not randomly and independently selected. We shall prove in our analysis, however, that Q^{r-1} is sufficiently non-manipulatable to guarantee that the leader of a round is honest with probability h' sufficiently close to h: namely, $h' > h^2(1 + h - h^2)$. For instance, if $h = 80\%$, then $h' > .7424$.

Having identified the leader of the round (which they correctly do when the leader ℓ^r is honest), the task of the step-2 verifiers is to start executing the BA using as initial values what they believe to be the block of the leader. Actually, in order to minimize the amount of communication required, a verifier $j \in SV^{r,2}$ does not use, as his input value v'_j to the Byzantine protocol, the block B_j that he has actually received from ℓ_j (the user j believes to be the leader), but the the leader, but the hash of that block, that is, $v'_j = H(B_i)$. Thus, upon termination of the BA protocol, the verifiers of the last step do not compute the desired round-r block B^r, but compute (authenticate and propagate) $H(B^r)$. Accordingly, since $H(B^r)$ is digitally signed by sufficiently many verifiers of the last step of the BA protocol, the users in the system will realize that $H(B^r)$ is the hash of the new block. However, they must also retrieve (or wait for, since the execution is quite asynchronous) the block B^r itself, which the protocol ensures that is indeed available, no matter what the Adversary might do.

Asynchrony and Timing *Algorand*$'_1$ and *Algorand*$'_2$ have a significant degree of asynchrony. This is so because the Adversary has large latitude in scheduling the delivery of the messages being propagated. In addition, whether the total number of steps in a round is capped or not, there is the variance contribute by the number of steps actually taken.

As soon as he learns the certificates of $B^0, \ldots, B^{r-1}$, a user i computes Q^{r-1} and starts working on round r, checking whether he is a potential leader, or a verifier in some step s of round r.

Assuming that i must act at step s, in light of the discussed asynchrony, i relies on various strategies to ensure that he has sufficient information before he acts.

For instance, he might wait to receive at least a given number of messages from the verifiers of the previous step, or wait for a sufficient time to ensure that he receives the messages of sufficiently many verifiers of the previous step.

The Seed Q^r and the Look-Back Parameter k Recall that, ideally, the quantities Q^r should random and independent, although it will suffice for them to be sufficiently non-manipulatable by the Adversary.

At a first glance, we could choose Q^{r-1} to coincide with $H\left(PAY^{r-1}\right)$, and thus avoid to specify Q^{r-1} explicitly in B^{r-1}. An elementary analysis reveals, however, that malicious users may take advantage of this selection mechanism.[11] Some additional effort shows that myriads of other

[11] We are at the start of round $r-1$. Thus, $Q^{r-2} = PAY^{r-2}$ is publicly known, and the Adversary privately knows who are the potential leaders he controls. Assume that the Adversary controls 10% of the users, and that, with very high probability, a malicious user w is the potential leader of round $r-1$. That is, assume that $H\left(SIG_w\left(r-2, 1, Q^{r-2}\right)\right)$ is so small that it is highly improbable an honest potential leader will actually be the leader of round $r-1$. (Recall that, since we choose potential leaders via a secret cryptographic sortition mechanism, the Adversary does not know who the honest potential leaders are.) The Adversary, therefore, is in the enviable position of choosing the payset PAY' he wants, and have it become the official payset of round $r-1$. However, he can do more. He can also ensure that, with high probability, (*) one of his malicious users will be the leader also of round r, so that he can freely select what PAY^r will be. (And so on. At least for a long while, that is, as long as these high-probability events really occur.) To guarantee (*), the Adversary acts as follows. Let PAY' be the payset the Adversary prefers for round $r-1$. Then, he computes $H(PAY')$ and checks whether, for some already malicious player z, $SIG_z(r, 1, H(PAY'))$ is particularly small, that is, small enough that with very high probability z will be the leader of round r. If this is the case, then he instructs w to choose his candidate block to be

alternatives, based on traditional block quantities are easily exploitable by the Adversary to ensure that malicious leaders are very frequent. We instead specifically and inductively define our brand new quantity Q^r so as to be able to prove that it is non-manipulatable by the Adversary. Namely,

$$Q^r \triangleq H(SIG_{\ell^r}(Q^{r-1}), r), \text{ if } B^r \text{ is not the empty block, and } Q^r \triangleq H(Q^{r-1}, r) \text{ otherwise.}$$

The intuition of why this construction of Q^r works is as follows. Assume for a moment that Q^{r-1} is truly randomly and independently selected. Then, will so be Q^r? When ℓ^r is honest the answer is (roughly speaking) yes. This is so because

$$H(SIG_{\ell^r}(\cdot), r) : \{0,1\}^{256} \longrightarrow \{0,1\}^{256}$$

is a random function. When ℓ^r is malicious, however, Q^r is no longer univocally defined from Q^{r-1} and ℓ^r. There are at least two separate values for Q^r. One continues to be $Q^r \triangleq H(SIG_{\ell^r}(Q^{r-1}), r)$, and the other is $H(Q^{r-1}, r)$. Let us first argue that, while the second choice is somewhat arbitrary, a second choice is absolutely mandatory. The reason for this is that a malicious ℓ^r can always cause totally different candidate blocks to be received by the honest verifiers of the second step.[12] Once this is the case, it is easy to ensure that the block ultimately agreed upon via the BA protocol of round r will be the default one, and thus will not contain anyone's digital signature of Q^{r-1}. But the system must continue, and for this, it needs a leader for round r. If this leader is automatically and openly selected, then the Adversary will trivially corrupt him. If it is selected by the previous Q^{r-1} via the same process, than ℓ^r will again be the leader in round $r+1$. We specifically propose to use the same secret cryptographic sortition mechanism, but applied to a new Q-quantity: namely, $H(Q^{r-1}, r)$. By having this quantity to be the output of H guarantees that the output is random, and by including r as the second input of H, while all other uses of H have one or 3+ inputs, "guarantees" that such a Q^r is independently selected. Again, our specific choice of alternative Q^r does not matter, what matter is that ℓ^r has two choice for Q^r, and thus he can double his chances to have another malicious user as the next leader.

The options for Q^r may even be more numerous for the Adversary who controls a malicious ℓ^r. For instance, let x, y, and z be three malicious potential leaders of round r such that

$$H\left(\sigma_x^{r,1}\right) < H\left(\sigma_y^{r,1}\right) < H\left(\sigma_z^{r,1}\right)$$

and $H\left(\sigma_z^{r,1}\right)$ is particulary small. That is, so small that there is a good chance that $H\left(\sigma_z^{r,1}\right)$ is smaller of the hashed credential of every honest potential leader. Then, by asking x to hide his credential, the Adversary has a good chance of having y become the leader of round $r-1$. This implies that he has another option for Q^r: namely, $SIG_y\left(Q^{r-1}\right)$. Similarly, the Adversary may ask both x and y of withholding their credentials, so as to have z become the leader of round $r-1$ and gaining another option for Q^r: namely, $SIG_z\left(Q^{r-1}\right)$.

Of course, however, each of these and other options has a non-zero chance to fail, because the Adversary cannot predict the hash of the digital signatures of the honest potential users.

$B_i^{r-1} = (r-1, PAY', H(B^{r-2})$. Else, he has two other malicious users x and y to keep on generating a new payment $\wp'$, from one to the other, until, for some malicious user z (or even for some fixed user z) $H\left(SIG_z\left(PAY' \cup \{\wp\}\right)\right)$ is particularly small too. This experiment will stop quite quickly. And when it does the Adversary asks w to propose the candidate block $B_i^{r-1} = (r-1, PAY' \cup \{\wp\}, H(B^{r-2})$.

[12] For instance, to keep it simple (but extreme), "when the time of the second step is about to expire", ℓ^r could directly email a different candidate block B_i to each user i. This way, whoever the step-2 verifiers might be, they will have received totally different blocks.

A careful, Markov-chain-like analysis shows that, no matter what options the Adversary chooses to make at round $r-1$, *as long as he cannot inject new users in the system*, he cannot decrease the probability of an honest user to be the leader of round $r+40$ much below h. This is the reason for which we demand that the potential leaders of round r are users already existing in round $r-k$. It is a way to ensure that, at round $r-k$, the Adversary cannot alter by much the probability that an honest user become the leader of round r. In fact, no matter what users he may add to the system in rounds $r-k$ through r, they are ineligible to become potential leaders (and *a fortiori* the leader) of round r. Thus the look-back parameter k ultimately is a security parameter. (Although, as we shall see in section 7, it can also be a kind of "convenience parameter" as well.)

Ephemeral Keys Although the execution of our protocol cannot generate a fork, except with negligible probability, the Adversary could generate a fork, at the rth block, after the legitimate block r has been generated.

Roughly, once B^r has been generated, the Adversary has learned who the verifiers of each step of round r are. Thus, he could therefore corrupt all of them and oblige them to certify a new block $\widetilde{B^r}$. Since this fake block might be propagated only after the legitimate one, users that have been paying attention would not be fooled.[13] Nonetheless, $\widetilde{B^r}$ would be syntactically correct and we want to prevent from being manufactured.

We do so by means of a new rule. Essentially, the members of the verifier set $SV^{r,s}$ of a step s of round r use ephemeral public keys $pk_i^{r,s}$ to digitally sign their messages. These keys are single-use-only and their corresponding secret keys $sk_i^{r,s}$ are destroyed once used. This way, if a verifier is corrupted later on, the Adversary cannot force him to sign anything else he did not originally sign.

Naturally, we must ensure that it is impossible for the Adversary to compute a new key $\widetilde{p_i^{r,s}}$ and convince an honest user that it is the right ephemeral key of verifier $i \in SV^{r,s}$ to use in step s.

4.2 Common Summary of Notations, Notions, and Parameters

Notations

- $r \geq 0$: the current round number.
- $s \geq 1$: the current step number in round r.
- B^r: the block generated in round r.
- PK^r: the set of public keys by the end of round $r-1$ and at the beginning of round r.
- S^r: the system status by the end of round $r-1$ and at the beginning of round r.[14]
- PAY^r: the payset contained in B^r.
- ℓ^r: round-r leader. ℓ^r chooses the payset PAY^r of round r (and determines the next Q^r).
- Q^r: the seed of round r, a quantity (i.e., binary string) that is generated at the end of round r and is used to choose verifiers for round $r+1$. Q^r is independent of the paysets in the blocks and cannot be manipulated by ℓ^r.

[13] Consider corrupting the news anchor of a major TV network, and producing and broadcasting today a newsreel showing secretary Clinton winning the last presidential election. Most of us would recognize it as a hoax. But someone getting out of a coma might be fooled.

[14] In a system that is not synchronous, the notion of "the end of round $r-1$" and "the beginning of round r" need to be carefully defined. Mathematically, PK^r and S^r are computed from the initial status S^0 and the blocks $B^1, \ldots, B^{r-1}$.

- $SV^{r,s}$: the set of verifiers chosen for step s of round r.
- SV^r: the set of verifiers chosen for round r, $SV^r = \cup_{s \geq 1} SV^{r,s}$.
- $MSV^{r,s}$ and $HSV^{r,s}$: respectively, the set of malicious verifiers and the set of honest verifiers in $SV^{r,s}$. $MSV^{r,s} \cup HSV^{r,s} = SV^{r,s}$ and $MSV^{r,s} \cap HSV^{r,s} = \emptyset$.
- $n_1 \in \mathbb{Z}^+$ and $n \in \mathbb{Z}^+$: respectively, the expected numbers of potential leaders in each $SV^{r,1}$, and the expected numbers of verifiers in each $SV^{r,s}$, for $s > 1$.
  Notice that $n_1 << n$, since we need at least one honest honest member in $SV^{r,1}$, but at least a majority of honest members in each $SV^{r,s}$ for $s > 1$.
- $h \in (0, 1)$: a constant greater than $2/3$. h is the honesty ratio in the system. That is, the fraction of honest users or honest money, depending on the assumption used, in each PK^r is at least h.
- H: a cryptographic hash function, modelled as a random oracle.
- $\perp$: A special string of the same length as the output of H.
- $F \in (0, 1)$: the parameter specifying the allowed error probability. A probability $\leq F$ is considered "negligible", and a probability $\geq 1 - F$ is considered "overwhelming".
- $p_h \in (0, 1)$: the probability that the leader of a round r, ℓ^r, is honest. Ideally $p_h = h$. With the existence of the Adversary, the value of p_h will be determined in the analysis.
- $k \in \mathbb{Z}^+$: the look-back parameter. That is, round $r - k$ is where the verifiers for round r are chosen from —namely, $SV^r \subseteq PK^{r-k}$.[15]
- $p_1 \in (0, 1)$: for the first step of round r, a user in round $r - k$ is chosen to be in $SV^{r,1}$ with probability $p_1 \triangleq \frac{n_1}{|PK^{r-k}|}$.
- $p \in (0, 1)$: for each step $s > 1$ of round r, a user in round $r - k$ is chosen to be in $SV^{r,s}$ with probability $p \triangleq \frac{n}{|PK^{r-k}|}$.
- $CERT^r$: the certificate for B^r. It is a set of t_H signatures of $H(B^r)$ from proper verifiers in round r.
- $\overline{B^r} \triangleq (B^r, CERT^r)$ is a proven block.
  A user i *knows* B^r if he possesses (and successfully verifies) both parts of the proven block. Note that the $CERT^r$ seen by different users may be different.
- τ_i^r: the (local) time at which a user i knows B^r. In the Algorand protocol each user has his own clock. Different users' clocks need not be synchronized, but must have the same speed. Only for the purpose of the analysis, we consider a reference clock and measure the players' related times with respect to it.
- $\alpha_i^{r,s}$ and $\beta_i^{r,s}$: respectively the (local) time a user i starts and ends his execution of Step s of round r.
- Λ and λ: essentially, the upper-bounds to, respectively, the time needed to execute Step 1 and the time needed for any other step of the Algorand protocol.
  Parameter Λ upper-bounds the time to propagate a single 1MB block. (In our notation, $\Lambda = \lambda_{\rho,1MB}$. Recalling our notation, that we set $\rho = 1$ for simplicity, and that blocks are chosen to be at most 1MB-long, we have $\Lambda = \lambda_{1,1,1MB}$.)

[15] Strictly speaking, "$r - k$" should be "$\max\{0, r - k\}$".

Parameter λ upperbounds the time to propagate one small message per verifier in a Step $s > 1$. (Using, as in Bitcoin, elliptic curve signatures with 32B keys, a verifier message is 200B long. Thus, in our notation, $\lambda = \lambda_{n,\rho,200B}$.)
We assume that $\Lambda = O(\lambda)$.

Notions

- Verifier selection.

  For each round r and step $s > 1$, $SV^{r,s} \triangleq \{i \in PK^{r-k} : .H(SIG_i(r, s, Q^{r-1})) \leq p\}$. Each user $i \in PK^{r-k}$ privately computes his signature using his long-term key and decides whether $i \in SV^{r,s}$ or not. If $i \in SV^{r,s}$, then $SIG_i(r, s, Q^{r-1})$ is i's (r, s)-credential, compactly denoted by $\sigma_i^{r,s}$.

  For the first step of round r, $SV^{r,1}$ and $\sigma_i^{r,1}$ are similarly defined, with p replaced by p_1. The verifiers in $SV^{r,1}$ are *potential leaders.*

- Leader selection.

  User $i \in SV^{r,1}$ is the leader of round r, denoted by ℓ^r, if $H(\sigma_i^{r,1}) \leq H(\sigma_j^{r,1})$ for all potential leaders $j \in SV^{r,1}$. Whenever the hashes of two players' credentials are compared, in the unlikely event of ties, the protocol always breaks ties lexicographically according to the (long-term public keys of the) potential leaders.

  By definition, the hash value of player ℓ^r's credential is also the smallest among all users in PK^{r-k}. Note that a potential leader cannot privately decide whether he is the leader or not, without seeing the other potential leaders' credentials.

  Since the hash values are uniform at random, when $SV^{r,1}$ is non-empty, ℓ^r always exists and is honest with probability at least h. The parameter n_1 is large enough so as to ensure that each $SV^{r,1}$ is non-empty with overwhelming probability.

- Block structure.

  A non-empty block is of the form $B^r = (r, PAY^r, SIG_{\ell^r}(Q^{r-1}), H(B^{r-1}))$, and an empty block is of the form $B_\epsilon^r = (r, \emptyset, Q^{r-1}, H(B^{r-1}))$.

  Note that a non-empty block may still contain an empty payset PAY^r, if no payment occurs in this round or if the leader is malicious. However, a non-empty block implies that the identity of ℓ^r, his credential $\sigma_{\ell^r}^{r,1}$ and $SIG_{\ell^r}(Q^{r-1})$ have all been timely revealed. The protocol guarantees that, if the leader is honest, then the block will be non-empty with overwhelming probability.

- Seed Q^r.

  If B^r is non-empty, then $Q^r \triangleq H(SIG_{\ell^r}(Q^{r-1}), r)$, otherwise $Q^r \triangleq H(Q^{r-1}, r)$.

Parameters

- *Relationships among various parameters.*
  - The verifiers and potential leaders of round r are selected from the users in PK^{r-k}, where k is chosen so that the Adversary cannot predict Q^{r-1} back at round $r - k - 1$ with probability better than F: otherwise, he will be able to introduce malicious users for round $r - k$, all of which will be potential leaders/verifiers in round r, succeeding in

having a malicious leader or a malicious majority in $SV^{r,s}$ for some steps s desired by him.

— For Step 1 of each round r, n_1 is chosen so that with overwhelming probability, $SV^{r,1} \neq \emptyset$.

- *Example choices of important parameters.*
  — The outputs of H are 256-bit long.
  — $h = 80\%$, $n_1 = 35$.
  — $\Lambda = 1$ minute and $\lambda = 10$ seconds.
- *Initialization of the protocol.*
  The protocol starts at time 0 with $r = 0$. Since there does not exist "B^{-1}" or "$CERT^{-1}$", syntactically B^{-1} is a public parameter with its third component specifying Q^{-1}, and all users know B^{-1} at time 0.

5 *Algorand'*$_1$

In this section, we construct a version of *Algorand'* working under the following assumption.

Honest Majority of Users Assumption: *More than 2/3 of the users in each PK^r are honest.*

In Section 8, we show how to replace the above assumption with the desired Honest Majority of Money assumption.

5.1 Additional Notations and Parameters

Notations

- $m \in \mathbb{Z}^+$: the maximum number of steps in the binary BA protocol, a multiple of 3.
- $L^r \leq m/3$: a random variable representing the number of Bernoulli trials needed to see a 1, when each trial is 1 with probability $\frac{p_h}{2}$ and there are at most $m/3$ trials. If all trials fail then $L^r \triangleq m/3$. L^r will be used to upper-bound the time needed to generate block B^r.
- $t_H = \frac{2n}{3} + 1$: the number of signatures needed in the ending conditions of the protocol.
- $CERT^r$: the certificate for B^r. It is a set of t_H signatures of $H(B^r)$ from proper verifiers in round r.

Parameters

- *Relationships among various parameters.*
  — For each step $s > 1$ of round r, n is chosen so that, with overwhelming probability,
  $$|HSV^{r,s}| > 2|MSV^{r,s}| \quad \text{and} \quad |HSV^{r,s}| + 4|MSV^{r,s}| < 2n.$$
  The closer to 1 the value of h is, the smaller n needs to be. In particular, we use (variants of) Chernoff bounds to ensure the desired conditions hold with overwhelming probability.
  — m is chosen such that $L^r < m/3$ with overwhelming probability.
- *Example choices of important parameters.*
  — $F = 10^{-12}$.
  — $n \approx 1500$, $k = 40$ and $m = 180$.

5.2 Implementing Ephemeral Keys in *Algorand*$'_1$

As already mentioned, we wish that a verifier $i \in SV^{r,s}$ digitally signs his message $m_i^{r,s}$ of step s in round r, relative to an ephemeral public key $pk_i^{r,s}$, using an ephemeral secrete key $sk_i^{r,s}$ that he promptly destroys after using. We thus need an efficient method to ensure that every user can verify that $pk_i^{r,s}$ is indeed the key to use to verify i's signature of $m_i^{r,s}$. We do so by a (to the best of our knowledge) new use of identity-based signature schemes.

At a high level, in such a scheme, a central authority A generates a public master key, PMK, and a corresponding secret master key, SMK. Given the identity, U, of a player U, A computes, via SMK, a secret signature key sk_U relative to the public key U, and privately gives sk_U to U. (Indeed, in an identity-based digital signature scheme, the public key of a user U is U itself!) This way, if A destroys SMK after computing the secret keys of the users he wants to enable to produce digital signatures, and does not keep any computed secret key, then U is the only one who can digitally sign messages relative to the public key U. Thus, anyone who knows "U's name", automatically knows U's public key, and thus can verify U's signatures (possibly using also the public master key PMK).

In our application, the authority A is user i, and the set of all possible users U coincides with the round-step pair (r, s) in —say— $S = \{i\} \times \{r', \ldots, r' + 10^6\} \times \{1, \ldots, m+3\}$, where r' is a given round, and $m + 3$ the upperbound to the number of steps that may occur within a round. This way, $pk_i^{r,s} \triangleq (i, r, s)$, so that everyone seeing i's signature $SIG^{r,s}_{pk_i^{r,s}}(m_i^{r,s})$ can, with overwhelming probability, immediately verify it for the first million rounds r following r'.

In other words, i first generates PMK and SMK. Then, he publicizes that PMK is i's master public key for any round $r \in [r', r' + 10^6]$, and uses SMK to privately produce and store the secret key $sk_i^{r,s}$ for each triple $(i, r, s) \in S$. This done, he destroys SMK. If he determines that he is not part of $SV^{r,s}$, then i may leave $sk_i^{r,s}$ alone (as the protocol does not require that he aunthenticates any message in Step s of round r). Else, i first uses $sk_i^{r,s}$ to digitally sign his message $m_i^{r,s}$, and then destroys $sk_i^{r,s}$.

Note that i can publicize his first public master key when he first enters the system. That is, the same payment $\wp$ that brings i into the system (at a round r' or at a round close to r'), may also specify, at i's request, that i's public master key for any round $r \in [r', r' + 10^6]$ is PMK —e.g., by including a pair of the form $(PMK, [r', r' + 10^6])$.

Also note that, since $m+3$ is the maximum number of steps in a round, assuming that a round takes a minute, the stash of ephemeral keys so produced will last i for almost two years. At the same time, these ephemeral secret keys will not take i too long to produce. Using an elliptic-curve based system with 32B keys, each secret key is computed in a few microseconds. Thus, if $m + 3 = 180$, then all 180M secret keys can be computed in less than one hour.

When the current round is getting close to $r' + 10^6$, to handle the next million rounds, i generates a new (PMK', SMK') pair, and informs what his next stash of ephemeral keys is by —for example— having $SIG_i(PMK', [r' + 10^6 + 1, r' + 2 \cdot 10^6 + 1])$ enter a new block, either as a separate "transaction" or as some additional information that is part of a payment. By so doing, i informs everyone that he/she should use PMK' to verify i's ephemeral signatures in the next million rounds. And so on.

(Note that, following this basic approach, other ways for implementing ephemeral keys without using identity-based signatures are certainly possible. For instance, via Merkle trees.[16])

[16] In this method, i generates a public-secret key pair $(pk_i^{r,s}, sk_i^{r,s})$ for each round-step pair (r, s) in —say—

Other ways for implementing ephemeral keys are certainly possible —e.g., via Merkle trees.

5.3 Matching the Steps of $Algorand'_1$ with those of $BA^\star$

As we said, a round in $Algorand'_1$ has at most $m+3$ steps.

STEP 1. In this step, each potential leader i computes and propagates his candidate block B_i^r, together with his own credential, $\sigma_i^{r,1}$.

Recall that this credential *explicitly* identifies i. This is so, because $\sigma_i^{r,1} \triangleq SIG_i(r,1,Q^{r-1})$.

Potential verifier i also propagates, as part of his message, his proper digital signature of $H(B_i^r)$. Not dealing with a payment or a credential, this signature of i is relative to his ephemeral public key $pk_i^{r,1}$: that is, he propagates $sig_{pk_i^{r,1}}(H(B_i^r))$.

Given our conventions, rather than propagating B_i^r and $sig_{pk_i^{r,1}}(H(B_i^r))$, he could have propagated $SIG_{pk_i^{r,1}}(H(B_i^r))$. However, in our analysis we need to have explicit access to $sig_{pk_i^{r,1}}(H(B_i^r))$.

STEPS 2. In this step, each verifier i sets ℓ_i^r to be the potential leader whose hashed credential is the smallest, and B_i^r to be the block proposed by ℓ_i^r. Since, for the sake of efficiency, we wish to agree on $H(B^r)$, rather than directly on B^r, i propagates the message he would have propagated in the first step of $BA^\star$ with initial value $v_i' = H(B_i^r)$. That is, he propagates v_i', after ephemerally signing it, of course. (Namely, after signing it relative to the right ephemeral public key, which in this case is $pk_i^{r,2}$.) Of course too, i also transmits his own credential.

Since the first step of $BA^\star$ consists of the first step of the graded consensus protocol GC, Step 2 of $Algorand'$ corresponds to the first step of GC.

STEPS 3. In this step, each verifier $i \in SV^{r,2}$ executes the second step of $BA^\star$. That is, he sends the same message he would have sent in the second step of GC. Again, i's message is ephemerally signed and accompanied by i's credential. (From now on, we shall omit saying that a verifier ephemerally signs his message and also propagates his credential.)

STEP 4. In this step, every verifier $i \in SV^{r,4}$ computes the output of GC, (v_i, g_i), and ephemerally signs and sends the same message he would have sent in the third step of $BA^\star$, that is, in the first step of $BBA^\star$, with initial bit 0 if $g_i = 2$, and 1 otherwise.

STEP $s = 5, \ldots, m+2$. Such a step, if ever reached, corresponds to step $s-1$ of $BA^\star$, and thus to step $s-3$ of $BBA^\star$.

Since our propagation model is sufficiently asynchronous, we must account for the possibility that, in the middle of such a step s, a verifier $i \in SV^{r,s}$ is reached by information proving him that block B^r has already been chosen. In this case, i stops his own execution of round r of $Algorand'$, and starts executing his round-$(r+1)$ instructions.

$\{r', \ldots, r'+10^6\} \times \{1, \ldots, m+3\}$. Then he orders these public keys in a canonical way, stores the jth public key in the jth leaf of a Merkle tree, and computes the root value R_i, which he publicizes. When he wants to sign a message relative to key $pk_i^{r,s}$, i not only provides the actual signature, but also the authenticating path for $pk_i^{r,s}$ relative to R_i. Notice that this authenticating path also proves that $pk_i^{r,s}$ is stored in the jth leaf. The rest of the details can be easily filled.

Accordingly, the instructions of a verifier $i \in SV^{r,s}$, in addition to the instructions corresponding to Step $s-3$ of $BBA^\star$, include checking whether the execution of $BBA^\star$ has halted in a prior Step s'. Since $BBA^\star$ can only halt is a Coin-Fixed-to-0 Step or in a Coin-Fixed-to-1 step, the instructions distinguish whether

A (Ending Condition 0): $s'-2 \equiv 0 \ mod\ 3$, or

B (Ending Condition 1): $s'-2 \equiv 1 \ mod\ 3$.

In fact, in case A, the block B^r is non-empty, and thus additional instructions are necessary to ensure that i properly reconstructs B^r, together with its proper certificate $CERT^r$. In case B, the block B^r is empty, and thus i is instructed to set $B^r = B^r_\varepsilon = (r, \emptyset, H(Q^{r-1}, r), H(B^{r-1}))$, and to compute $CERT^r$.

If, during his execution of step s, i does not see any evidence that the block B^r has already been generated, then he sends the same message he would have sent in step $s-3$ of $BBA^\star$.

Step $m+3$. If, during step $m+3$, $i \in SV^{r,m+3}$ sees that the block B^r was already generated in a prior step s', then he proceeds just as explained above.

Else, rather then sending the same message he would have sent in step m of $BBA^\star$, i is instructed, based on the information in his possession, to compute B^r and its corresponding certificate $CERT^r$.

Recall, in fact, that we upperbound by $m+3$ the total number of steps of a round.

5.4 The Actual Protocol

Recall that, in each step s of a round r, a verifier $i \in SV^{r,s}$ uses his long-term public-secret key pair to produce his credential, $\sigma_i^{r,s} \triangleq SIG_i(r, s, Q^{r-1})$, as well as $SIG_i\left(Q^{r-1}\right)$ in case $s = 1$. Verifier i uses his ephemeral secret key $sk_i^{r,s}$ to sign his (r,s)-message $m_i^{r,s}$. For simplicity, when r and s are clear, we write $esig_i(x)$ rather than $sig_{pk_i^{r,s}}(x)$ to denote i's proper ephemeral signature of a value x in step s of round r, and write $ESIG_i(x)$ instead of $SIG_{pk_i^{r,s}}(x)$ to denote $(i, x, esig_i(x))$.

Step 1: Block Proposal

Instructions for every user $i \in PK^{r-k}$: User i starts his own Step 1 of round r as soon as he knows B^{r-1}.

- User i computes Q^{r-1} from the third component of B^{r-1} and checks whether $i \in SV^{r,1}$ or not.
- If $i \notin SV^{r,1}$, then i stops his own execution of Step 1 right away.
- If $i \in SV^{r,1}$, that is, if i is a potential leader, then he collects the round-r payments that have been propagated to him so far and computes a maximal payset PAY_i^r from them. Next, he computes his "candidate block" $B_i^r = (r, PAY_i^r, SIG_i(Q^{r-1}), H(B^{r-1}))$. Finally, he computes the message $m_i^{r,1} = (B_i^r, esig_i(H(B_i^r)), \sigma_i^{r,1})$, destroys his ephemeral secret key $sk_i^{r,1}$, and then propagates $m_i^{r,1}$.

Remark. In practice, to shorten the global execution of Step 1, it is important that the $(r,1)$-messages are *selectively propagated*. That is, for every user i in the system, for the first $(r,1)$-message that he ever receives and successfully verifies,[17] player i propagates it as usual. For all the other $(r,1)$-messages that player i receives and successfully verifies, he propagates it only if the hash value of the credential it contains is the *smallest* among the hash values of the credentials contained in all $(r,1)$-messages he has received and successfully verified so far. Furthermore, as suggested by Georgios Vlachos, it is useful that each potential leader i also propagates his credential $\sigma_i^{r,1}$ separately: those small messages travel faster than blocks, ensure timely propagation of the $m_j^{r,1}$'s where the contained credentials have small hash values, while make those with large hash values disappear quickly.

Step 2: The First Step of the Graded Consensus Protocol GC

Instructions for every user $i \in PK^{r-k}$: User i starts his own Step 2 of round r as soon as he knows B^{r-1}.

- User i computes Q^{r-1} from the third component of B^{r-1} and checks whether $i \in SV^{r,2}$ or not.
- If $i \notin SV^{r,2}$ then i stops his own execution of Step 2 right away.
- If $i \in SV^{r,2}$, then after waiting an amount of time $t_2 \triangleq \lambda + \Lambda$, i acts as follows.
  1. He finds the user ℓ such that $H(\sigma_\ell^{r,1}) \leq H(\sigma_j^{r,1})$ for all credentials $\sigma_j^{r,1}$ that are part of the successfully verified $(r,1)$-messages he has received so far.[a]
  2. If he has received from ℓ a valid message $m_\ell^{r,1} = (B_\ell^r, esig_\ell(H(B_\ell^r)), \sigma_\ell^{r,1})$,[b] then i sets $v_i' \triangleq H(B_\ell^r)$; otherwise i sets $v_i' \triangleq \perp$.
  3. i computes the message $m_i^{r,2} \triangleq (ESIG_i(v_i'), \sigma_i^{r,2})$,[c] destroys his ephemeral secret key $sk_i^{r,2}$, and then propagates $m_i^{r,2}$.

[a] Essentially, user i privately decides that the leader of round r is user ℓ.

[b] Again, player ℓ's signatures and the hashes are all successfully verified, and PAY_ℓ^r in B_ℓ^r is a valid payset for round r —although i does not check whether PAY_ℓ^r is maximal for ℓ or not.

[c] The message $m_i^{r,2}$ signals that player i considers v_i' to be the hash of the next block, or considers the next block to be empty.

[17] That is, all the signatures are correct and both the block and its hash are valid —although i does not check whether the included payset is maximal for its proposer or not.

Step 3: The Second Step of GC

Instructions for every user $i \in PK^{r-k}$: User i starts his own Step 3 of round r as soon as he knows B^{r-1}.

- User i computes Q^{r-1} from the third component of B^{r-1} and checks whether $i \in SV^{r,3}$ or not.
- If $i \notin SV^{r,3}$, then i stops his own execution of Step 3 right away.
- If $i \in SV^{r,3}$, then after waiting an amount of time $t_3 \triangleq t_2 + 2\lambda = 3\lambda + \Lambda$, i acts as follows.
  1. If there exists a value $v' \neq \perp$ such that, among all the valid messages $m_j^{r,2}$ he has received, more than 2/3 of them are of the form $(ESIG_j(v'), \sigma_j^{r,2})$, without any contradiction,[a] then he computes the message $m_i^{r,3} \triangleq (ESIG_i(v'), \sigma_i^{r,3})$. Otherwise, he computes $m_i^{r,3} \triangleq (ESIG_i(\perp), \sigma_i^{r,3})$.
  2. i destroys his ephemeral secret key $sk_i^{r,3}$, and then propagates $m_i^{r,3}$.

[a] That is, he has not received two valid messages containing $ESIG_j(v')$ and a different $ESIG_j(v'')$ respectively, from a player j. Here and from here on, except in the Ending Conditions defined later, whenever an honest player wants messages of a given form, messages contradicting each other are never counted or considered valid.

Step 4: Output of GC and The First Step of $BBA^\star$

Instructions for every user $i \in PK^{r-k}$: User i starts his own Step 4 of round r as soon as he knows B^{r-1}.

- User i computes Q^{r-1} from the third component of B^{r-1} and checks whether $i \in SV^{r,4}$ or not.
- If $i \notin SV^{r,4}$, then i his stops his own execution of Step 4 right away.
- If $i \in SV^{r,4}$, then after waiting an amount of time $t_4 \triangleq t_3 + 2\lambda = 5\lambda + \Lambda$, i acts as follows.
  1. He computes v_i and g_i, the output of GC, as follows.
     (a) If there exists a value $v' \neq \perp$ such that, among all the valid messages $m_j^{r,3}$ he has received, more than 2/3 of them are of the form $(ESIG_j(v'), \sigma_j^{r,3})$, then he sets $v_i \triangleq v'$ and $g_i \triangleq 2$.
     (b) Otherwise, if there exists a value $v' \neq \perp$ such that, among all the valid messages $m_j^{r,3}$ he has received, more than 1/3 of them are of the form $(ESIG_j(v'), \sigma_j^{r,3})$, then he sets $v_i \triangleq v'$ and $g_i \triangleq 1$.[a]
     (c) Else, he sets $v_i \triangleq H(B_\epsilon^r)$ and $g_i \triangleq 0$.
  2. He computes b_i, the input of $BBA^\star$, as follows:
     $b_i \triangleq 0$ if $g_i = 2$, and $b_i \triangleq 1$ otherwise.
  3. He computes the message $m_i^{r,4} \triangleq (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,4})$, destroys his ephemeral secret key $sk_i^{r,4}$, and then propagates $m_i^{r,4}$.

[a] It can be proved that the v' in case (b), if exists, must be unique.

Step s, $5 \leq s \leq m+2$, $s-2 \equiv 0 \mod 3$: A Coin-Fixed-To-0 Step of $BBA^\star$

Instructions for every user $i \in PK^{r-k}$: User i starts his own Step s of round r as soon as he knows B^{r-1}.

- User i computes Q^{r-1} from the third component of B^{r-1} and checks whether $i \in SV^{r,s}$.
- If $i \notin SV^{r,s}$, then i stops his own execution of Step s right away.
- If $i \in SV^{r,s}$ then he acts as follows.
  - He waits until an amount of time $t_s \triangleq t_{s-1} + 2\lambda = (2s-3)\lambda + \Lambda$ has passed.
  - *Ending Condition 0:* If, during such waiting and at any point of time, there exists a string $v \neq \perp$ and a step s' such that
    (a) $5 \leq s' \leq s$, $s'-2 \equiv 0 \mod 3$ —that is, Step s' is a Coin-Fixed-To-0 step,
    (b) i has received at least $t_H = \frac{2n}{3} + 1$ valid messages $m_j^{r,s'-1} = (ESIG_j(0), ESIG_j(v), \sigma_j^{r,s'-1})$,[a] and
    (c) i has received a valid message $m_j^{r,1} = (B_j^r, esig_j(H(B_j^r)), \sigma_j^{r,1})$ with $v = H(B_j^r)$,
    then, i stops his own execution of Step s (and in fact of round r) right away without propagating anything; sets $B^r = B_j^r$; and sets his own $CERT^r$ to be the set of messages $m_j^{r,s'-1}$ of sub-step (b).[b]
  - *Ending Condition 1:* If, during such waiting and at any point of time, there exists a step s' such that
    (a') $6 \leq s' \leq s$, $s'-2 \equiv 1 \mod 3$ —that is, Step s' is a Coin-Fixed-To-1 step, and
    (b') i has received at least t_H valid messages $m_j^{r,s'-1} = (ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s'-1})$,[c]
    then, i stops his own execution of Step s (and in fact of round r) right away without propagating anything; sets $B^r = B_\epsilon^r$; and sets his own $CERT^r$ to be the set of messages $m_j^{r,s'-1}$ of sub-step (b').
  - Otherwise, at the end of the wait, user i does the following.
    He sets v_i to be the majority vote of the v_j's in the second components of all the valid $m_j^{r,s-1}$'s he has received.
    He computes b_i as follows.
    If more than 2/3 of all the valid $m_j^{r,s-1}$'s he has received are of the form $(ESIG_j(0), ESIG_j(v_j), \sigma_j^{r,s-1})$, then he sets $b_i \triangleq 0$.
    Else, if more than 2/3 of all the valid $m_j^{r,s-1}$'s he has received are of the form $(ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s-1})$, then he sets $b_i \triangleq 1$.
    Else, he sets $b_i \triangleq 0$.
    He computes the message $m_i^{r,s} \triangleq (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$, destroys his ephemeral secret key $sk_i^{r,s}$, and then propagates $m_i^{r,s}$.

[a] Such a message from player j is counted even if player i has also received a message from j signing for 1. Similar things for Ending Condition 1. As shown in the analysis, this is done to ensure that all honest users know B^r within time λ from each other.

[b] User i now knows B^r and his own round r finishes. He still helps propagating messages as a generic user, but does not initiate any propagation as a (r,s)-verifier. In particular, he has helped propagating all messages in his $CERT^r$, which is enough for our protocol. Note that he should also set $b_i \triangleq 0$ for the binary BA protocol, but b_i is not needed in this case anyway. Similar things for all future instructions.

[c] In this case, it does not matter what the v_j's are.

Step s, $6 \leq s \leq m+2$, $s-2 \equiv 1 \mod 3$: A Coin-Fixed-To-1 Step of $BBA^{\star}$

Instructions for every user $i \in PK^{r-k}$: User i starts his own Step s of round r as soon as he knows B^{r-1}.

- User i computes Q^{r-1} from the third component of B^{r-1} and checks whether $i \in SV^{r,s}$ or not.
- If $i \notin SV^{r,s}$, then i stops his own execution of Step s right away.
- If $i \in SV^{r,s}$ then he does the follows.
  - He waits until an amount of time $t_s \triangleq (2s-3)\lambda + \Lambda$ has passed.
  - *Ending Condition 0:* The same instructions as Coin-Fixed-To-0 steps.
  - *Ending Condition 1:* The same instructions as Coin-Fixed-To-0 steps.
  - Otherwise, at the end of the wait, user i does the following.

    He sets v_i to be the majority vote of the v_j's in the second components of all the valid $m_j^{r,s-1}$'s he has received.

    He computes b_i as follows.

    If more than 2/3 of all the valid $m_j^{r,s-1}$'s he has received are of the form $(ESIG_j(0), ESIG_j(v_j), \sigma_j^{r,s-1})$, then he sets $b_i \triangleq 0$.

    Else, if more than 2/3 of all the valid $m_j^{r,s-1}$'s he has received are of the form $(ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s-1})$, then he sets $b_i \triangleq 1$.

    Else, he sets $b_i \triangleq 1$.

    He computes the message $m_i^{r,s} \triangleq (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$, destroys his ephemeral secret key $sk_i^{r,s}$, and then propagates $m_i^{r,s}$.

Step s, $7 \leq s \leq m+2$, $s-2 \equiv 2 \mod 3$: A Coin-Genuinely-Flipped Step of $BBA^\star$

Instructions for every user $i \in PK^{r-k}$: User i starts his own Step s of round r as soon as he knows B^{r-1}.

- User i computes Q^{r-1} from the third component of B^{r-1} and checks whether $i \in SV^{r,s}$ or not.
- If $i \notin SV^{r,s}$, then i stops his own execution of Step s right away.
- If $i \in SV^{r,s}$ then he does the follows.
  - He waits until an amount of time $t_s \triangleq (2s-3)\lambda + \Lambda$ has passed.
  - *Ending Condition 0:* The same instructions as Coin-Fixed-To-0 steps.
  - *Ending Condition 1:* The same instructions as Coin-Fixed-To-0 steps.
  - Otherwise, at the end of the wait, user i does the following.

    He sets v_i to be the majority vote of the v_j's in the second components of all the valid $m_j^{r,s-1}$'s he has received.

    He computes b_i as follows.

    If more than 2/3 of all the valid $m_j^{r,s-1}$'s he has received are of the form $(ESIG_j(0), ESIG_j(v_j), \sigma_j^{r,s-1})$, then he sets $b_i \triangleq 0$.

    Else, if more than 2/3 of all the valid $m_j^{r,s-1}$'s he has received are of the form $(ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s-1})$, then he sets $b_i \triangleq 1$.

    Else, let $SV_i^{r,s-1}$ be the set of $(r, s-1)$-verifiers from whom he has received a valid message $m_j^{r,s-1}$. He sets $b_i \triangleq \mathtt{lsb}(\min_{j \in SV_i^{r,s-1}} H(\sigma_j^{r,s-1}))$.

    He computes the message $m_i^{r,s} \triangleq (ESIG_i(b_i), ESIG_i(v_i), \sigma_i^{r,s})$, destroys his ephemeral secret key $sk_i^{r,s}$, and then propagates $m_i^{r,s}$.

Step $m+3$: The Last Step of $BBA^\star$ [a]

Instructions for every user $i \in PK^{r-k}$: User i starts his own Step $m+3$ of round r as soon as he knows B^{r-1}.

- User i computes Q^{r-1} from the third component of B^{r-1} and checks whether $i \in SV^{r,m+3}$ or not.
- If $i \notin SV^{r,m+3}$, then i stops his own execution of Step $m+3$ right away.
- If $i \in SV^{r,m+3}$ then he does the follows.
  - He waits until an amount of time $t_{m+3} \triangleq t_{m+2} + 2\lambda = (2m+3)\lambda + \Lambda$ has passed.
  - *Ending Condition 0:* The same instructions as Coin-Fixed-To-0 steps.
  - *Ending Condition 1:* The same instructions as Coin-Fixed-To-0 steps.
  - Otherwise, at the end of the wait, user i does the following.
    He sets $out_i \triangleq 1$ and $B^r \triangleq B^r_\epsilon$.
    He computes the message $m_i^{r,m+3} = (ESIG_i(out_i), ESIG_i(H(B^r)), \sigma_i^{r,m+3})$, destroys his ephemeral secret key $sk_i^{r,m+3}$, and then propagates $m_i^{r,m+3}$ to certify B^r.[b]

[a] With overwhelming probability $BBA^\star$ has ended before this step, and we specify this step for completeness.

[b] A certificate from Step $m+3$ does not have to include $ESIG_i(out_i)$. We include it for uniformity only: the certificates now have a uniform format no matter in which step they are generated.

Reconstruction of the Round-r Block by Non-Verifiers

Instructions for every user i in the system: User i starts his own round r as soon as he knows B^{r-1}, and waits for block information as follows.

- If, during such waiting and at any point of time, there exists a string v and a step s' such that
  (a) $5 \leq s' \leq m+3$ with $s'-2 \equiv 0 \mod 3$,
  (b) i has received at least t_H valid messages $m_j^{r,s'-1} = (ESIG_j(0), ESIG_j(v), \sigma_j^{r,s'-1})$, and
  (c) i has received a valid message $m_j^{r,1} = (B_j^r, esig_j(H(B_j^r)), \sigma_j^{r,1})$ with $v = H(B_j^r)$,
  then, i stops his own execution of round r right away; sets $B^r = B_j^r$; and sets his own $CERT^r$ to be the set of messages $m_j^{r,s'-1}$ of sub-step (b).
- If, during such waiting and at any point of time, there exists a step s' such that
  (a') $6 \leq s' \leq m+3$ with $s'-2 \equiv 1 \mod 3$, and
  (b') i has received at least t_H valid messages $m_j^{r,s'-1} = (ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s'-1})$,
  then, i stops his own execution of round r right away; sets $B^r = B_\epsilon^r$; and sets his own $CERT^r$ to be the set of messages $m_j^{r,s'-1}$ of sub-step (b').
- If, during such waiting and at any point of time, i has received at least t_H valid messages $m_j^{r,m+3} = (ESIG_j(1), ESIG_j(H(B_\epsilon^r)), \sigma_j^{r,m+3})$, then i stops his own execution of round r right away, sets $B^r = B_\epsilon^r$, and sets his own $CERT^r$ to be the set of messages $m_j^{r,m+3}$ for 1 and $H(B_\epsilon^r)$.

5.5 Analysis of *Algorand*$'_1$

We introduce the following notations for each round $r \geq 0$, used in the analysis.

- Let T^r be the time when the first honest user knows B^{r-1}.
- Let I^{r+1} be the interval $[T^{r+1}, T^{r+1} + \lambda]$.

Note that $T^0 = 0$ by the initialization of the protocol. For each $s \geq 1$ and $i \in SV^{r,s}$, recall that $\alpha_i^{r,s}$ and $\beta_i^{r,s}$ are respectively the starting time and the ending time of player i's step s. Moreover, recall that $t_s = (2s-3)\lambda + \Lambda$ for each $2 \leq s \leq m+3$. In addition, let $I^0 \triangleq \{0\}$ and $t_1 \triangleq 0$.

Finally, recall that $L^r \leq m/3$ is a random variable representing the number of Bernoulli trials needed to see a 1, when each trial is 1 with probability $\frac{p_h}{2}$ and there are at most $m/3$ trials. If all trials fail then $L^r \triangleq m/3$.

In the analysis we ignore computation time, as it is in fact negligible relative to the time needed to propagate messages. In any case, by using slightly larger λ and Λ, the computation time can be incorporated into the analysis directly. Most of the statements below hold "with overwhelming probability," and we may not repeatedly emphasize this fact in the analysis.

5.6 Main Theorem

Theorem 5.1. *The following properties hold with overwhelming probability for each round $r \geq 0$:*

1. *All honest users agree on the same block B^r.*
2. *When the leader ℓ^r is honest, the block B^r is generated by ℓ^r, B^r contains a maximal payset received by ℓ^r by time $\alpha_{\ell^r}^{r,1}$, $T^{r+1} \leq T^r + 8\lambda + \Lambda$ and all honest users know B^r in the time interval I^{r+1}.*
3. *When the leader ℓ^r is malicious, $T^{r+1} \leq T^r + (6L^r + 10)\lambda + \Lambda$ and all honest users know B^r in the time interval I^{r+1}.*
4. *$p_h = h^2(1 + h - h^2)$ for L^r, and the leader ℓ^r is honest with probability at least p_h.*

Before proving our main theorem, let us make two remarks.

Remarks.

- *Block-Generation and True Latency.* The time to generate block B^r is defined to be $T^{r+1} - T^r$. That is, it is defined to be the difference between the first time some honest user learns B^r and the first time some honest user learns B^{r-1}. When the round-r leader is honest, Property 2 our main theorem guarantees that the *exact* time to generate B^r is $8\lambda + \Lambda$ time, no matter what the precise value of $h > 2/3$ may be. When the leader is malicious, Property 3 implies that the *expected* time to generate B^r is upperbounded by $(\frac{12}{p_h} + 10)\lambda + \Lambda$, again no matter the precise value of h.[18] However, the expected time to generate B^r depends on the precise value of h. Indeed, by Property 4, $p_h = h^2(1 + h - h^2)$ and the leader is honest with probability at least p_h, thus

  $$\mathbb{E}[T^{r+1} - T^r] \leq h^2(1 + h - h^2) \cdot (8\lambda + \Lambda) + (1 - h^2(1 + h - h^2))((\frac{12}{h^2(1 + h - h^2)} + 10)\lambda + \Lambda).$$

  For instance, if $h = 80\%$, then $\mathbb{E}[T^{r+1} - T^r] \leq 12.7\lambda + \Lambda$.

- *λ vs. Λ.* Note that the size of the messages sent by the verifiers in a step *Algorand′* is dominated by the length of the digital signature keys, which can remain fixed, even when the number of users is enormous. Also note that, in any step $s > 1$, the same expected number n of verifiers can be used whether the number of users is 100K, 100M, or 100M. This is so because n solely depends on h and F. In sum, therefore, barring a sudden need to increase secret key length, the value of λ should remain the same no matter how large the number of users may be in the foreseeable future.

  By contrast, for any transaction rate, the number of transactions grows with the number of users. Therefore, to process all new transactions in a timely fashion, the size of a block should also grow with the number of users, causing Λ to grow too. Thus, in the long run, we should have $\lambda << \Lambda$. Accordingly, it is proper to have a larger coefficient for λ, and actually a coefficient of 1 for Λ.

Proof of Theorem 5.1. We prove Properties 1–3 by induction: assuming they hold for round $r - 1$ (without loss of generality, they automatically hold for "round -1" when $r = 0$), we prove them for round r.

[18] Indeed, $\mathbb{E}[T^{r+1} - T^r] \leq (6\mathbb{E}[L^r] + 10)\lambda + \Lambda = (6 \cdot \frac{2}{p_h} + 10)\lambda + \Lambda = (\frac{12}{p_h} + 10)\lambda + \Lambda$.

Since B^{r-1} is uniquely defined by the inductive hypothesis, the set $SV^{r,s}$ is uniquely defined for each step s of round r. By the choice of n_1, $SV^{r,1} \neq \emptyset$ with overwhelming probability. We now state the following two lemmas, proved in Sections 5.7 and 5.8. Throughout the induction and in the proofs of the two lemmas, the analysis for round 0 is almost the same as the inductive step, and we will highlight the differences when they occur.

Lemma 5.2. *[Completeness Lemma] Assuming Properties 1–3 hold for round $r-1$, when the leader ℓ^r is honest, with overwhelming probability,*

- *All honest users agree on the same block B^r, which is generated by ℓ^r and contains a maximal payset received by ℓ^r by time $\alpha_{\ell^r}^{r,1} \in I^r$; and*
- *$T^{r+1} \leq T^r + 8\lambda + \Lambda$ and all honest users know B^r in the time interval I^{r+1}.*

Lemma 5.3. *[Soundness Lemma] Assuming Properties 1–3 hold for round $r-1$, when the leader ℓ^r is malicious, with overwhelming probability, all honest users agree on the same block B^r, $T^{r+1} \leq T^r + (6L^r + 10)\lambda + \Lambda$ and all honest users know B^r in the time interval I^{r+1}.*

Properties 1–3 hold by applying Lemmas 5.2 and 5.3 to $r = 0$ and to the inductive step. Finally, we restate Property 4 as the following lemma, proved in Section 5.9.

Lemma 5.4. *Given Properties 1–3 for each round before r, $p_h = h^2(1 + h - h^2)$ for L^r, and the leader ℓ^r is honest with probability at least p_h.*

Combining the above three lemmas together, Theorem 5.1 holds. ■

The lemma below states several important properties about round r given the inductive hypothesis, and will be used in the proofs of the above three lemmas.

Lemma 5.5. *Assume Properties 1–3 hold for round $r-1$. For each step $s \geq 1$ of round r and each honest verifier $i \in HSV^{r,s}$, we have that*

(a) $\alpha_i^{r,s} \in I^r$;

(b) if player i has waited an amount of time t_s, then $\beta_i^{r,s} \in [T^r + t_s, T^r + \lambda + t_s]$ for $r > 0$ and $\beta_i^{r,s} = t_s$ for $r = 0$; and

(c) if player i has waited an amount of time t_s, then by time $\beta_i^{r,s}$, he has received all messages sent by all honest verifiers $j \in HSV^{r,s'}$ for all steps $s' < s$.

Moreover, for each step $s \geq 3$, we have that

(d) there do not exist two different players $i, i' \in SV^{r,s}$ and two different values v, v' of the same length, such that both players have waited an amount of time t_s, more than 2/3 of all the valid messages $m_j^{r,s-1}$ player i receives have signed for v, and more than 2/3 of all the valid messages $m_j^{r,s-1}$ player i' receives have signed for v'.

Proof. Property (a) follows directly from the inductive hypothesis, as player i knows B^{r-1} in the time interval I^r and starts his own step s right away. Property (b) follows directly from (a): since player i has waited an amount of time t_s before acting, $\beta_i^{r,s} = \alpha_i^{r,s} + t_s$. Note that $\alpha_i^{r,s} = 0$ for $r = 0$.

We now prove Property (c). If $s = 2$, then by Property (b), for all verifiers $j \in HSV^{r,1}$ we have

$$\beta_i^{r,s} = \alpha_i^{r,s} + t_s \geq T^r + t_s = T^r + \lambda + \Lambda \geq \beta_j^{r,1} + \Lambda.$$

Since each verifier $j \in HSV^{r,1}$ sends his message at time $\beta_j^{r,1}$ and the message reaches all honest users in at most Λ time, by time $\beta_i^{r,s}$ player i has received the messages sent by all verifiers in $HSV^{r,1}$ as desired.

If $s > 2$, then $t_s = t_{s-1} + 2\lambda$. By Property (b), for all steps $s' < s$ and all verifiers $j \in HSV^{r,s'}$,

$$\beta_i^{r,s} = \alpha_i^{r,s} + t_s \geq T^r + t_s = T^r + t_{s-1} + 2\lambda \geq T^r + t_{s'} + 2\lambda = T^r + \lambda + t_{s'} + \lambda \geq \beta_j^{r,s'} + \lambda.$$

Since each verifier $j \in HSV^{r,s'}$ sends his message at time $\beta_j^{r,s'}$ and the message reaches all honest users in at most λ time, by time $\beta_i^{r,s}$ player i has received all messages sent by all honest verifiers in $HSV^{r,s'}$ for all $s' < s$. Thus Property (c) holds.

Finally, we prove Property (d). Note that the verifiers $j \in SV^{r,s-1}$ sign at most two things in Step $s-1$ using their ephemeral secret keys: a value v_j of the same length as the output of the hash function, and also a bit $b_j \in \{0,1\}$ if $s - 1 \geq 4$. That is why in the statement of the lemma we require that v and v' have the same length: many verifiers may have signed both a hash value v and a bit b, thus both pass the 2/3 threshold.

Assume for the sake of contradiction that there exist the desired verifiers i, i' and values v, v'. Note that some malicious verifiers in $MSV^{r,s-1}$ may have signed both v and v', but each honest verifier in $HSV^{r,s-1}$ has signed at most one of them. By Property (c), both i and i' have received all messages sent by all honest verifiers in $HSV^{r,s-1}$.

Let $HSV^{r,s-1}(v)$ be the set of honest $(r, s-1)$-verifiers who have signed v, $MSV_i^{r,s-1}$ the set of malicious $(r, s-1)$-verifiers from whom i has received a valid message, and $MSV_i^{r,s-1}(v)$ the subset of $MSV_i^{r,s-1}$ from whom i has received a valid message signing v. By the requirements for i and v, we have

$$ratio \triangleq \frac{|HSV^{r,s-1}(v)| + |MSV_i^{r,s-1}(v)|}{|HSV^{r,s-1}| + |MSV_i^{r,s-1}|} > \frac{2}{3}. \tag{1}$$

We first show

$$|MSV_i^{r,s-1}(v)| \leq |HSV^{r,s-1}(v)|. \tag{2}$$

Assuming otherwise, by the relationships among the parameters, with overwhelming probability $|HSV^{r,s-1}| > 2|MSV^{r,s-1}| \geq 2|MSV_i^{r,s-1}|$, thus

$$ratio < \frac{|HSV^{r,s-1}(v)| + |MSV_i^{r,s-1}(v)|}{3|MSV_i^{r,s-1}|} < \frac{2|MSV_i^{r,s-1}(v)|}{3|MSV_i^{r,s-1}|} \leq \frac{2}{3},$$

contradicting Inequality 1.

Next, by Inequality 1 we have

$$\begin{aligned} & 2|HSV^{r,s-1}| + 2|MSV_i^{r,s-1}| < 3|HSV^{r,s-1}(v)| + 3|MSV_i^{r,s-1}(v)| \\ \leq\ & 3|HSV^{r,s-1}(v)| + 2|MSV_i^{r,s-1}| + |MSV_i^{r,s-1}(v)|. \end{aligned}$$

Combining with Inequality 2,

$$2|HSV^{r,s-1}| < 3|HSV^{r,s-1}(v)| + |MSV_i^{r,s-1}(v)| \leq 4|HSV^{r,s-1}(v)|,$$

which implies

$$|HSV^{r,s-1}(v)| > \frac{1}{2}|HSV^{r,s-1}|.$$

Similarly, by the requirements for i' and v', we have

$$|HSV^{r,s-1}(v')| > \frac{1}{2}|HSV^{r,s-1}|.$$

Since an honest verifier $j \in HSV^{r,s-1}$ destroys his ephemeral secret key $sk_j^{r,s-1}$ before propagating his message, the Adversary cannot forge j's signature for a value that j did not sign, after learning that j is a verifier. Thus, the two inequalities above imply $|HSV^{r,s-1}| \geq |HSV^{r,s-1}(v)| + |HSV^{r,s-1}(v')| > |HSV^{r,s-1}|$, a contradiction. Accordingly, the desired i, i', v, v' do not exist, and Property (d) holds. ■

5.7 The Completeness Lemma

Lemma 5.2. [Completeness Lemma, restated] *Assuming Properties 1–3 hold for round $r-1$, when the leader ℓ^r is honest, with overwhelming probability,*

- *All honest users agree on the same block B^r, which is generated by ℓ^r and contains a maximal payset received by ℓ^r by time $\alpha_{\ell^r}^{r,1} \in I^r$; and*
- *$T^{r+1} \leq T^r + 8\lambda + \Lambda$ and all honest users know B^r in the time interval I^{r+1}.*

Proof. By the inductive hypothesis and Lemma 5.5, for each step s and verifier $i \in HSV^{r,s}$, $\alpha_i^{r,s} \in I^r$. Below we analyze the protocol step by step.

Step 1. By definition, every honest verifier $i \in HSV^{r,1}$ propagates the desired message $m_i^{r,1}$ at time $\beta_i^{r,1} = \alpha_i^{r,1}$, where $m_i^{r,1} = (B_i^r, esig_i(H(B_i^r)), \sigma_i^{r,1})$, $B_i^r = (r, PAY_i^r, SIG_i(Q^{r-1}), H(B^{r-1}))$, and PAY_i^r is a maximal payset among all payments that i has seen by time $\alpha_i^{r,1}$.

Step 2. Arbitrarily fix an honest verifier $i \in HSV^{r,2}$. By Lemma 5.5, when player i is done waiting at time $\beta_i^{r,2} = \alpha_i^{r,2} + t_2$, he has received all messages sent by verifiers in $HSV^{r,1}$, including $m_{\ell^r}^{r,1}$. By the definition of ℓ^r, there does not exist another player in PK^{r-k} whose credential's hash value is smaller than $H(\sigma_{\ell^r}^{r,1})$. Of course, the Adversary can corrupt ℓ^r after seeing that $H(\sigma_{\ell^r}^{r,1})$ is very small, but by that time player ℓ^r has destroyed his ephemeral key and the message $m_{\ell^r}^{r,1}$ has been propagated. Thus verifier i sets his own leader to be player ℓ^r. Accordingly, at time $\beta_i^{r,2}$, verifier i propagates $m_i^{r,2} = (ESIG_i(v_i'), \sigma_i^{r,2})$, where $v_i' = H(B_{\ell^r}^r)$. When $r = 0$, the only difference is that $\beta_i^{r,2} = t_2$ rather than being in a range. Similar things can be said for future steps and we will not emphasize them again.

Step 3. Arbitrarily fix an honest verifier $i \in HSV^{r,3}$. By Lemma 5.5, when player i is done waiting at time $\beta_i^{r,3} = \alpha_i^{r,3} + t_3$, he has received all messages sent by verifiers in $HSV^{r,2}$.

By the relationships among the parameters, with overwhelming probability $|HSV^{r,2}| > 2|MSV^{r,2}|$. Moreover, no honest verifier would sign contradicting messages, and the Adversary cannot forge a signature of an honest verifier after the latter has destroyed his corresponding ephemeral secret key. Thus more than 2/3 of all the valid $(r,2)$-messages i has received are from honest verifiers and of the form $m_j^{r,2} = (ESIG_j(H(B_{\ell^r}^r)), \sigma_j^{r,2})$, with no contradiction.

Accordingly, at time $\beta_i^{r,3}$ player i propagates $m_i^{r,3} = (ESIG_i(v'), \sigma_i^{r,3})$, where $v' = H(B_{\ell^r}^r)$.

Step 4. Arbitrarily fix an honest verifier $i \in HSV^{r,4}$. By Lemma 5.5, player i has received all messages sent by verifiers in $HSV^{r,3}$ when he is done waiting at time $\beta_i^{r,4} = \alpha_i^{r,4} + t_4$. Similar to Step 3, more than 2/3 of all the valid $(r,3)$-messages i has received are from honest verifiers and of the form $m_j^{r,3} = (ESIG_j(H(B_{\ell^r}^r)), \sigma_j^{r,3})$.

Accordingly, player i sets $v_i = H(B_{\ell^r}^r)$, $g_i = 2$ and $b_i = 0$. At time $\beta_i^{r,4} = \alpha_i^{r,4} + t_4$ he propagates $m_i^{r,4} = (ESIG_i(0), ESIG_i(H(B_{\ell^r}^r)), \sigma_i^{r,4})$.

Step 5. Arbitrarily fix an honest verifier $i \in HSV^{r,5}$. By Lemma 5.5, player i would have received all messages sent by the verifiers in $HSV^{r,4}$ if he has waited till time $\alpha_i^{r,5} + t_5$. Note that $|HSV^{r,4}| \geq t_H$.[19] Also note that all verifiers in $HSV^{r,4}$ have signed for $H(B_{\ell^r}^r)$.

As $|MSV^{r,4}| < t_H$, there does not exist any $v' \neq H(B_{\ell^r}^r)$ that could have been signed by t_H verifiers in $SV^{r,4}$ (who would necessarily be malicious), so player i does not stop before he has received t_H valid messages $m_j^{r,4} = (ESIG_j(0), ESIG_j(H(B_{\ell^r}^r)), \sigma_j^{r,4})$. Let T be the time when the latter event happens. Some of those messages may be from malicious players, but because $|MSV^{r,4}| < t_H$, at least one of them is from an honest verifier in $HSV^{r,4}$ and is sent after time $T^r + t_4$. Accordingly, $T \geq T^r + t_4 > T^r + \lambda + \Lambda \geq \beta_{\ell^r}^{r,1} + \Lambda$, and by time T player i has also received the message $m_{\ell^r}^{r,1}$. By the construction of the protocol, player i stops at time $\beta_i^{r,5} = T$ without propagating anything; sets $B^r = B_{\ell^r}^r$; and sets his own $CERT^r$ to be the set of $(r,4)$-messages for 0 and $H(B_{\ell^r}^r)$ that he has received.

Step $s > 5$. Similarly, for any step $s > 5$ and any verifier $i \in HSV^{r,s}$, player i would have received all messages sent by the verifiers in $HSV^{r,4}$ if he has waited till time $\alpha_i^{r,s} + t_s$. By the same analysis, player i stops without propagating anything, setting $B^r = B_{\ell^r}^r$ (and setting his own $CERT^r$ properly). Of course, the malicious verifiers may not stop and may propagate arbitrary messages, but because $|MSV^{r,s}| < t_H$, by induction no other v' could be signed by t_H verifiers in any step $4 \leq s' < s$, thus the honest verifiers only stop because they have received t_H valid $(r,4)$-messages for 0 and $H(B_{\ell^r}^r)$.

Reconstruction of the Round-r Block. The analysis of Step 5 applies to a generic honest user i almost without any change. Indeed, player i starts his own round r in the interval I^r and will only stop at a time T when he has received t_H valid $(r,4)$-messages for $H(B_{\ell^r}^r)$. Again because at least one of those messages are from honest verifiers and are sent after time $T^r + t_4$, player i has also received $m_{\ell^r}^{r,1}$ by time T. Thus he sets $B^r = B_{\ell^r}^r$ with the proper $CERT^r$.

It only remains to show that all honest users finish their round r within the time interval I^{r+1}. By the analysis of Step 5, every honest verifier $i \in HSV^{r,5}$ knows B^r on or before $\alpha_i^{r,5} + t_5 \leq T^r + \lambda + t_5 = T^r + 8\lambda + \Lambda$. Since T^{r+1} is the time when the first honest user i^r knows B^r, we have

$$T^{r+1} \leq T^r + 8\lambda + \Lambda$$

as desired. Moreover, when player i^r knows B^r, he has already helped propagating the messages in his $CERT^r$. Note that all those messages will be received by all honest users within time λ, even if

[19] Strictly speaking, this happens with very high probability but not necessarily overwhelming. However, this probability slightly effects the running time of the protocol, but does not affect its correctness. When $h = 80\%$, then $|HSV^{r,4}| \geq t_H$ with probability $1 - 10^{-8}$. If this event does not occur, then the protocol will continue for another 3 steps. As the probability that this does not occur in two steps is negligible, the protocol will finish at Step 8. In expectation, then, the number of steps needed is almost 5.

player i^r were the first player to propagate them. Moreover, following the analysis above we have $T^{r+1} \geq T^r + t_4 \geq \beta_{\ell^r}^{r,1} + \Lambda$, thus all honest users have received $m_{\ell^r}^{r,1}$ by time $T^{r+1} + \lambda$. Accordingly, all honest users know B^r in the time interval $I^{r+1} = [T^{r+1}, T^{r+1} + \lambda]$.

Finally, for $r = 0$ we actually have $T^1 \leq t_4 + \lambda = 6\lambda + \Lambda$. Combining everything together, Lemma 5.2 holds. ■

5.8 The Soundness Lemma

Lemma 5.3. [Soundness Lemma, restated] *Assuming Properties 1–3 hold for round $r-1$, when the leader ℓ^r is malicious, with overwhelming probability, all honest users agree on the same block B^r, $T^{r+1} \leq T^r + (6L^r + 10)\lambda + \Lambda$ and all honest users know B^r in the time interval I^{r+1}.*

Proof. We consider the two parts of the protocol, GC and $BBA^\star$, separately.

GC. By the inductive hypothesis and by Lemma 5.5, for any step $s \in \{2, 3, 4\}$ and any honest verifier $i \in HSV^{r,s}$, when player i acts at time $\beta_i^{r,s} = \alpha_i^{r,s} + t_s$, he has received all messages sent by all the honest verifiers in steps $s' < s$. We distinguish two possible cases for step 4.

Case 1. *No verifier $i \in HSV^{r,4}$ sets $g_i = 2$.*

In this case, by definition $b_i = 1$ for all verifiers $i \in HSV^{r,4}$. That is, they start with an agreement on 1 in the binary BA protocol. They may not have an agreement on their v_i's, but this does not matter as we will see in the binary BA.

Case 2. *There exists a verifier $\hat{i} \in HSV^{r,4}$ such that $g_{\hat{i}} = 2$.*

In this case, we show that

(1) $g_i \geq 1$ for all $i \in HSV^{r,4}$,

(2) there exists a value v' such that $v_i = v'$ for all $i \in HSV^{r,4}$, and

(3) there exists a valid message $m_\ell^{r,1}$ from some verifier $\ell \in SV^{r,1}$ such that $v' = H(B_\ell^r)$.

Indeed, since player $\hat{i}$ is honest and sets $g_{\hat{i}} = 2$, more than 2/3 of all the valid messages $m_j^{r,3}$ he has received are for the same value $v' \neq \perp$, and he has set $v_{\hat{i}} = v'$.

By Property (d) in Lemma 5.5, for any other honest $(r, 4)$-verifier i, it cannot be that more than 2/3 of all the valid messages $m_j^{r,3}$ that i' has received are for the same value $v'' \neq v'$. Accordingly, if i sets $g_i = 2$, it must be that i has seen $> 2/3$ majority for v' as well and set $v_i = v'$, as desired.

Now consider an arbitrary verifier $i \in HSV^{r,4}$ with $g_i < 2$. Similar to the analysis of Property (d) in Lemma 5.5, because player $\hat{i}$ has seen $> 2/3$ majority for v', more than $\frac{1}{2}|HSV^{r,3}|$ honest $(r, 3)$-verifiers have signed v'. Because i has received all messages by honest $(r, 3)$-verifiers by time $\beta_i^{r,4} = \alpha_i^{r,4} + t_4$, he has in particular received more than $\frac{1}{2}|HSV^{r,3}|$ messages from them for v'. Because $|HSV^{r,3}| > 2|MSV^{r,3}|$, i has seen $> 1/3$ majority for v'. Accordingly, player i sets $g_i = 1$, and Property (1) holds.

Does player i necessarily set $v_i = v'$? Assume there exists a different value $v'' \neq \perp$ such that player i has also seen $> 1/3$ majority for v''. Some of those messages may be from malicious verifiers, but at least one of them is from some honest verifier $j \in HSV^{r,3}$: indeed, because $|HSV^{r,3}| > 2|MSV^{r,3}|$ and i has received all messages from $HSV^{r,3}$, the set of malicious verifiers from whom i has received a valid $(r, 3)$-message counts for $< 1/3$ of all the valid messages he has received.

By definition, player j must have seen $> 2/3$ majority for v'' among all the valid $(r, 2)$-messages he has received. However, we already have that some other honest $(r, 3)$-verifiers have seen $> 2/3$ majority for v' (because they signed v'). By Property (d) of Lemma 5.5, this cannot happen and such a value v'' does not exist. Thus player i must have set $v_i = v'$ as desired, and Property (2) holds.

Finally, given that some honest $(r, 3)$-verifiers have seen $> 2/3$ majority for v', some (actually, more than half of) honest $(r, 2)$-verifiers have signed for v' and propagated their messages. By the construction of the protocol, those honest $(r, 2)$-verifiers must have received a valid message $m_\ell^{r,1}$ from some player $\ell \in SV^{r,1}$ with $v' = H(B_\ell^r)$, thus Property (3) holds.

BBA⋆. We again distinguish two cases.

Case 1. *All verifiers $i \in HSV^{r,4}$ have $b_i = 1$.*

This happens following Case 1 of GC. As $|MSV^{r,4}| < t_H$, in this case no verifier in $SV^{r,5}$ could collect or generate t_H valid $(r, 4)$-messages for bit 0. Thus, no honest verifier in $HSV^{r,5}$ would stop because he knows a non-empty block B^r.

Moreover, although there are at least t_H valid $(r, 4)$-messages for bit 1, $s' = 5$ does not satisfy $s' - 2 \equiv 1 \mod 3$, thus no honest verifier in $HSV^{r,5}$ would stop because he knows $B^r = B_\epsilon^r$. Instead, every verifier $i \in HSV^{r,5}$ acts at time $\beta_i^{r,5} = \alpha_i^{r,5} + t_5$, by when he has received all messages sent by $HSV^{r,4}$ following Lemma 5.5. Thus player i has seen $> 2/3$ majority for 1 and sets $b_i = 1$.

In Step 6 which is a Coin-Fixed-To-1 step, although $s' = 5$ satisfies $s' - 2 \equiv 0 \mod 3$, there do not exist t_H valid $(r, 4)$-messages for bit 0, thus no verifier in $HSV^{r,6}$ would stop because he knows a non-empty block B^r. However, with $s' = 6$, $s' - 2 \equiv 1 \mod 3$ and there do exist $|HSV^{r,5}| \geq t_H$ valid $(r, 5)$-messages for bit 1 from $HSV^{r,5}$.

For every verifier $i \in HSV^{r,6}$, following Lemma 5.5, on or before time $\alpha_i^{r,6} + t_6$ player i has received all messages from $HSV^{r,5}$, thus i stops without propagating anything and sets $B^r = B_\epsilon^r$. His $CERT^r$ is the set of t_H valid $(r, 5)$-messages $m_j^{r,5} = (ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,5})$ received by him when he stops.

Next, let player i be either an honest verifier in a step $s > 6$ or a generic honest user (i.e., non-verifier). Similar to the proof of Lemma 5.2, player i sets $B^r = B_\epsilon^r$ and sets his own $CERT^r$ to be the set of t_H valid $(r, 5)$-messages $m_j^{r,5} = (ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,5})$ he has received.

Finally, similar to Lemma 5.2,

$$T^{r+1} \leq \min_{i \in HSV^{r,6}} \alpha_i^{r,6} + t_6 \leq T^r + \lambda + t_6 = T^r + 10\lambda + \Lambda,$$

and all honest users know B^r in the time interval I^{r+1}, because the first honest user i who knows B^r has helped propagating the $(r, 5)$-messages in his $CERT^r$.

Case 2. *There exists a verifier $\hat{i} \in HSV^{r,4}$ with $b_{\hat{i}} = 0$.*

This happens following Case 2 of GC and is the more complex case. By the analysis of GC, in this case there exists a valid message $m_\ell^{r,1}$ such that $v_i = H(B_\ell^r)$ for all $i \in HSV^{r,4}$. Note that the verifiers in $HSV^{r,4}$ may not have an agreement on their b_i's.

For any step $s \in \{5, \ldots, m + 3\}$ and verifier $i \in HSV^{r,s}$, by Lemma 5.5 player i would have received all messages sent by all honest verifiers in $HSV^{r,4} \cup \cdots \cup HSV^{r,s-1}$ if he has waited for time t_s.

We now consider the following event E: *there exists a step $s^* \geq 5$ such that, for the first time in the binary BA, some player $i^* \in SV^{r,s^*}$ (whether malicious or honest) should stop without propagating anything.* We use "should stop" to emphasize the fact that, if player i^* is malicious, then he may pretend that he should not stop according to the protocol and propagate messages of the Adversary's choice.

Moreover, by the construction of the protocol, either

($E.a$) i^* is able to collect or generate at least t_H valid messages $m_j^{r,s'-1} = (ESIG_j(0), ESIG_j(v), \sigma_j^{r,s'-1})$ for the same v and s', with $5 \leq s' \leq s^*$ and $s' - 2 \equiv 0 \mod 3$; or

($E.b$) i^* is able to collect or generate at least t_H valid messages $m_j^{r,s'-1} = (ESIG_j(1), ESIG_j(v_j), \sigma_j^{r,s'-1})$ for the same s', with $6 \leq s' \leq s^*$ and $s' - 2 \equiv 1 \mod 3$.

Because the honest $(r, s'-1)$-messages are received by all honest (r, s')-verifiers before they are done waiting in Step s', and because the Adversary receives everything no later than the honest users, without loss of generality we have $s' = s^*$ and player i^* is malicious. Note that we did not require the value v in $E.a$ to be the hash of a valid block: as it will become clear in the analysis, $v = H(B_\ell^r)$ in this sub-event.

Below we first analyze Case 2 following event E, and then show that the value of s^* is essentially distributed accordingly to L^r (thus event E happens before Step $m+3$ with overwhelming probability given the relationships for parameters). To begin with, for any step $5 \leq s < s^*$, every honest verifier $i \in HSV^{r,s}$ has waited time t_s and set v_i to be the majority vote of the valid $(r, s-1)$-messages he has received. Since player i has received all honest $(r, s-1)$-messages following Lemma 5.5, since all honest verifiers in $HSV^{r,4}$ have signed $H(B_\ell^r)$ following Case 2 of GC, and since $|HSV^{r,s-1}| > 2|MSV^{r,s-1}|$ for each s, by induction we have that player i has set

$$v_i = H(B_\ell^r).$$

The same holds for every honest verifier $i \in HSV^{r,s^*}$ who does not stop without propagating anything. Now we consider Step s^* and distinguish four subcases.

Case 2.1.a. *Event $E.a$ happens and there exists an honest verifier $i' \in HSV^{r,s^*}$ who should also stop without propagating anything.*

In this case, we have $s^* - 2 \equiv 0 \mod 3$ and Step s^* is a Coin-Fixed-To-0 step. By definition, player i' has received at least t_H valid (r, s^*-1)-messages of the form $(ESIG_j(0), ESIG_j(v), \sigma_j^{r,s^*-1})$. Since all verifiers in HSV^{r,s^*-1} have signed $H(B_\ell^r)$ and $|MSV^{r,s^*-1}| < t_H$, we have $v = H(B_\ell^r)$.

Since at least $t_H - |MSV^{r,s^*-1}| \geq 1$ of the (r, s^*-1)-messages received by i' for 0 and v are sent by verifiers in HSV^{r,s^*-1} after time $T^r + t_{s^*-1} \geq T^r + t_4 \geq T^r + \lambda + \Lambda \geq \beta_\ell^{r,1} + \Lambda$, player i' has received $m_\ell^{r,1}$ by the time he receives those (r, s^*-1)-messages. Thus player i' stops without propagating anything; sets $B^r = B_\ell^r$; and sets his own $CERT^r$ to be the set of valid (r, s^*-1)-messages for 0 and v that he has received.

Next, we show that, any other verifier $i \in HSV^{r,s^*}$ has either stopped with $B^r = B_\ell^r$, or has set $b_i = 0$ and propagated $(ESIG_i(0), ESIG_i(H(B_\ell^r)), \sigma_i^{r,s})$. Indeed, because Step s^* is the first time some verifier should stop without propagating anything, there does not exist a step $s' < s^*$ with $s' - 2 \equiv 1 \mod 3$ such that t_H $(r, s'-1)$-verifiers have signed 1. Accordingly, no verifier in HSV^{r,s^*} stops with $B^r = B_\epsilon^r$.