h\left(x, G_t^{oracle}\right)\right| > gap\left(x, G\right) + n_j\right\} = \\
&\left\{\left|X_{win}\left(x, G_t^{pub}\right) \cap anticone_h\left(x, G_t^{oracle}\right)\right| + gap\left(x, G_t^v\right) > gap\left(x, G\right) + n_j\right\} = \\
&\left\{\left|X_{win}\left(x, G_t^{pub}\right) \cap anticone_h\left(x, G_t^{oracle}\right)\right| > n_j\right\} = \\
&\left(\left\{\left|X_{win}\left(x, G_t^{pub}\right) \cap anticone_h\left(x, G_t^{oracle}\right)\right| > n_j\right\} \cap A_{n'}\right) \cup \\
&\left(\left\{\left|X_{win}\left(x, G_t^{pub}\right) \cap anticone_h\left(x, G_t^{oracle}\right)\right| > n_j\right\} \cap A_{n'}^{\complement}\right) \subseteq \\
&A_{n'} \cup \left\{\left|L_{n'}^{\complement} \cap anticone_h\left(x, G_t^{oracle}\right)\right| > n_j\right\} = \\
&A_{n'} \cup \left\{\left|L_{n'}^{\complement} \cap anticone_h\left(x, G_t^{oracle}\right) \cap \overline{anticone_h}\left(z_e, G_t^{oracle}\right)\right| + \right. \\
&\left.\left|L_{n'}^{\complement} \cap anticone_h\left(x, G_t^{oracle}\right) \cap future_h\left(z_e, G_t^{oracle}\right)\right| > n_j\right\} = \\
&A_{n'} \cup \left\{\left|L_{n'}^{\complement} \cap anticone_h\left(x, G_t^{oracle}\right) \cap future_h\left(z_e, G_t^{oracle}\right)\right| > \right. \\
&\left. n_j - \left|L_{n'}^{\complement} \cap anticone_h\left(x, G_t^{oracle}\right) \cap \overline{anticone_h}\left(z_e, G_t^{oracle}\right)\right|\right\} \subseteq \\
&A_{n'} \cup \left\{\left|anticone_h\left(x, G_t^{oracle}\right) \cap future_h\left(z_e, G_t^{oracle}\right)\right| > \right. \\
&\left. n_j - \left|\overline{anticone_h}\left(z_e, G_t^{oracle}\right)\right|\right\}.
\end{aligned}$$

As $z_e \in L_{n'}$, and by the definition of n', it cannot be the case that $anticone_h\left(x, G_t^{oracle}\right) \cap future_h\left(z_e, G_t^{oracle}\right)$ contains more than $n_j - \left|\overline{anticone_h}\left(z_e, G_t^{oracle}\right)\right|$ blocks. Thus, the event $\left\{\left|anticone_h\left(x, G_t^{oracle}\right) \cap future_h\left(z_e, G_t^{oracle}\right)\right| > n_j - \left|\overline{anticone_h}\left(z_e, G_t^{oracle}\right)\right|\right\}$ occurs w.p. 0, and we obtain: $\Pr\left(\left\{\left|anticone_h\left(x, G_t^{oracle}\right)\right| > gap\left(x, G\right) + n_j\right\}\right) \leq \Pr\left(A_{n'}\right)$.

Observe that all blocks in $future\left(z, anticone_h\left(x, G_{t_x}^v\right)\right)$ vote strongly in favour of z against x, for any $z \in L_{n'}$, and that by definition there are at least n' such votes at time t_x. Consequently, we can apply the result of Corollary 21 with respect to the following parameters: $v = pub$, $t = t_x$, $X =$ the leaf-blocks of $L_{n'}$, $Y = \{x\}$, $g := n'$, $n_x := n_j$, and $l' = \max_{z \in G_{t_x}^{oracle} \cap honest}\left\{\left|future_a\left(z, G_{t_x}^{oracle}\right)\right| - \left|future_h\left(z, G_{t_x}^{oracle}\right)\right|\right\}$, to obtain:

$$\begin{aligned}
&\Pr\left(A_{n'}\right) = \Pr\left(\exists z \in L_{n'} : z \in X_{win}\left(x, G_t^v\right)\right) \leq \\
&\Pr\left(\exists s > t_x, \exists z \in L_{n'} : z \in X_{win}\left(x, G_s^v\right)\right) \leq \\
&f_{post\_mine}\left(n_j, n', l'\right).
\end{aligned}$$

As the value of l' is unknown to us, we use Lemma 22 to conclude that with probability $\geq 1-f_{pre\_mine}(l)$ its value is at most l. Fix $l=\sqrt{n_j}$. Similarly, the value of n' is unknown to us. However, blocks in $\overline{anticone_h}\left(z_e, G_t^{oracle}\right)$ are created in the time interval $[time(z_e), time(z_e)+d]$ (by its choice), hence $\left|\overline{anticone_h}\left(z_e, G_t^{oracle}\right)\right|$ is a Poisson variable with parameter $(1-\alpha)\cdot\lambda\cdot d$. We thus conclude that:

$$\Pr\left(\left\{\left|anticone_h\left(x, G_t^{oracle}\right)\right| > gap\,(x, G)+n_j\right\}\right) \leq$$
$$f_{pre\_mine}\left(\sqrt{n_j}\right)+\sum_{h'=0}^{\infty}\mathcal{P}_{oiss}((1-\alpha)\cdot\lambda\cdot d, h')\cdot$$
$$f_{post\_mine}\left(n_j, n_j-h'+1, \sqrt{n_j}\right)=$$
$$f_{pre\_pub}(n_j).$$

□

It is easy to verify that $f_{post\_mine}\left(n_j, n_j-h'+1, \sqrt{n_j}\right)$ decreases exponentially (we do this in fact in subsequent lemmas). Therefore:

Lemma 30. *$f_{pre\_pub}(n_j) \leq C_j \cdot e^{-B_j\cdot n_j}$ for some positive constants B_j, C_j.*

In the lemma below, $oracle^u$ is a (hypothetical) node such that $G_s^{oraclej} := G_s^u \cup \left(G_s^{oracle}\cap attacker\right)$.

Lemma 16. *Conditioned on the event $\widehat{\mathcal{E}}_{t\to\infty}^{all}(x,y)$, there exists a time $\tau\in[t,\infty)$ such that $\forall s\geq\tau$: $M(oracle^u, s) \geq \left|future_a\left(x, G_s^{oracle^u}\right)\cap G_{[t,s]}^{oracle}\setminus V_{x\prec y}(G_s^{oracle^u})\right| - m^*$, for some m^* that remains fixed after τ (and with $\mathbb{E}[m^*]$ determined by the events up to time t).*

Proof. ***Part I:*** If $y\notin G_s^{oracle^u}$ then $M(oracle^u, s)=0$ (line 6), $V_{x\prec y}(G_s^{oracle^u}) = future\left(x, G_s^{oracle^u}\right)$, and the required inequality follows trivially. Assume $y\in G_s^{oracle^u}$.

Let G be any block DAG that equals the past-set of some (possibly virtual) block. Observe that conditioned on $\widehat{\mathcal{E}}_{t\to\infty}^{all}(x,y)$, for some constant C_t determined at time t, if $\left|G_{[t,s]}\cap attacker\right| - \left|G_{[t,s]}^{oracle^u}\cap honest\right| < -C_t$ then $vote_{x,y}\left(virtual\left(G\right)\right)=-1$.[16] This follows from the proof of Lemma 12: We take the LHS of (17), replace g by $\sum_{z'\in future(x,G_t)} vote_{x,y}\left(z', G\right)$, and observe that the value of the remaining term $2\cdot\left|anticone_h\left(z_{late}, G_{time(z_{late})}^{oracle^u}\right)\right| - \left|future_h\left(z, G_{time(z_{late})}^{oracle^u}\right)\right| + \left|future_a\left(z, G_{time(z_{late})}^{oracle^u}\right)\right| + \left|anticone_h\left(z_{late}, G_{[time(z_{late}),time(z_{late})+d]}^{oracle^u}\right)\right| + \left|G_{[t-d,t]}^{oracle}\cap honest\right| + \left|future_h\left(z_{late}, G_t^v\right)\setminus future_h\left(x, G_t^v\right)\right| + \left|future_a\left(z_{late}, G_t^{oracle^u}\right)\setminus future_a\left(x, G_t^v\right)\right|$ is determined by time t, hence we can denote it C_t.

Let $z\in future\left(x, G_{t,s}^{oracle}\right)$. By the conditioning on $\widehat{\mathcal{E}}_{t\to\infty}^{all}(x,y)$, $z\in attacker$. Fix the DAG $G^z := past\left(z\right)$. The above argument holds in particular for G^z: If $\left|G_{[t,s]}^z\cap attacker\right| - \left|G_{[t,s]}^z\cap honest\right| < -C_t$ then $vote_{x,y}\left(z\right) = vote_{x,y}\left(virtual\left(past\left(z\right)\right)\right) = -1$ (since z is a

[16]We write here $G_{[t,s]}^z$ for $G^z\cap before(s)\setminus before(t)$.

strong voter we do not need to specify the context of its vote). Consequently, if $z \in G_{t,s}^{oracle} \setminus V_{x \prec y}\left(G_s^{oracle}\right)$ then $\left|G_{[t,time(z)]}^{z} \cap attacker\right| - \left|G_{[t,time(z)]}^{z} \cap honest\right| \geq -C_t$.[17]

We arrive at the following important implication: If $z \in future\left(x, G_s^{oracle^u}\right) \setminus V_{x \prec y}\left(G_s^{oracle^u}\right)$ then:

$$\left|anticone\left(z, G_s^{pub}\right)\right| \geq \left|anticone\left(z, G_{time(z)}^{pub}\right)\right| \geq \tag{57}$$

$$\left|anticone\left(z, G_{time(z)}^{pub}\right) \setminus G_t^{oracle}\right| = \tag{58}$$

$$\left|G_{time(z)}^{pub} \setminus G_t^{oracle}\right| - \left|\overline{past}\,(z) \setminus G_t^{oracle}\right| \geq \tag{59}$$

$$\left|G_{time(z)}^{pub} \cap honest \setminus G_t^{oracle}\right| - G_{[t,time(z)]}^{z} \cap honest \geq \tag{60}$$

$$\left|G_{[t,time(z)]}^{pub} \cap honest\right| - G_{[t,time(z)]}^{z} \cap attacker - C_t. \tag{61}$$

Part II: Let $z_1, z_2, ...$ the order of creation of blocks in $future_a\left(x, G_s^{oracle^u} \setminus G_t^{oracle}\right) \setminus V_{x \prec y}\left(G_s^{oracle^u}\right)$. Fix z_m, and let b_m be the earliest block in $anticone_h\left(z_m, future\left(x, G_s^{oracle^u}\right)\right)$. With probability $\mathcal{P}_{oiss}(d \cdot (1-\alpha) \cdot \lambda, h')$, $\left|anticone_h\left(b_m, G_\infty^{pub}\right)\right| = h'$. By the choice of b_m together with (57) we obtain:

$$\begin{aligned}
&\left|future_h\left(b_m, G_{time(z_m)}^{pub}\right)\right| = \\
&\left|anticone_h\left(z_m, G_{time(z_m)}^{pub}\right) \setminus anticone_h\left(b_m, G_{time(z_m)}^{pub}\right)\right| \geq \\
&\left|anticone_h\left(z_m, G_{time(z_m)}^{pub}\right)\right| - \left|anticone_h\left(b_m, G_{time(z_m)}^{pub}\right)\right| \geq \\
&\left|G_{[t,time(z_m)]}^{pub} \cap honest\right| - \left|G_{[t,time(z_m)]}^{z_m} \cap attacker\right| - C_t - h' = \\
&\left|G_{[t,time(z_m)]}^{pub} \cap honest\right| - m - C_t - h',
\end{aligned} \tag{62}$$

where we used the fact that $past\,(b_m) \cap anticone_h\,(z_m) = \emptyset$, by the choice of b_m, and that $anticone_h\left(z_m, G_{time(z_m)}^{pub}\right) = antipast_h\left(z_m, G_{time(z_m)}^{pub}\right)$.

Part III: Given m, $\left|G_{[t,time(z_m)]}^{pub} \cap honest\right|$ is distributed according to a negative binomial distribution: $\Pr\left(\left|G_{[t,time(z_m)]}^{pub} \cap honest\right| = n\right) = \binom{n+m-1}{n} \cdot (1-\alpha)^n \cdot \alpha^m$. We claim that the probability that the honest block b_m will ever be preceded by z_m in the order of $virtual\left(\langle G_s^{oracle^u}, z_m, K\rangle\right)$ is at most

$$\begin{aligned}
&\sum_{l=0}^{\infty} \pi(l) \cdot \sum_{k=0}^{\infty} \mathcal{P}_{oiss}(5 \cdot d \cdot (1-\alpha) \cdot \lambda, k) \cdot \sum_{h=0}^{\infty} \mathcal{P}_{oiss}(d \cdot (1-\alpha) \cdot \lambda, h) \cdot \\
&\sum_{n=0}^{\infty} \binom{n+m-1}{m} \cdot (1-\alpha)^n \cdot \alpha^m \cdot \left(\frac{\alpha}{1-\alpha}\right)^{(n-m-K-2\cdot h-C_t-k-l)^+}.
\end{aligned} \tag{63}$$

[17]Note that G^z contains only blocks created up to $time(z)$.

This follows from a similar analysis to that made in the proof of Lemma 12 and of Corollary 21. Indeed, at $time(z_m)$ there were at least $n-m-h'-C_t$ blocks in $future\,(b_m)\setminus\overline{future}\,(z_m)$, by the above lower bound on $\left|future_h\left(b_m, G^{pub}_{time(z_m)}\right)\right|$; and while $future_a\left(b_m, G^{pub}_{time(z_m)}\right) = \emptyset$, as $b_m \notin past\,(z_m)$, there are additional K hypothetical blocks that vote $y \prec x$, by the construction of $\langle G_s^{oracle^u}, z_m, K\rangle$. Instead of reducing h' in the exponent (as in the bound given in Corollary 21), we added $2\cdot d\cdot(1-\alpha)\cdot\lambda$ to the variable k, as the sum of Poisson variables is a Poisson variable. Finally, we use the result of Lemma 22 to ensure that $\pi(l)$ upper bounds the distribution over l,[18]

As $dist\_gap(G_s^{oracle^u}, z_m) \le K$ requires z_m to precede b_m in the order of $virtual\left(\langle G_s^{oracle^u}, z_m, K\rangle\right)$, (63) serves as an upper bound also to the probability that $dist\_gap(G_s^{oracle^u}, z_m) \le K$.

Part IV: Using Lemma 23 it is easy to verify the existence of constants a, b, and W such that $\Pr\left(k + l + 2\cdot h > W\right) \le e^{-a\cdot W+b}$.

Put $K(oracle^u, s) = \sqrt{|future\,(x, G_s^{oracle^u})|}$. The block z_m is counted into $M(oracle^u, s)$ in line 9 of Algorithm 3 whenever $dist\_gap(G_s^{oracle^u}, z_m) > K(oracle^u, s)$. From (63) we conclude that the probability that z_m does *not* increment by 1 the value of $M(oracle^u, s)$ is upper bounded by

$$\Pr\left(dist\_gap(G_s^{oracle^u}, z_m) \le K(oracle^u, s)\right) \le \tag{64}$$

$$\sum_{n=0}^{\infty}\binom{n+m-1}{m}\cdot(1-\alpha)^n\cdot\alpha^m\cdot\left(\frac{\alpha}{1-\alpha}\right)^{(n-m-K(oracle^u,s)-W-C_t)^+} <$$

$$\left(\frac{\alpha}{1-\alpha}\right)^{-W-C_t-K(oracle^u,s)}\cdot\sum_{n=0}^{\infty}\binom{n+m-1}{n}\cdot(1-\alpha)^n\cdot\alpha^m\cdot\left(\frac{\alpha}{1-\alpha}\right)^{(n-m)^+} = \tag{65}$$

$$\left(\frac{\alpha}{1-\alpha}\right)^{-W-C_t-K(oracle^u,s)}\cdot\left(\Pr_{n\sim Z(m,1-\alpha)}(n>m) + \Pr_{n\sim Z(m,\alpha)}(n\le m)\right), \tag{66}$$

where $Z(n,p)$ denotes a negative binomial random variable.

We now aim at showing that the last term is upper bounded by some $e^{-D\cdot m}$. The proof is very similar to that given in Lemma 17 below.

Part V: For large enough m's, a variable distributed according to $Z(1-\alpha, m)$ converges to a normal variable with mean $m\cdot\frac{\alpha}{1-\alpha}$ and variance $m\cdot\frac{\alpha}{(1-\alpha)^2}$.[19] The second multiplicand in (66)

[18] l here represents $\max_{z\in G^{oracle}_{time(b_m)}\cap honest}\left\{\left|future_a\left(z, G^{oracle^u}_{time(b_m)}\right)\right| - \left|future_h\left(z, G^{oracle^u}_{time(b_m)}\right)\right|\right\}$.

[19] We rely here on the assumption specified in the proof of Lemma 12 according to which, in the worst case, after time t the attacker publishes all his blocks to all nodes immediately after their creation.

thus converges, as m grows, to

$$\Pr_{z\sim\mathcal{N}(0,1)}\left(z \le \frac{m - \frac{1-\alpha}{\alpha}\cdot m}{\sqrt{\frac{1-\alpha}{\alpha^2}\cdot m}}\right) + \Pr_{z\sim\mathcal{N}(0,1)}\left(z \ge \frac{m - \frac{\alpha}{1-\alpha}\cdot m}{\sqrt{\frac{\alpha}{(1-\alpha)^2}\cdot m}}\right) = \tag{67}$$

$$\Pr_{z\sim\mathcal{N}(0,1)}\left(z \ge \frac{\frac{1-\alpha}{\alpha}\cdot m - m}{\sqrt{\frac{1-\alpha}{\alpha^2}\cdot m}}\right) + \Pr_{z\sim\mathcal{N}(0,1)}\left(z \ge \frac{m - \frac{\alpha}{1-\alpha}\cdot m}{\sqrt{\frac{\alpha}{(1-\alpha)^2}\cdot m}}\right). \tag{68}$$

The following inequality is due to Komatu (1955). Let $x \ge 0$ and let $z \sim \mathcal{N}(0,1)$. Then: $\Pr(z > x) \le \frac{1}{\sqrt{2\cdot\pi}}\cdot\frac{2\cdot e^{-x^2/2}}{x+\sqrt{2+x^2}}$. Put $x_1 := \frac{\frac{1-2\cdot\alpha}{\alpha}\cdot m}{\sqrt{\frac{1-\alpha}{\alpha^2}\cdot m}}$ and $x_2 := \frac{\frac{1-2\cdot\alpha}{1-\alpha}\cdot m}{\sqrt{\frac{\alpha}{(1-\alpha)^2}\cdot m}}$. We obtain an upper bound on (68):

$$\frac{1}{\sqrt{2\cdot\pi}}\cdot\frac{2\cdot e^{-x_1^2/2}}{x_1+\sqrt{2+x_1^2}} + \frac{1}{\sqrt{2\cdot\pi}}\cdot\frac{2\cdot e^{-x_2^2/2}}{x_2+\sqrt{2+x_2^2}} \le \tag{69}$$

$$C_1\cdot e^{-x_1^2/2} + C_2\cdot e^{-x_2^2/2} = C_1\cdot e^{-D_1\cdot m} + C_2\cdot e^{-D_2\cdot m} \le C_3\cdot e^{-D_3\cdot m} \tag{70}$$

for some positive constants C_i, D_i that depend on α (a property which applies to the constants below as well).

When this term is multiplied by $\left(\frac{\alpha}{1-\alpha}\right)^{-W-C_t-K(oracle^u,s)}$ we obtain

$$\left(\frac{\alpha}{1-\alpha}\right)^{-W-C_t-K(oracle^u,s)}\cdot C_3\cdot e^{-D_3\cdot m} \le \tag{71}$$

$$C_4\cdot e^{-D_3\cdot m + D_4\cdot K(oracle^u,s)} = C_4\cdot e^{-D_3\cdot m + D_4\cdot\sqrt{|future(x,G_s^{oracle^u})|}}. \tag{72}$$

There exists therefore an M_1 such that if $m > \left|future\left(x, G_s^{oracle^u}\right)\right| > M_1$ then the last expression is upper bounded by $C_5\cdot e^{-D_5\cdot m}$ for some C_5, D_5.

Part VI: After some ψ (with expected value M_1/λ), the condition $\left|future\left(x, G_s^{oracle^u}\right)\right| \ge M_1$ is satisfied. Put $s_m := time(z_m)$ and assume $s_m \ge \psi$.

As $\sum_{m=\sqrt{\left|future\left(x,G_{s_m}^{oracle^u}\right)\right|}+1}^{\infty} C_5\cdot e^{-D_5\cdot m} < \infty$, Fatou's lemma implies that there exists (a.s.) an $m^* > \sqrt{\left|future\left(x, G_{s_m}^{oracle^u}\right)\right|}$ such that for all $m \ge m^*$, $dist\_gap(z_m) > K(oracle^u, s_m)$. The expected waiting time for z_{m^*} is finite.[20] Define $\tau = \max\{\psi, time(z_{m^*})\}$. Then, for any

[20]We have $\Pr(m^* \ge r) \le \sum_{m=\sqrt{\left|future\left(x,G_s^{oracle^u}\right)\right|}+1}^{r-1} C_5\cdot e^{-D_5\cdot m}$. Therefore,

$$\mathbb{E}[m^*] \le \sum_{r=\sqrt{\left|future\left(x,G_{s_m}^{oracle^u}\right)\right|}+1}^{\infty}\ \sum_{m=\sqrt{\left|future\left(x,G_{s_m}^{oracle^u}\right)\right|}+1}^{r-1} C_5\cdot e^{-D_5\cdot m} =$$

$$\sum_{m=\sqrt{\left|future\left(x,G_{s_m}^{oracle^u}\right)\right|}+1}^{\infty}\ \sum_{r=m+1}^{\infty} C_5\cdot e^{-D_5\cdot m} = \sum_{m=\sqrt{\left|future\left(x,G_{s_m}^{oracle^u}\right)\right|}+1}^{\infty} C_6\cdot e^{-D_6\cdot m} \le$$

$$C_7\cdot e^{-D_7\cdot\sqrt{\left|future\left(x,G_{s_m}^{oracle^u}\right)\right|}}.$$

$s \geq \tau$: $M(oracle^u, s) \geq \left| future_a\left(x, G_s^{oracle^u}\right) \cap G_{[t,s]}^{oracle} \setminus V_{x \prec y}(G_s^{oracle^u})\right| - m^*$.[21] $\square$

Lemma 17. *There exists a* $\psi \in [t, \infty)$ *such that* $\Pr\left(\mathcal{E}_{t\to\infty}^{all}(x,y,\epsilon)^{\complement} \mid \mathcal{E}_t^v(x,y,\epsilon)\right) < \epsilon$. *Moreover,* $\mathbb{E}[\psi - t] < \epsilon$.

Proof. ***Part I:*** We show that if all honest blocks vote in favour of x then all error functions converge to zero. Indeed, the event $\mathcal{E}_t^v(x,y,\epsilon)$ implies that $f_{pre\_mine}(l(G_t^v)) + f_{pre\_pub}(n_j(G_t^v)) + f_{post\_pub}(|future(x, G_t^v)|) + f_{post\_mine}(n_x(G_t^v), g(G_t^v), l(G_t^v)) < \epsilon$. By the union bound, and by Lemmas 22, 27, and 29 respectively, the following relations hold with probability $\geq 1 - \epsilon$:

- $\max_{z \in G_t^{oracle} \cap honest}\left\{\left|future_a\left(z, G_{time(x)}^{oracle}\right)\right| - \left|future_h\left(z, G_{time(x)}^{oracle}\right)\right|\right\} \leq l(G_t^v)$
- $|future_h(x, G_t^v)| \leq n_x$
- $\left|anticone_h\left(x, G_t^{oracle}\right)\right| \leq gap(x, G) + n_j =: j$

Conditioned on these relations, by Corollary 15 the event $\widehat{\mathcal{E}}_{t\to\infty}^{all}(x,y)$ occurs w.p. $\geq 1 - f_{post\_mine}(n_x(G_t^v), g(G_t^v), l(G_t^v))$. All in all, conditioned on $\mathcal{E}_t^v(x,y,\epsilon)$, the event $\widehat{\mathcal{E}}_{t\to\infty}^{all}(x,y)$ occurs w.p. $\geq 1 - \epsilon$.

Part II: We proceed to show that, conditioned on $\widehat{\mathcal{E}}_{t\to\infty}^{all}(x,y)$ and on the above relations, the value of $Risk(G_s^u, x, y)$ goes (almost surely) to 0 as time develops, for all $u \in honest$.[22]

That $f_{pre\_mine}(l(G_s^u)) + f_{pre\_pub}(n_j(G_s^u)) + f_{post\_pub}(|future(x, G_s^u)|)$ goes to 0 as s grows follows immediately from Lemmas 23, 30, and 28. Let $\epsilon_0 > 0$. We now prove that after some τ of finite expectation, $f_{post\_mine}(n_x(G_s^u), g(G_s^u), l(G_s^u)) < \epsilon_0$.

We claim that

$$M(oracle^u, s) + g(oracle^u, s) - n_x(oracle^u, s) \geq -2 \cdot \left|G_{[time(x),t]}^{oracle}\right| - m^* \qquad (73)$$

where m^* is the variable described in Lemma 16. Assume first that $attacker \cap G_s^{oracle} \subseteq G_s^{oracle^u}$. Let us decompose $future\left(x, G_s^{oracle^u}\right)$ as follows:

- Blocks in $G_{[time(x),t]}^{oracle}$. Clearly, the number of blocks in this set does not grow with s. Their contribution is lower bounded by $-2 \cdot \left|G_{[time(x),t]}^{oracle}\right|$.
- Blocks in $V_{x \prec y}(G_s^{oracle^u}) \setminus G_t^{oracle}$: Every z in this set adds $(+1)$ to $g(oracle^u, s)$. As z cannot decrement the value of $M(oracle^u, s) - n_x(oracle^u, s)$ by more than 1, the contribution of this set is at least 0. [23]

The expected waiting time for z_{m^*} is the last term divided by $\alpha \cdot \lambda$.

[21]Note that $\mathbb{E}[m^*]$ is determined by the events up to time t: take the expected value of the expression in the previous proof, where the distribution over the values of $\left|future\left(x, G_{s_m}^{oracle^u}\right)\right|$ (and of the s_m's themselves) is conditioned on $\left|future\left(x, G_t^{oracle^u}\right)\right|$ (for the $oracle^u$ which maximizes the expected value).

[22]In fact, we need to show that $\max\{Risk(G_s^u, x, y)\}$ goes to 0. However, since our analysis below takes the worst case regarding u, namely, that messages from it and to it arrive at a delay of precisely d, these events are equivalent in the worst case, and thus we will relate to u as a fixed honest node.

[23]In fact, by the conditioning on the relation $|future_h(x, G_t^v)| \leq n_x$, we know that all honest blocks belong to this category, hence we can arrive at a tighter bound: $M(oracle^u, s) + g(oracle^u, s) - n_x(oracle^u, s) \geq -\left|G_{[time(x),t]}^{oracle} \cap attacker\right| - m^*$.

- Blocks in $G_s^{oracle^u} \setminus \left(V_{x\prec y}(G_s^{oracle^u}) \cup G_t^{oracle}\right)$: Lemma 16 guarantees that, conditioned on the event $\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)$, at least $\left|future_a\left(x, G_s^{oracle^u}\right) \cap G_{[t,s]}^{oracle}\right| - m^*$ of the blocks that are published after some τ and that do not belong to $V_{x\prec y}(G_s^{oracle^u})$ – hence that add (-1) to $g(oracle^u, s)$[24] – add $(+1)$ to the value of $M(oracle^u, s)$. In other words, at most m^* blocks from the set $future_a\left(x, G_s^{oracle^u}\right) \cap G_{[t,s]}^{oracle} \setminus V_{x\prec y}(G_s^{oracle^u})$ add (-1) to $g(oracle^u, s)$ and are not canceled out by a $(+1)$ increment to the value of $M(oracle^u, s)$. The contribution of this set is therefore lower bounded by $-m^*$.

Part III: We now claim that

$$M\left(G_s^u\right) + g\left(G_s^u\right) - n_x\left(G_s^u\right) \geq -2 \cdot \left|G_{[time(x),t]}^{oracle}\right| - m^* \tag{74}$$

Indeed, let $C(z)$ be the contribution of z to (73) and let $c(z)$ be its contribution to (74). First, $C(z) \geq -2$, hence the contribution of all $z \in G_{[time(x),t]}^{oracle}$ is at least $-2 \cdot \left|G_{[time(x),t]}^{oracle}\right|$, as previously.

Assume that $z \in G_s^{oracle^u} \setminus G_t^{oracle}$ and that it votes $x \prec y$. Then z is not counted into $M(oracle^u, s)$, hence its contribution to $M(oracle^u, s) + g(oracle^u, s) - n_x(oracle^u, s)$ is $0 + 1 - 1 = 0$, i.e., $c(z) = 0$. And for the same argument $C(z) = 0$.

Assume that $z \in G_s^{oracle^u} \setminus G_t^{oracle}$ and that it votes $y \prec x$. Then $z \in attacker$ (by the conditioning on $\widehat{\mathcal{E}}^{all}_{t\to\infty}(x,y)$). In the analysis of Lemma 16 we assumed the following worst case: that for any three blocks $v, z, w \in G_s^{oracle^u}$, such that $v, z \in attacker$ and $w \in honest$, v votes strongly for $z \prec w$.[25] Under this worst case assumption regarding the votes of attacker blocks, $dist\_gap(z, G_s^{oracle^u}) \leq dist\_gap(z, G_s^u)$, as $G_s^{oracle^u} \setminus G_s^u$ contains only attacker blocks. Thus, if z was counted in $M(oracle^u, s)$ then it is counted also in $M\left(G_s^u\right)$; in particular, $C(z) \geq c(z)$.

Consequently, using the analysis from Lemma 16,

$$\begin{aligned}
&-m^* \leq \sum_{z \in G_s^{oracle^u} \setminus G_t^{oracle}} c(z) \leq \\
&\sum_{z \in G_s^{oracle^u} \setminus (G_t^{oracle} \cup V_{x\prec y}(G_s^{oracle^u}))} c(z) = \\
&\sum_{z \in G_s^{u} \setminus (G_t^{oracle} \cup V_{x\prec y}(G_s^{oracle^u}))} c(z) + \sum_{z \in G_s^{oracle^u} \setminus (G_s^{u} \cup V_{x\prec y}(G_s^{oracle^u}))} c(z) \leq \\
&\sum_{z \in G_s^{u} \setminus (G_t^{oracle} \cup V_{x\prec y}(G_s^{oracle^u}))} c(z) \leq \sum_{z \in G_s^{u} \setminus (G_t^{oracle} \cup V_{x\prec y}(G_s^{oracle^u}))} C(z).
\end{aligned}$$

All in all, $M\left(G_s^u\right) + g\left(G_s^u\right) - n_x\left(G_s^u\right) \geq -2 \cdot \left|G_{[time(x),t]}^{oracle}\right| - m^*$.

Part IV: In the remainder of the proof we occasionally abbreviate $n_x\left(G_s^u\right)$ and write simply n_x, and similarly for the rest of the variables, for convenience. Lemmas 23 and 30 imply further

[24]They cannot add 0 since only strong voters are counted into these variables.

[25]Indeed, therein we only counted honest voters in favour of honest blocks. This could be formalized using pseudo-votes, as in Lemma 18.

that there exist constants a, b, and W such that $\Pr(k+l+2\cdot h+j>W)\leq e^{-a\cdot W+b}$ (as in the proof of the previous lemma, but not necessarily with the same constants). Take W such that $e^{-a\cdot W+b}<\epsilon_0/4$. Thus, with probability $\geq 1-\epsilon_0/4$:

$$f_{post\_mine}\left(n_x\left(G_s^u\right),g\left(G_s^u\right),l\left(G_s^u\right)\right)=$$

$$\sum_{k=0}^{\infty}\mathcal{P}_{oiss}(3\cdot d\cdot(1-\alpha)\cdot\lambda,k)\cdot\sum_{h=0}^{\infty}\mathcal{P}_{oiss}(d\cdot(1-\alpha)\cdot\lambda,h)\cdot \tag{75}$$

$$\left(\sum_{m'=M}^{\infty}\binom{n_x+j+h+m'-1}{m'}\cdot(1-\alpha)^{n_x+j+h}\cdot\alpha^{m'}\right)^{-1}\cdot \tag{76}$$

$$\sum_{m=M}^{\infty}\binom{n_x+j+h+m-1}{m}\cdot(1-\alpha)^{n_x+j+h}\cdot\alpha^{m}\cdot\left(\frac{\alpha}{1-\alpha}\right)^{(g-2\cdot h-k-j-l-(m-M))^+}. \tag{77}$$

For large enough n_x's, this term is at most $\epsilon_0/4$ away from

$$\left(\frac{\alpha}{1-\alpha}\right)^{g+M-n_x-W}\cdot\left(\sum_{m'=M}^{\infty}\binom{n_x+m'-1}{m'}\cdot(1-\alpha)^{n_x}\cdot\alpha^{m'}\right)^{-1}\cdot \tag{78}$$

$$\sum_{m=M}^{\infty}\binom{n_x+m-1}{m}\cdot(1-\alpha)^{n_x}\cdot\alpha^{m}\cdot\left(\frac{\alpha}{1-\alpha}\right)^{(n_x-m)^+}. \tag{79}$$

Part V: As for the first multiplicand of (78), by ***Part II*** of this proof, after some τ of finite expectation: $M\left(G_s^u\right)+g\left(G_s^u\right)-n_x\left(G_s^u\right)\geq-\left|G^{oracle}_{[time(x),t]}\right|-m^*=:D_2$ (a constant determined by time τ). Assume $s\geq\tau$. We conclude that the term $\left(\frac{\alpha}{1-\alpha}\right)^{g+M-n_x-W}$ is upper bounded by $e^{D_3\cdot D_4}$ (with $D_3=\ln\left(\frac{1-\alpha}{\alpha}\right)$). Thus, in order to show that (78) vanishes suffice it to show that

$$\left(\sum_{m'=M}^{\infty}\binom{n_x+m'-1}{m'}\cdot(1-\alpha)^{n_x}\cdot\alpha^{m'}\right)^{-1}\cdot \tag{80}$$

$$\sum_{m=M}^{\infty}\binom{n_x+m-1}{m}\cdot(1-\alpha)^{n_x}\cdot\alpha^{m}\cdot\left(\frac{\alpha}{1-\alpha}\right)^{(n_x-m)^+} \tag{81}$$

vanishes.

The last term equals

$$\left(\Pr_{m\sim Z(1-\alpha,n_x)}(m\geq M)\right)^{-1}\cdot\left(\Pr_{m\sim Z(\alpha,n_x)}(m\leq n_x)+\Pr_{m\sim Z(1-\alpha,n_x)}(m\geq n_x)\right). \tag{82}$$

For large enough n_x's, a variable distributed according to $Z(1-\alpha, n_x)$ converges to a normal variable with mean $n_x \cdot \frac{\alpha}{1-\alpha}$ and variance $n \cdot \frac{\alpha}{(1-\alpha)^2}$. The last term is therefore at most $\epsilon_0/4$ away from

$$\left(\Pr_{z\sim\mathcal{N}(0,1)}\left(z \geq \frac{M - \frac{\alpha}{1-\alpha}\cdot n_x}{\sqrt{\frac{\alpha}{(1-\alpha)^2}\cdot n_x}}\right)\right)^{-1} \cdot \tag{83}$$

$$\left(\Pr_{z\sim\mathcal{N}(0,1)}\left(z \leq \frac{n_x - \frac{1-\alpha}{\alpha}\cdot n_x}{\sqrt{\frac{1-\alpha}{\alpha^2}\cdot n_x}}\right) + \Pr_{z\sim\mathcal{N}(0,1)}\left(z \geq \frac{n_x - \frac{\alpha}{1-\alpha}\cdot n_x}{\sqrt{\frac{\alpha}{(1-\alpha)^2}\cdot n_x}}\right)\right) = \tag{84}$$

$$\left(\Pr_{z\sim\mathcal{N}(0,1)}\left(z \geq \frac{M - \frac{\alpha}{1-\alpha}\cdot n_x}{\sqrt{\frac{\alpha}{(1-\alpha)^2}\cdot n_x}}\right)\right)^{-1} \cdot \tag{85}$$

$$\left(\Pr_{z\sim\mathcal{N}(0,1)}\left(z \geq \frac{\frac{1-\alpha}{\alpha}\cdot n_x - n_x}{\sqrt{\frac{1-\alpha}{\alpha^2}\cdot n_x}}\right) + \Pr_{z\sim\mathcal{N}(0,1)}\left(z \geq \frac{n_x - \frac{\alpha}{1-\alpha}\cdot n_x}{\sqrt{\frac{\alpha}{(1-\alpha)^2}\cdot n_x}}\right)\right). \tag{86}$$

We use the following inequalities due to Komatu (1955), for $x \geq 0$, and a standard normal variable $z \sim \mathcal{N}(0,1)$: $\frac{1}{\sqrt{2\cdot\pi}} \cdot \frac{2\cdot e^{-x^2/2}}{x+\sqrt{4+x^2}} \leq \Pr(z > x) \leq \frac{1}{\sqrt{2\cdot\pi}} \cdot \frac{2\cdot e^{-x^2/2}}{x+\sqrt{2+x^2}}$.

Put $x_1 := \frac{M-\frac{\alpha}{1-\alpha}\cdot n_x}{\sqrt{\frac{\alpha}{(1-\alpha)^2}\cdot n_x}}$, $x_2 := \frac{\frac{1-\alpha}{\alpha}\cdot n_x - n_x}{\sqrt{\frac{1-\alpha}{\alpha^2}\cdot n_x}}$, and $x_3 := \frac{n_x - \frac{\alpha}{1-\alpha}\cdot n_x}{\sqrt{\frac{\alpha}{(1-\alpha)^2}\cdot n_x}}$.

We obtain an upper bound on (86):

$$\sqrt{\pi/2}\cdot\left(x_1+\sqrt{4+{x_1}^2}\right)\cdot e^{{x_1}^2/2}\cdot\left(\frac{1}{\sqrt{\pi/2}}\cdot\frac{e^{-x_2^2/2}}{x_2}+\frac{1}{\sqrt{\pi/2}}\cdot\frac{e^{-x_3^2/2}}{x_3}\right) = \tag{87}$$

$$\left(x_1+\sqrt{4+{x_1}^2}\right)\cdot e^{{x_1}^2/2}\cdot\left(\frac{e^{-x_2^2/2}}{x_2}+\frac{e^{-x_3^2/2}}{x_3}\right) \tag{88}$$

We further observe that, for large n_x's: $x_2 \geq C_2 \cdot \sqrt{n_x}$ and $x_3 \geq C_3 \cdot \sqrt{n_x}$, for some positive constants C_i (this applies to all constants below as well). Therefore, $\frac{\left(x_1+\sqrt{4+{x_1}^2}\right)}{\min\{x_2,x_3\}} \leq C_1/\max\{C_2, C_3\} =: D_1$. The above term is therefore upper bounded, up to a multiplicative factor of D_1, by

$$\begin{aligned}&e^{x_1^2/2-x_2^2/2}+e^{x_1^2/2-x_3^2/2} = \\ &e^{0.5\cdot\left(\left(\frac{M-\frac{\alpha}{1-\alpha}\cdot n_x}{\sqrt{\frac{\alpha}{(1-\alpha)^2}\cdot n_x}}\right)^2-\left(\frac{\frac{1-\alpha}{\alpha}\cdot n_x-n_x}{\sqrt{\frac{1-\alpha}{\alpha^2}\cdot n_x}}\right)^2\right)}+e^{0.5\cdot\left(\left(\frac{M-\frac{\alpha}{1-\alpha}\cdot n_x}{\sqrt{\frac{\alpha}{(1-\alpha)^2}\cdot n_x}}\right)^2-\left(\frac{n_x-\frac{\alpha}{1-\alpha}\cdot n_x}{\sqrt{\frac{\alpha}{(1-\alpha)^2}\cdot n_x}}\right)^2\right)} \leq \\ &e^{0.5\cdot\left(\frac{(1-\alpha)^2}{\alpha\cdot n_x}\cdot\left(M-\frac{\alpha}{1-\alpha}\cdot n_x\right)^2-\frac{(1-2\cdot\alpha)^2}{1-\alpha}\cdot n_x\right)}+e^{0.5\cdot\left(\frac{(1-\alpha)^2}{\alpha\cdot n_x}\cdot\left(M-\frac{\alpha}{1-\alpha}\cdot n_x\right)^2-\frac{(1-2\cdot\alpha)^2}{\alpha}\cdot n\right)}.\end{aligned} \tag{89}$$

Conditioned on the relation $\left|future_h\left(x, G_t^{oracle}\right)\right| \leq n_x$, $M \leq future_a\left(x, G_s^u\right)$, hence its expected value is at most $\frac{\alpha}{1-\alpha}\cdot n_x$. For any $\delta > 0$, by the Strong Law of Large Numbers, after some τ (of finite expectation), $\forall s \geq \tau : M \leq (1+\delta)\cdot\mathbb{E}[M] \leq (1+\delta)\cdot\frac{\alpha}{1-\alpha}\cdot n_x$.

Consequently, (89) is upper bounded by

$$e^{0.5\cdot\frac{(1-\alpha)^2}{\alpha\cdot n_x}\cdot\left(M-\frac{\alpha}{1-\alpha}\cdot n_x\right)^2-0.5\cdot\frac{(1-2\cdot\alpha)^2}{1-\alpha}\cdot n_x}+e^{0.5\cdot\frac{(1-\alpha)^2}{\alpha\cdot n_x}\cdot\left(M-\frac{\alpha}{1-\alpha}\cdot n_x\right)^2-0.5\cdot\frac{(1-2\cdot\alpha)^2}{\alpha}\cdot n_x}\leq \tag{90}$$

$$e^{0.5\cdot\frac{(1-\alpha)^2}{\alpha\cdot n_x}\cdot\left(\delta\cdot\frac{\alpha}{1-\alpha}\cdot n_x\right)^2-0.5\cdot\frac{(1-2\cdot\alpha)^2}{1-\alpha}\cdot n_x}+e^{0.5\cdot\frac{(1-\alpha)^2}{\alpha\cdot n_x}\cdot\left(\delta\cdot\frac{\alpha}{1-\alpha}\cdot n_x\right)^2-0.5\cdot\frac{(1-2\cdot\alpha)^2}{\alpha}\cdot n_x}\leq \tag{91}$$

$$e^{R_1/n_x-R_2\cdot n_x}+e^{R^3/n_x-R_4\cdot n_x}\leq e^{-R_5\cdot n_x}, \tag{92}$$

for some positive constants R_i, where the last inequality holds for large enough n_x's, and the preceding inequality holds for small enough δ's ($\delta < 1/n_x$).

Taking n_x to be greater than $n_x > \ln\left(4\cdot D_1/\epsilon_0\right)/R_5$ we conclude that for some large enough n_x:

$$f_{post\_mine}\left(n_x\left(G_s^u\right),g\left(G_s^u\right),l\left(G_s^u\right)\right) < 4\cdot\epsilon_0/4=\epsilon_0. \tag{93}$$

(Note that the expected waiting time for the first τ such that $\forall j\in \mathit{honest}: n_x(u,\psi)$ is at least some n_0 is at most $n_0\cdot\left((1-\alpha)\cdot\lambda\right)^{-1}+d$: it is $1/((1-\alpha)\cdot\lambda)$ for the creation of every honest block, and d for the last one to arrive at all nodes.) □

The same technique used in the proof of Lemma 17 is used below to prove the Progress property (Proposition 7); indeed, in the proof we see that the term that aggregates all the error functions vanishes as time develops, w.h.p., in the perspective of all honest nodes. In particular, for v (the node that originally ϵ-accepted the transaction), it becomes smaller than ϵ' w.h.p. A yet similar usage of this argument is used below to prove Weak Liveness (Proposition 8); indeed, in the latter we only need to regard the case where $y = NULL$. In this case, all published blocks are strong voters in favour of x, and so we can guarantee the convergence of the error functions without going through Lemma 12 and the analysis that follows.

G. Proof of Weak Liveness (blocks)

We've seen that the error functions $f_{pre\_mine}(l\left(G_s^u\right))$, $f_{pre\_pub}\left(n_j\left(G_s^u\right)\right)$, and $f_{post\_pub}\left(\left|\mathit{future}\left(x,G_s^u\right)\right|\right)$ go to 0 as s grows. For any $s<\psi$, $y\notin G_s^{pub}$, hence by line 5 of Algorithm 3, $g\left(G_s^v\right)=\left|\mathit{future}\left(x,G\right)\right|=n_x\left(G_s^u\right)$, and $M\left(G_s^v\right)=0$. In particular, the relation (73) is satisfied trivially, and the analysis in the proof of Lemma 17 applies, proving that the term f_{post_mine} vanishes as time grows. In particular, since these functions decrease exponentially, it becomes smaller than ϵ after a number of honest blocks in the order of $\mathcal{O}(\ln(1/\epsilon))$ are created, and the expected waiting time for this is obtained by dividing this number by $(1-\alpha)\cdot\lambda$ (and adding d for all honest blocks to receive these blocks).

H. Proof of Progress (blocks)

This follows immediately from the proof of Lemma 17, in which it was shown that, conditioned on the event $\widehat{\mathcal{E}}^{all}_{\rightarrow\infty}t(x,y)$, $f_{pre\_mine}(l\left(G_s^v\right))+f_{pre\_pub}\left(n_j\left(G_s^u\right)\right)+f_{post\_pub}\left(\left|\mathit{future}\left(x,G_t^u\right)\right|\right)+f_{post\_mine}\left(n_x\left(G_s^u\right),g\left(G_s^u\right),l\left(G_s^u\right)\right)$ vanishes as s grows indefinitely. In Lemma 12 it was shown that, up to a probability of ϵ, the event $\mathcal{E}_t^v(x,y,\epsilon)$ is contained in $\widehat{\mathcal{E}}^{all}_{\rightarrow\infty}t(x,y)$ (i.e., when the former is intersected with an event of probability $\geq 1-\epsilon$).

I. Proof of Safety

Part I: Denote by $risk_{acc}(G_s^u, tx, subG)$ ($risk_{rej}$) the output of Algorithm 4 (respectively, Algorithm 5) when given the inputs G_s^u (for some honest u), tx, and $subG$ (such that $subG$ is the past of some (possibly virtual) block). For any $z \in [tx] \cap subG$, denote by $risk^z_{acc}(G_s^u, tx, subG)$ the value of the $risk$ variable as the loop in line 2 of $RiskTxAccept$ terminates its run over z. Denote similarly $minrisk^z_{rej}(G_s^u, tx, subG)$ w.r.t. the variable $minrisk$ in $RiskTxReject$.

We claim that, with probability $> 1 - risk_{acc}(G_t^v, tx, subG)$, there exists a τ_{acc} of finite expectation such that for all $s \geq \tau_{acc}$, for all $u \in honest$, and for all $subG' \supseteq subG$:

$$risk_{acc}(G_t^v, tx, subG) \geq risk_{acc}(G_s^u, tx, subG') \tag{94}$$

Similarly, we claim that, with probability $> 1 - risk_{rej}(G_t^v, tx, subG)$, there exists a τ_{rej} of finite expectation such that for all $s \geq \tau_{rej}$, for all $u \in honest$:

$$risk_{rej}(G_t^v, tx, subG) \geq risk_{rej}(G_s^u, tx, subG). \tag{95}$$

Assume we have proved this for all $subG$ of size $< k$. We now prove this for $subG_k$ of size k.

By the definition of $risk_{acc}$, there exists a $z_{tx} \in subG_k \cap [tx]$ such that $risk_{acc}(G_t^v, tx, subG') = risk^{z_{tx}}_{acc}(G_t^v, tx, subG')$.

Part II: Denote by Z_2 the set of instantiations of the third-loop-variable z_2, inside the iteration of the first-loop with $z_1 = z_{tx}$. By Propositions 6 and 7, $\forall z_2 \in Z_2$, with probability $\geq 1 - Risk\left(G_t^v, (vote\,(z'))_{z' \in C}, z_1, z_2\right)$, for any ϵ', after some τ (of finite expectation), $\forall z_2' \in (G_s^u \setminus G_t^v) \cup \{z_2\} : Risk\left(G_s^u, (vote\,(z))_{z \in C}, z_{tx}, z_2'\right) \leq \epsilon'$. Moreover, in the proof of Proposition 17 it was shown that the minimal ϵ' for which this property holds at time s decreases exponentially with n (which grows linearly with s). Thus, for all s greater than some τ, $\sum_{z_2' \in (G_s^u \setminus G_t^v) \cup Z_2} : Risk\left(G_s^u, (vote\,(z))_{z \in C}, z_{tx}, z_2'\right) \leq \sum_{z_2' \in Z_2} Risk\left(G_t^v, (vote\,(z))_{z \in C}, z_{tx}, z_2'\right)$.

Part III: Similarly, by Proposition 6, with probability of at least $Risk\left(G_t^v, (vote\,(z))_{z \in C}, z_{tx}, \emptyset\right)$, after some τ (of finite expectation), $Risk\left(G_s^u, (vote\,(z))_{z \in C}, z_{tx}, \emptyset\right) \leq Risk\left(G_t^v, (vote\,(z))_{z \in C}, z_{tx}, \emptyset\right)$.

Part IV: Let $\epsilon_i(G_s^u, tx, subG)$ be the series of values returned by the call to $RiskTxAccept$ in line 7 of $RiskTxAccept$ (when given the inputs $(G_s^u, tx, subG)$) and to $RiskTxReject$ in line 9 of $RiskTxAccept$ (with these inputs). By the induction hypothesis, with probability $\geq 1 - \epsilon_i$, after some time τ, $\epsilon_i(G_s^u, tx, past\,(z_{tx})) \leq \epsilon_i(G_t^v, tx, past\,(z_{tx}))$.[26]

Part V: The above arguments show that, with probability $\geq 1 - risk^{z_{tx}}_{acc}(G_t^v, tx, subG)$, the sum of increments to the value of $risk^{z_{tx}}_{acc}(G_s^u, tx, subG')$ is upper bounded by the

[26]Technically, the indexes i on both hand-sides of this inequality should be described more carefully. To save cumbersome notation, we rely on the understanding of the reader. Informally, every instantiation of the loop-variables inside $RiskTxAccept$ (when given the inputs $(G_t^v, tx, subG)$) is also realized by future calls of $RiskTxAccept$ (when given the inputs $(G_s^u, tx, subG')$). We thus compare the results of the increments in the former to those in the latter. This is also true vice versa (for $z_1 = z_{tx}$): Inside the first-loop's iteration over $z_1 = z_{tx}$, the exact same calls to $RiskTxAccept$ and $RiskTxReject$ are made, because $past\,(z_{tx})$ does not evolve with time.

the sum of increments to the value of $risk_{acc}^{z_{tx}}(G_t^v, tx, subG)$, for all $s \geq \tau$, where τ is of finite expectation. As $risk_{acc}(G_s^u, tx, subG') \leq risk_{acc}^{z_{tx}}(G_s^u, tx, subG')$, and as $risk_{acc}(G_t^v, tx, subG) = risk_{acc}^{z_{tx}}(G_t^v, tx, subG)$, this proves that, with probability $\geq 1 - risk_{acc}^{z_{tx}}(G_t^v, tx, subG)$ the inequality $risk_{acc}(G_s^u, tx, subG') \leq risk_{acc}(G_t^v, tx, subG)$ holds.

Part VI: Similar arguments prove the induction step w.r.t. $RiskTxReject$. The difference in the proof is that, since $risk_{rej}^{z_1}$ is not a sum, rather a minimum, hence we can ignore the fact that $anticone\,(z_1, G_s^u)$ may grow in time and add loop-iterations that might further reduce the value of $risk_{rej}^{z_1}$. Note further that the induction claim, w.r.t. $RiskTxReject$, is restricted to the case $subG' = subG$. Hence, the fact that the set $Z_G([tx])$ possibly grows with time is of no consequence, since the first loop-variable is chosen from $Z_G([tx]) \cap subG$. We thus conclude that, with probability $\geq 1 - risk_{rej}(G_s^u, tx, subG_k)$, there exists a τ of finite expectation such that for all $s \geq \tau$ and all $u \in honest$: $risk_{rej}(G_s^u, tx, subG_k) \leq risk_{rej}(G_t^v, tx, subG_k)$.

This completes the proof of the induction claim.

Part VII: Algorithm 6 returns ACCEPT if and only if $RiskTxAccept$ returned a value smaller than ϵ. The above claim implies that, if $risk_{acc}(G_t^v, tx, G_t^v) < \epsilon$, with probability $\geq 1 - \epsilon$, for all $s \geq \tau$, for some τ of finite expectation, for all $u \in honest$: $risk_{acc}(G_s^u, tx, G_s^u) < \epsilon$. In other words, conditioned on $\mathcal{A}_t^v(tx, \epsilon)$, the event $\cap_{u \in honest, s \in (\tau(t), \infty)} \mathcal{A}_s^u(tx, \epsilon)$ occurs with probability $\geq 1 - \epsilon$.

J. Proof of Liveness

Fix some $z_1 \in Z_G([tx])$ for $G = G_t^v$. The condition that until $\psi(t)$: $conflict\,(tx) \cap G_s^{pub} = \emptyset$, implies that lines 6 and 7 of $RiskTxAccept$ do not contribute to the value of $risk_{acc}(G_s^u, tx, subG)$. The assumption $\sum_{[tx_i] \in inputs(tx)} RiskTxAccept\left(G_t^v, (vote\,(z))_{z \in C}, [tx_i], G_t^v\right) < \epsilon/2$ implies that, with probability $\geq 1 - \epsilon/2$, the overall contribution of the fourth loop to the value of $risk_{acc}^{z_1}(G_s^u, [tx], G_s^u)$ is at most $\epsilon/2$ (after some τ). Finally, by Proposition 8, the contribution of line 3 to $risk_{acc}^{z_1}$ is less than $\epsilon/2$, after some τ of finite expectation. We conclude that after some τ of finite expectation, the value of $risk_{acc}^{z_1}(G_s^u, [tx], G_s^u)$ is smaller than $\epsilon/2 + \epsilon/2 = \epsilon$, for all $s \geq \tau$ and $u \geq s$, hence $risk_{acc}(G_s^u, [tx], G_s^u) < \epsilon$, which implies the event $\cap_{u \in honest, s \in (\tau(t), \infty)} \mathcal{A}_s^u(tx, \epsilon)$.

K. Proof of Progress

The proof of this proposition is similar in structure to that of Proposition 2. Therein we have already argued that the contributions to the value of $risk_{acc}^{z_{tx}}$ (and similarly for $risk_{rej}^{z_{tx}}$) of lines 3 and of 6 go to 0; and the increments of lines 7 and 9 go to 0 by the induction hypothesis. Thus, $risk_{acc}(G_s^v, tx, G_s^v)$ goes to 0 as time develops, with probability $\geq 1 - risk_{acc}(G_t^v, tx, G_t^v)$. As $\epsilon > risk_{acc}(G_t^v, tx, G_t^v)$, we conclude that, with probability $\geq 1 - \epsilon$ Algorithm 6 returns ACCEPT for all G_s^u with $s \geq \tau$ and $u \in honest$.