Round-Efficient Byzantine Broadcast Under Strongly Adaptive and Majority Corruptions

Jun Wan[1(✉)], Hanshen Xiao[1], Srinivas Devadas[1], and Elaine Shi[2,3]

[1] Massachusetts Institute of Technology, Cambridge, USA
{junwan,hsxiao,devadas}@mit.edu
[2] CMU, Pittsburgh, USA
runting@gmail.com
[3] Cornell, Ithaca, USA

Abstract. The round complexity of Byzantine Broadcast (BB) has been a central question in distributed systems and cryptography. In the honest majority setting, expected constant round protocols have been known for decades even in the presence of a strongly adaptive adversary. In the corrupt majority setting, however, no protocol with sublinear round complexity is known, even when the adversary is allowed to *strongly adaptively* corrupt only 51% of the players, and even under reasonable setup or cryptographic assumptions. Recall that a strongly adaptive adversary can examine what original message an honest player would have wanted to send in some round, adaptively corrupt the player in the same round and make it send a completely different message instead.

In this paper, we are the first to construct a BB protocol with sublinear round complexity in the corrupt majority setting. Specifically, assuming the existence of time-lock puzzles with suitable hardness parameters and that the decisional linear assumption holds in suitable bilinear groups, we show how to achieve BB in $(\frac{n}{n-f})^2 \cdot \mathsf{poly}\log\lambda$ rounds with $1-\mathsf{negl}(\lambda)$ probability, where n denotes the total number of players, f denotes the maximum number of corrupt players, and λ is the security parameter. Our protocol completes in polylogarithmically many rounds even when 99% of the players can be corrupt.

1 Introduction

Byzantine Broadcast (BB), first defined by Lamport et al. [LSP82], is a foundational abstraction in distributed systems and cryptography, and has been studied for more than three decades. In Byzantine Broadcast, a designated sender wants to send a bit $b \in \{0,1\}$ to n nodes, and we would like to guarantee *consistency*, i.e., all honest nodes output the same bit; and *validity*, i.e., if the designated sender is honest, all honest nodes must output the sender's input. In BB, an important performance metric is the protocol's round complexity. Due to the elegant work of Dolev and Strong [DS83], it is long known that any *deterministic* BB protocol must incur at least $\Omega(n)$ number of rounds, and indeed Dolev

R. Pass and K. Pietrzak (Eds.): TCC 2020, LNCS 12550, pp. 412–456, 2020.
https://doi.org/10.1007/978-3-030-64375-1_15

and Strong [DS83] also demonstrate a round-optimal deterministic protocol with $\Theta(n)$ rounds. It is also well-known that with randomization, expected constant-round BB protocols exist in the honest-majority setting [FM97, KK09, ADD+19]. On the other hand, for quite a long time, no sublinear (randomized) protocol was known for the corrupt majority setting. In 2007, after progress had been stagnant for a long while, Garay et al. [GKKO07] first showed a glimpse of hope for the corrupt majority setting, by constructing a protocol that achieved $O((2f-n)^2)$ round complexity where f denotes the number of corrupt nodes. Subsequently, Fitzi et al. [FN09] improved their result to $O(2f-n)$ rounds. Both Garay et al. [GKKO07] and Fitzi et al. [FN09], however, were somewhat unsatisfying, since the regime under which they give sublinear round complexity is rather narrow: even the latter work [FN09] requires $f/n - \frac{1}{2}$ to be $o(1)$ fraction to achieve sublinear round complexity. Even when $f = 51\% \cdot n$, both these works would incur at least linear number of rounds. Progress became somewhat stagnant again until very recently, Chan et al. [CPS20] made some long-awaited progress, demonstrating a new BB protocol that achieved $O(\frac{n}{n-f}) \cdot \mathsf{poly}\log\lambda$ number of rounds even in the corrupt majority setting, where the protocol's failure probability is guaranteed to be negligibly small in the security parameter λ. Interestingly, their result is also optimal up to a poly-logarithmic factor due to an $\Omega(\frac{n}{n-f})$ lower bound by Garay et al. [GKKO07] even for randomized protocols and even assuming static corruptions. Subsequently, a companion work by Wan et al. [WXSD20] showed how to construct expected $O((\frac{n}{n-f})^2)$-round BB under corrupt majority; and this result can be viewed as a further improvement of Chan et al. [CPS20] for a broad range of parameters, e.g., when 1% (or any arbitrarily small constant fraction) of the nodes are honest.

Nonetheless, the constructions by Chan et al. [CPS20] and Wan et al. [WXSD20] remain somewhat unsatisfying, since to achieve their result, the two works [CPS20, WXSD20] had to significantly weaken the adversary's capabilities relative to the prior results in this space. All aforementioned works [DS83, GKKO07, FN09] prior to Chan et al. [CPS20] secured against a *strongly adaptive* adversary, i.e., the adversary can observe the messages in flight from honest nodes in a round, adaptively corrupt a subset of nodes and erase an arbitrary subset of their messages in flight, and moreover, make the newly corrupt nodes send additional messages in the same round. In fact, the strongly adaptive model is the well-accepted model in the early days of distributed consensus and multi-party protocols (see also Definition 1 in Feldman's thesis [Fel88] and Figure 4, page 176 of Canetti's excellent work [Can00]). By contrast, the approaches of Chan et al. [CPS20] and Wan et al. [WXSD20] defend only against a *weakly adaptive* adversary—such an adversary can observe the messages honest nodes want to send in a round, adaptively corrupt a subset of the nodes, and make the newly corrupt nodes send additional messages in the same round; however, the adversary cannot perform "after-the-fact-removal", i.e., it cannot retroactively erase the messages any node had sent before it became corrupt in the same round. The weakly adaptive model is akin to the atomic message model first introduced by Garay et al. [GKKZ11] as a way to overcome

a lower bound pertaining to a particular adaptive, simulation-based notion of security proven by Hirt and Zikas [HZ10]. The only slight difference is that in the atomic message model, not only is the adversary unable to perform "after-the-fact" message removal, it also must wait for one network delay after a node i becomes corrupt, before it is able to inject messages on behalf of i. More recently, a line of works inspired by blockchains [GKL15,PSS17,PS17c,PS17d,DPS16, ACD+19,CPS19a] also adopted the weakly adaptive model to get bandwidth-efficient protocols—it turns out that the weakly adaptive relaxation is necessary for constructing Byzantine Agreement with sublinear communication complexity [ACD+19].

In some settings, however, the weakly adaptive model may be unsatisfactory, e.g., if the adversary can control intermediate routers in the network, it indeed can examine the honest messages in flight, then corrupt a subset of the nodes and fail to deliver any subset of their messages already in flight. Thus, the state of the art begs the following natural question,

Are there (randomized) BB protocols with sublinear round complexity, and secure in the presence of a strongly adaptive adversary that is allowed to corrupt a majority of the nodes?

1.1 Our Results and Contributions

Main Result. We give the first affirmative answer to the above question, and to achieve this we rely on the existence of a trusted setup, the decisional linear assumption in suitable bilinear groups, as well as the existence of time-lock puzzles with suitable hardness parameters. Our main result is stated in the following theorem.

Theorem 1.1 (Main result)**.** *Assuming the existence of a trusted setup, the decisional linear assumption in suitable bilinear groups (see Appendix A.1), as well as the existence of time-lock puzzles* [RSW96] *with hardness parameter ξ, there exists a protocol that achieves BB in $(\frac{n}{n-f})^2 \cdot \frac{\mathsf{poly}\log\lambda}{\xi}$ number of rounds with probability $1 - \mathsf{negl}(\lambda)$ where $\mathsf{negl}(\cdot)$ is a suitable negligible function (of the security parameter λ).*

More concretely, a time-lock puzzle with hardness parameter ξ ensures that the puzzle solution remains hidden from any machine running in time that is at most ξ fraction of the honest evaluation time, even when the machine has access to unbounded polynomial parallelism. As a typical example, consider the case when $\xi \in (0, 1)$ is a constant just like what prior works have assumed [RSW96, BBBF18], and moreover, suppose that $\frac{n}{n-f}$ is also a constant (e.g., 99% may be corrupt)—in this case, our protocol's round complexity is simply $\mathsf{poly}\log\lambda$.

To the best of our knowledge, no prior work can achieve sublinear-round BB in the strongly adaptive setting under any reasonable setup assumption, and even for only 51% corruption. In this sense our result significantly improves our understanding of the round complexity of BB.

Interpreting the Result. Our result currently requires a trusted setup. We do not know if the trusted setup is necessary, but some form of setup is necessary: without any setup, Byzantine Broadcast is impossible under 1/3 or more corruptions due to an elegant lower bound by Lamport et al. [LSP82]. Besides trusted setup, we also assume the existence of time-lock puzzles; therefore, another open question is to understand whether time-lock puzzles are necessary. In fact, without time-lock puzzles, *even sublinear-round BB* under 51% strongly adaptive corruption remains open. New upper- or lower-bounds along this direction would be very exciting and seem very challenging.

Another natural question is whether we can improve the round complexity to expected constant. Due to the companion result of Wan et al. [WXSD20], we know that expected constant round is possible with a weakly adaptive adversary, and assuming trusted setup and the decisional linear assumption in suitable bilinear groups. Naturally, it seems tempting to ask for the same in the strongly adaptive setting. Unfortunately, with our techniques, we do not know how to go beyond polylogarithmic number of rounds. In fact, even our underlying message distribution primitive itself already takes polylogarithmically many rounds as we explain in Sect. 2 and the subsequent technical sections. Therefore, whether expected constant round BB is possible for 51% *strongly* adaptive corruption remains an open question—constructing an upper bound seems challenging even assuming static corruption, and allowing any reasonable setup assumptions; similarly, whether there is possibly a lower bound also seems challenging.

1.2 Technical Highlights

We give a high-level overview of our main techniques.

Delayed-Exposure Message Distribution. One major new technique we introduce is a delayed-exposure message distribution mechanism. Specifically, we devise a novel poly-logarithmic round, randomized protocol that allows all n honest nodes to each distribute a time-lock puzzle that embeds a message they want to send; moreover, by the end of poly-logarithmically many rounds, all honest nodes can obtain the solutions of all other honest nodes' puzzles. On the other hand, even if the adversary has unbounded parallelism, it cannot learn any information about honest nodes' messages encoded in the puzzles within one round of time; and thus the adversary cannot make informed adaptive corruptions based on the message contents.

To solve this problem, we need to overcome several technical challenges. First, although we allow the adversary to have access to unbounded parallelism, it is unrealistic to expect that honest machines are also equipped with up to n amount of parallelism. Like in the standard distributed protocol literature, we assume that the honest nodes are *sequential* RAM machines and thus they cannot solve puzzles in parallel. However, if they solved all the puzzles sequentially it would take linear number of rounds which is what we want to avoid in the first place.

Second, the honest nodes do not even have a consistent view of the puzzles being distributed which makes it difficult to coordinate who solves which

puzzles. To overcome these challenges, we devise a novel *age-based sampling* technique where nodes sample puzzles to solve and the probability of sampling grows exponentially w.r.t. how long ago the puzzle was first seen. We defer the detailed construction and its analysis to later sections.

We stress that the delayed-exposure primitive can be of independent interest in other protocol design contexts—in this sense, besides our new construction, we also make a conceptual contribution in defining this new primitive and formulating its security properties (see Sect. 4).

Applying the Delayed-Exposure Distribution Mechanism. Once we have the delayed-exposure distribution mechanism, we can combine it with techniques proposed in the recent work by Chan et al. [CPS20], and upgrade their weakly adaptive protocol to a strongly adaptive one while still preserving a polylogarithmic round complexity. For this upgrade, the most challenging aspect is how to prove security. The most natural approach towards proving security is to first prove that the real-world protocol securely emulates a natural ideal-world counterpart, and then argue the security of the ideal-world protocol (which does not use cryptography) using an information-theoretic argument. Unfortunately, this approach fails partly because the time-lock puzzles only provide transient secrecy of the messages they encode. Instead, we work around this issue by devising a sequence of hybrids from the real-world execution to an ideal-world execution without cryptography, and we show that for every adjacent pair of hybrids, the probability of certain relevant bad events can only increase. Eventually, we upper bound the probability of bad events in the ideal world using an information theoretic argument.

Soundness of Cryptography w.r.t. an Adaptive Adversary. Last but not the least, in our construction and when arguing about the sequence of hybrids, one technicality that arises is that the adversary is strongly adaptive, and therefore some of the cryptographic building blocks we use must be commensurate and secure against selective opening attacks. This technicality will show up throughout the paper in our definitions, constructions, and proofs.

1.3 Additional Related Work

Several other works [KY, CMS89] proved lower bounds on the *worst-case* round complexity of randomized BA; and an online full version [CPS19b] of the recent work by Chan et al. [CPS20] presented a complete version of these proofs. Note that these lower bounds are incomparable to Garay et al.'s lower bound [GKKO07]. Cohen et al. [CHM+19] prove lower bounds on the round complexity of randomized Byzantine agreement (BA) protocols, bounding the halting probability of such protocols after one and two rounds.

A line of works in the literature [HZ10, GKKZ11, CCGZ16] have focused on a simulation-based notion of adaptive security for Byzantine Broadcast, where the concern is that the adversary should not be able to observe what the sender wants to broadcast, and then adaptively corrupt the sender to flip the bit. This simulation-based notion is stronger than the property-based security definitions

in this paper. To achieve this strong notion of security, Garay et al. [GKKZ11] adopted the "atomic message model". As mentioned earlier, the atomic message model is almost the same as our weakly adaptive model, except that in the atomic message model, when a node i becomes newly corrupt, the adversary must wait for a network delay before it can inject corrupt messages on behalf of i.

In this paper, we consider the "broadcast" version of consensus commonly called Byzantine Broadcast. There is also another "agreement" version of the formulation, commonly called Byzantine Agreement. In the agreement version, each node has an input bit b and they all want to agree on a bit. The consistency requirement is unchanged, and the validity requirement instead stipulates that if all honest nodes have the same input bit b, then all honest nodes must output b. It turns out that the agreement version of the formulation is only possible assuming honest majority, and that is why our paper does not discuss this formulation.

A line of work has focused on a repeated consensus abstraction either called State Machine Replication [Sch90, PS17b, PS17a, CL99, Lam98, GKL15] or blockchains. Imprecisely speaking, a blockchain protocol must reach consensus repeatedly over time whereas Byzantine Broadcast achieves single-shot consensus. There are typically two approaches for constructing a blockchain protocol: 1) through sequential/parallel composition of a single-shot abstraction (e.g., Byzantine Broadcast); and 2) direct blockchain construction [CL99, Lam98, YMR+19, CS20].

Finally, our paper is not the first that uses time-lock puzzles in the context of distributed consensus. Prior works have used time-lock puzzles to construct "proof-of-work" type of consensus protocols [KMS14, EFL17].

2 Technical Roadmap

For simplicity, in our informal technical roadmap, we may assume that the adversary may adaptively corrupt an arbitrarily large constant fraction of the nodes. In other words, we assume that $f = (1 - \epsilon)n$ in the remainder of this section for an arbitrarily small constant $\epsilon \in (0, 1)$. Our full protocol in the subsequent sections will be stated formally for more general parameters.

The most natural starting point appears to be the very recent work by Chan et al. [CPS20] which achieves BB with polylogarithmic round complexity in a weakly adaptive corruption model even in the presence of majority corruptions. Unfortunately, their protocol needed the weakly adaptive restriction not just in the proofs; in fact, their protocol is prone to an explicit attack that breaks consistency assuming a strongly adaptive adversary.

2.1 Chan et al. Breaks Under a Strongly Adaptive Adversary

To aid understanding, we first describe Chan et al.'s approach [CPS20] at a very high level. Their main idea is a new method of committee election relying on an adaptively secure Verifiable Random Function (VRF) [MVR99, ACD+19]

which they call "bit-specific" committee election. More concretely, during setup, a VRF public- and secret-key pair denoted $(\mathsf{pk}_u, \mathsf{sk}_u)$ is selected for every node $u \in [n]$ and the corresponding public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$ are published in the sky. Recall that a designated sender wants to broadcast a bit to all other nodes. Using the VRF, we can define two committees called the 0-committee and the 1-committee, each responsible for voting for the 0-bit and the 1-bit, respectively. Specifically, the b-committee consists of all nodes whose VRF evaluation outcome on the input b is smaller than an appropriate difficulty parameter. The difficulty parameter is chosen such that each committee's size is polylogarithmic in expectation. Now, committee members for each bit b will engage in poly-logarithmically many rounds of voting based on certain voting rules; moreover, all nodes, including committee members and non-committee members, keep relaying the votes they have seen. We will not go into the details of the voting rules, but what is important here is that the security of their scheme critically relies on the fact that the committee members remain secret until they actually cast a vote for the corresponding bit b. More specifically, after the setup phase, each node knows whether it is a member of the 0-committee (or the 1-committee resp.) but this knowledge is kept secret until the node has cast a vote for the bit 0 (or the bit 1 resp.). Further, when a vote for b is cast, the vote is attached with a VRF proof that vouches for the fact that the voter is a member of the b-committee.

In Chan et al.'s scheme [CPS20], if somehow the adversary could predict who is in which committee, then security could be broken, since the adversary would have enough budget to corrupt all members of the 0-committee (or the 1-committee resp.) before their vote gets propagated to everyone. However, since the VRF scheme satisfies pseudorandomness under adaptive corruptions, essentially the adversary cannot effectively guess which nodes are in either committee until the nodes actually cast a vote for the corresponding bit b, divulging the fact that they are in the b-committee. Even though upon observing a node u's vote for a bit b, the adversary can act instantly and corrupt the voter u (who must be a member of the b-committee), it is already too late—u's vote is guaranteed to propagate to all other nodes since a weakly adaptively adversary cannot retroactively erase the vote u had already sent prior to corruption.

Now, if the adversary is actually strongly adaptive, then an explicit attack is to wait till a node u casts a vote for b, act immediately and corrupt u in the same round, and cause u's vote to be delivered to a subset of the honest recipients but not all of them—without going into details, such an attack would break the consistency of Chan et al.'s protocol.

It might be tempting to try to fix the above problem with naïve solutions along the following vein: have all honest nodes first commit to their messages (which is either a valid vote or a dummy message), wait for a while, and then open their messages to reveal whether it is a vote. However, such naïve attempts do not fundamentally fix the problem, because a strongly adaptive adversary can always immediately erase a vote as soon as it is opened.

2.2 A Strawman Scheme

A first strawman idea is to use time-lock puzzles to transiently hide message contents and thereby defeat the agility of the strongly adaptive adversary. Imagine that in some round, some members of the b-committee want to cast a vote for b (henceforth called voters), and other nodes do not want to cast votes (henceforth called non-voters). Recall that the adversary cannot predict a-priori which nodes are members of the b-committee, but if the votes are cast in the clear, then the voter immediately reveals itself to be a b-committee member.

Our idea is 1) for voters to lock the votes temporarily in a time-lock puzzle and send the resulting puzzle rather than the clear-text vote; and 2) for non-voters to send chaff of the same length, also temporarily locked in puzzles. Even if the adversary may have unbounded parallelism, it cannot distinguish within one round of time which nodes are voters and which ones are non-voters. Although the adversary can adaptively corrupt a subset of nodes and prevent their puzzles from being delivered, such adaptive corruption is basically performed in a blindfolded manner. Finally, if a node is not corrupt in the round in which the puzzle is sent, essentially the puzzle is let through and honest nodes can solve it later given enough time and obtain the message locked inside it.

In the strawman scheme, each voting round is prolonged to the time needed for a single node to solve *all* puzzles (plus one more round for sending the puzzles). If every honest node had n parallel processors, then it could solve all puzzles in parallel consuming only a constant number of rounds. However, it is quite unreasonable to assume that all honest nodes have so much parallelism—in particular, note that the amount of parallelism must scale linearly with the number of nodes. Therefore, we would like a better solution where the honest nodes may run on sequential machines, and yet the adversary is allowed unbounded polynomial parallelism.

2.3 Our Approach

In our approach, all nodes propagate their own puzzle to everyone else in the first round. If a node remains honest till the end of the first round, then its puzzle is guaranteed to be received by all honest nodes—henceforth we call such puzzles *honest puzzles*. Puzzles sent by nodes that are corrupt before the end of the first round are said to be *corrupt puzzles*.

We now repeat logarithmically many iterations: in each iteration, all nodes share the workload of solving the puzzles, and send solutions to each other. After logarithmically many iterations, we will show that except with negligible probability, all honest puzzles will have been solved and their solutions received by all honest nodes (but we do not guarantee that corrupt puzzles are also solved).

To make this idea work, however, is non-trivial, and we are faced with several challenges. One difficulty arises from the fact that the honest nodes do not have common knowledge of the set of puzzles at the start of the protocol, since corrupt nodes can reveal their puzzles only to a subset of the honest nodes. Similarly,

at the end of each iteration, honest nodes also do not have common knowledge of which set of puzzles have been solved. Therefore, nodes must coordinate and share the work-load of solving puzzles, regardless of their different views of what the remaining puzzles are. Our idea is for nodes to randomly select a somewhat small subset of puzzles to solve in every iteration but how to choose a random subset is somewhat tricky.

Idealized Randomized Process Assuming Perfect Knowledge of Leftover Honest Puzzles. Although we use randomness to overcome the inconsistency in nodes' views of the remaining puzzle set, it still helps to first think of how a random strategy might work in a perfect, imaginary world where everyone always knows which are the set of *leftover* honest puzzles at the beginning of each iteration—here, a puzzle is said to be *leftover* if no so-far honest node has solved it and propagated its solution. In such a perfect world, we could use the following randomized strategy: in the first iteration, there are at most n honest puzzles to start with. Now, everyone chooses each of the n puzzles with some probability p_1 such that the expected number of puzzles each node solves is $p_1 \cdot n$. Note that the adversary can examine which puzzles' solutions each node is propagating at the end of the first iteration, and then adaptively decide on a set of nodes to corrupt, and make these nodes fail to propagate their puzzle solutions. If all honest nodes that tried to solve a specific puzzle Z are corrupt, then Z will be leftover. A smart adversary can try to pick a set of nodes to corrupt such that the set of leftover honest puzzles is maximized. One can prove that as long as $p_1 \cdot n$ is a sufficiently large constant, even if the adversary chooses the worst-case set of size $(1 - \epsilon)n$ to corrupt, there cannot be more than $n/2$ leftover honest puzzles except with negligible probability. In other words, the adversary cannot simultaneously deny too many honest puzzles from being solved. Now, in the second iteration, there at most $n/2$ honest puzzles left, and we can repeat the same but setting $p_2 = 2p_1$, i.e., the probability of sampling each honest puzzle doubles. Again, one can show that after two iterations, there cannot be more than $n/4$ leftover honest puzzles except with negligible probability, and this goes on for logarithmically many rounds at which point all honest puzzles are solved except with negligible probability.

Working with Imperfect Knowledge and Corrupt Puzzles. Our actual protocol needs to somehow "embed" the above idealized random process in a world where there can be corrupt puzzles, and moreover, honest nodes do not have a consistent view of which puzzles are solved. This turns out to be tricky—for example, the simplest idea is for nodes to always pick puzzles from the set of puzzles they have seen by the end of the first round (recall that during the first round nodes propagate their own puzzles to each other). However, if in the first round, the adversary discloses $\Theta(n)$ corrupt puzzles (henceforth denoted Q) only to one honest node u, then no one else will be helping to solve these corrupt puzzles Q; and if the probability p keeps doubling in each iteration, u will need to solve $\Theta(n)$ puzzles in the last iteration. So we want u to be able to propagate Q to others, so that when others receive them, they can start solving them too. But this introduces a new issue: the adversary can suddenly disclose a new set

of $\Theta(n)$ puzzles late in the protocol, i.e., when the probability p has doubled logarithmically many times and is close to 1. In this case, the same node would have to solve too many puzzles.

To overcome the above issues, our approach adjusts the sampling probability based on the puzzle's *age*. Roughly speaking, we define a puzzle's age as how many iterations ago the puzzle was first seen. Given a puzzle of age α, we will sample it with probability $p_1 \cdot 2^\alpha$. In other words, the older the puzzle is, the more likely it will get sampled, and the probability of being sampled doubles with every iteration. Finally, if a node v is detected to have double-signed more than one puzzle, all of v's puzzles will henceforth be ignored.

We will prove later in the technical sections that except with negligible probability, in every iteration, the number of leftover puzzles of age α in the union of the honest nodes' views is upper bounded by $n/2^{\alpha-1}$, as long as each iteration is long enough such that honest nodes can indeed solve all puzzles they have sampled. Note that since puzzles of age α are each sampled with probability $p_1 \cdot 2^\alpha$, the expected number of puzzles of age α that are chosen is $\Theta(np_1) = \Theta(1)$. By the Chernoff bound, we can show that except with $\mathsf{negl}(\lambda)$ probability, no more than $\mathsf{poly}\log\lambda$ puzzles of each age α are chosen. Since the protocol runs for logarithmically many iterations, there can be at most logarithmically many different ages. Therefore, in each iteration, every node must solve only polylogarithmically many puzzles (except with negligible probability).

Other Subtleties. So far, we have implicitly assumed that there is a way to convince another node that some purported puzzle solution is indeed the correct solution, since otherwise the adversary can convince honest nodes to accept wrong solutions. To make sure this is indeed the case, honest nodes sign the message they want to distribute and then lock both the message and the signature inside a puzzle. In this way, a valid solution for a correctly constructed puzzle would be verifiable. However, this is not enough since the adversary can still construct bad puzzles, and when honest nodes solve them, they cannot convince others that the solution is valid. This issue can be fixed if we simply attach a zero-knowledge proof to the puzzle vouching for the fact that it correctly encodes a message and a signature from the purported sender.

Putting it All Together. In summary, to obtain Byzantine Broadcast, we first construct a *delayed-exposure message distribution* mechanism called Distribute. In a Distribute protocol, at the beginning every node u receives an input message m_u, and each node would like to distribute its input messages to others. If a node u remains honest at the end of the first round of the protocol, we say its input m_u is an *honest* message.

The Distribute protocol guarantees the following: 1) *liveness*, i.e., all honest messages must be delivered to every honest node at the end of the protocol; and 2) *momentary secrecy*, i.e., by the end of the first round, no probabilistic polynomial-time adversary, even when allowed unbounded parallelism, could have learned any information about the honest messages (although eventually, given sufficiently long polynomial time, the adversary could solve the puzzles

and learn the input messages). The protocol works as follows (described below for the special case when $f = (1-\epsilon)n$ where $\epsilon \in (0,1)$ is a constant):

- **Round 1:** Every node $u \in [n]$ computes a signature σ_u on its input message m_u, and computes a puzzle Z_u that encodes (m_u, σ_u). Further, u computes a non-interactive zero-knowledge proof denoted π_u that the puzzle Z_u indeed encodes a pair where the second term is a valid signature on the first one (w.r.t. u's public key).
  Node u now propagates (Z_u, π_u) to everyone along with a signature on the pair.
- **Repeat $\Theta(\log n)$ iterations:** Each iteration has duration $\mathcal{T}_{\text{solve}} \cdot \mathsf{poly}\log\lambda + 1$ where $\mathcal{T}_{\text{solve}}$ is the time it takes for a sequential machine to solve a single puzzle. During each iteration, a node samples each puzzle of age α to solve with independent probability $\min(p_1 \cdot 2^{\alpha}, 1)$, and once solved it propagates the solution (m, σ) to others if σ is a valid signature for m from the purported sender of m.
  At any time, if a node v is detected to have double-signed two different puzzles, all puzzles signed by v will henceforth be ignored.
- **Output.** Finally, for each node $v \in [n]$, if a valid pair (m, σ) has been observed where σ is a valid signature on m under v's public key, then (m, σ) is output as the message received from v; if no such valid pair has been seen, output $\perp$ as the received message from v.

Intuitively, the security properties of Distribute ensure that honest messages will indeed be delivered, and moreover, the adversary cannot base its corruption decisions based on the contents of the messages, and must make corruptions blindly. To get Byzantine Broadcast, we can now plug in the Distribute protocol to distribute batches of votes in the protocol by Chan et al. [CPS20]. Just like in the strawman scheme in Sect. 2.2, here, nodes who do not want to transmit batches of votes must transmit chaff using the Distribute protocol. Through this transformation, we effectively constrain a strongly adaptive adversary such that its capability is roughly the same as a weakly adaptive adversary. We defer the details of the Byzantine Broadcast (BB) protocol to the subsequent technical sections.

Challenges in Proving Security. Although the intuition is clear, formally reasoning the security of our BB protocol is actually rather subtle due to the use of cryptography. Ideally, we would like to abstract the cryptography away and reason about the core randomized process captured by the protocol in an ideal-world execution. Unfortunately, the real-world protocol does *not* securely emulate the most natural ideal-world protocol that captures the core randomized process. For example, one concrete challenge is the following: we would like to argue that the first-round messages of a Distribute protocol can be simulated by a simulator without knowing the actual input messages of the honest nodes. Unfortunately, the simulated messages can only fool the adversary for a small amount of time, and the adversary could eventually discover that they were being simulated.

To tackle this challenge, our actual proof defines a sequence of hybrids from the real-world experiment with cryptography, to an ideal-world experiment without cryptography. Instead of arguing that the adversary's view is computationally indistinguishable in adjacent hybrids, we argue that the probability of certain bad events happening in the next hybrid is an upper bound of the probability in the previous hybrid (ignoring negligible differences). This means that bad events cannot happen with higher probability in the real-world than in the ideal-world. Finally, since the ideal-world execution does not involve cryptography, we can argue through a probabilistic argument that the probability of relevant bad events is negligibly small.

Finally, another subtlety in both our construction and proofs is the fact that the adversary is adaptive; and therefore we need to rely on cryptographic primitives with suitable adaptive notions of security.

Remark 1 (On how general our Distributeprimitive is)**.** One natural question is whether our Distribute primitive can be used to upgrade any weakly adaptive protocol to a strongly adaptive one preserving its security properties. Our result does not imply such a general weakly to strongly adaptive compiler, partly due to the technical challenges mentioned earlier, specifically, the fact that we cannot prove that the real-world protocol emulates some natural ideal-world protocol. As an exciting future direction, it would be great to understand how general our Distribute primitive is, i.e., for which class of protocols it is applicable. Further, another exciting question is whether we can get a general weakly to strongly adaptive compiler (see also Sect. 7).

3 Preliminaries

3.1 Definitions

Protocol Execution Model. We assume a standard protocol execution model with n nodes numbered $1, 2, \ldots, n$, respectively. An adversary denoted $\mathcal{A}$ can adaptively corrupt nodes during the middle of the execution. The nodes that have not been corrupted are called honest nodes. All corrupt nodes are under the control of $\mathcal{A}$, i.e., the messages they receive are forwarded to $\mathcal{A}$, and $\mathcal{A}$ controls what messages they will send once they become corrupt.

We assume a *synchronous* network model, i.e., honest nodes can send messages to each other within one *round*. More precisely, if an honest node u sends a message to v during round r, as long as u and v are still honest at the beginning of round $r+1$, then v would have received the message by the beginning of round $r+1$. We assume that in each round, a node first reads incoming messages from the network, then it performs some local computation and sends messages.

The adversary $\mathcal{A}$ is allowed to examine the messages honest nodes send during a round r, and then decide who to corrupt in round r, and what messages corrupt nodes send in round r. If a node u becomes newly corrupt in round r, any message it wanted to send in round r can be erased by $\mathcal{A}$; further, $\mathcal{A}$ can make the newly

corrupt u send additional messages of its choice in the same round r. Recall that such an adversary is said to be *strongly adaptive*.

We assume that the protocol's execution may be parameterized by a security parameter denoted $\lambda \in \mathbb{N}$. We would like the protocol to ensure the desired security properties with $1 - \mathsf{negl}(\lambda)$ probability for some negligible function $\mathsf{negl}(\cdot)$.

Modeling Honest and Adversarial Machines. We model honest nodes as probablistic, sequential Random Access Machines (RAMs), and model the adversary as a non-uniform probabilistic, parallel machine with unbounded polynomial parallelism running in unbounded polynomial parallel time.

In constructing our protocol, we will leverage certain cryptographic primitives whose security is only guaranteed against an adversary that is restricted to run in a small, bounded number of parallel steps. Therefore, we define a T-bounded adversary as follows:

Definition 3.1. We say that $\mathcal{A}$ is a T-bounded, non-uniform $p.p.t.$ parallel machine iff $\mathcal{A}$ is a non-uniform probabilistic parallel machine with unbounded polynomial parallelism, but restricted to run in at most T parallel steps. Note that here the usage of the term $p.p.t.$ actually means polynomially bounded in total work—since in the parallel algorithms literature, the terms "work" and "sequential time" are used interchangeably to describe a PRAM algorithm's work, we preserve the familiar short-hand $p.p.t.$.

Note that although some of our underlying cryptographic primitives are secure only against T-bounded adversaries, we need to prove our protocol secure against an adversary running in unbounded parallel time. This is partly why our proofs are non-trivial (see Sect. 6.2 for more details).

Duration of a Round. Finally, we discuss the duration of a *round* in our execution model. The standard distributed systems and cryptography literature implicitly assumes that a round is of polynomial duration and it is long enough such that an honest node can perform the prescribed cryptographic operations, e.g., verify signatures on all messages received, verify the zero-knowledge proofs attached to the messages received, sign the messages it wants to send, and so on. We make the same assumption in this paper, with the exception of the Solve algorithm of our time-lock puzzle scheme. Specifically, we will later on parametrize our time-lock puzzle such that a sequential machine (e.g., an honest node's machine) would take multiple rounds to solve a single puzzle.

Byzantine Broadcast. Recall that there are n nodes, and without loss of generality, we call node 1 the designated sender. Prior to protocol start, the designated sender receives an input $b \in \{0, 1\}$ from $\mathcal{A}$. At the end of the protocol, every node $u \in [n]$ outputs a bit b_u. We would like to guarantee the following security properties with $1 - \mathsf{negl}(\lambda)$ probability over the randomized execution:

- *Consistency:* if a forever-honest node u outputs a bit b_u and a forever-honest node v outputs a bit b_v, it must be that $b_u = b_v$;
- *Validity:* if the designated sender is forever-honest, it must be that every forever-honest node outputs the sender's input bit b.

In the above, forever-honest means that the corresponding node remains honest till the end of the protocol.

Notations. Throughout the paper, we use n to denote the total number of nodes, f to denote the maximum number of corrupt nodes, and $h = n - f$ to denote the number of honest nodes. Since we care about the asymptotical behavior of the round complexity w.r.t. n, without loss of generality, we may assume $n \geq \log^2 \lambda$ where λ is the security parameter as mentioned.

3.2 Time-Lock Puzzles

We review the notion of time-lock puzzles [BGJ+16, RSW96, LPS17].

Definition 3.2 (Time-lock puzzles)**.** Let $\mathcal{S}$ be a finite domain of size at most $2^{\mathsf{poly}(\lambda)}$. A time-lock puzzle (TLP) with solution space $\mathcal{S}$ is a tuple of algorithms $(\mathsf{Gen}, \mathsf{Solve})$ as defined below:

- $Z \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{T}, s)$: a probabilistic algorithm that takes a security parameter λ, a time parameter $\mathcal{T}$, and a solution $s \in \mathcal{S}$, and outputs a puzzle Z.
- $s \leftarrow \mathsf{Solve}(Z)$: a deterministic algorithm that takes in a puzzle Z, and outputs a solution s.

Correctness. We require that for every λ, every $s \in \mathcal{S}$, every $\mathcal{T}$, $\mathsf{Solve}(\mathsf{Gen}(1^\lambda, \mathcal{T}, s))$ outputs the correct solution s with probability 1.

Efficiency. We require that on a sequential Random-Access Machine, $\mathsf{Gen}(1^\lambda, \mathcal{T}, s)$ runs in at most $\mathsf{poly}(\lambda, \log \mathcal{T})$ steps for any $s \in \mathcal{S}$; and moreover $\mathsf{Solve}(Z)$ runs in at most $\mathcal{T}$ number of steps for any Z in the support of $\mathsf{Gen}(1^\lambda, \mathcal{T}, \cdot)$.

ξ-hardness. An TLP scheme $(\mathsf{Gen}, \mathsf{Solve})$ is said to be ξ-hard iff there exists a polynomial $\widetilde{\mathcal{T}}$ such that for all polynomials $\mathcal{T}(\cdot) \geq \widetilde{\mathcal{T}}(\cdot)$ and every $\xi\mathcal{T}$-bounded, non-uniform $p.p.t.$ parallel machine $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for all $\lambda \in \mathbb{N}$ and for all $s_0, s_1 \in \mathcal{S}$ it holds that

$$\left|\Pr\left[\mathcal{A}(\mathsf{Gen}(1^\lambda, \mathcal{T}, s_0)) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{Gen}(1^\lambda, \mathcal{T}, s_1)) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

3.3 Verifiable Random Functions

A verifiable random function (VRF) [MVR99] includes the following (possibly randomized) algorithms:

- $(\mathsf{crs}, \{\mathsf{pk}_u, \mathsf{sk}_u\}_{u\in[n]}) \leftarrow \mathsf{Gen}(1^\lambda)$: takes in a security parameter λ and generates public parameters crs, and a public and secret key pair $(\mathsf{pk}_u, \mathsf{sk}_u)$ for each node $u \in [n]$; each sk_u is of the form $\mathsf{sk}_u := (s_u, \rho_u)$ where s_u is said to be the *evaluation key* and ρ_u is said to be the *proof key* for u.

- $(y, \sigma) \leftarrow \mathsf{Eval}(\mathsf{crs}, \mathsf{sk}_u, x)$: we shall assume that $\mathsf{Eval} := (E, P)$ has two sub-routines E and P where $\mathsf{Eval}.E$ is *deterministic* and $\mathsf{Eval}.P$ is possibly randomized. Given the public parameters crs, the secret key $\mathsf{sk}_u = (s_u, \rho_u)$, and input $x \in \{0,1\}^{|x|}$, compute $y := \mathsf{Eval}.E(\mathsf{crs}, s_u, x)$ and $\sigma := \mathsf{Eval}.P(\mathsf{crs}, s_u, \rho_u, x)$, and output (y, σ).
- $\{0,1\} \leftarrow \mathsf{Vf}(\mathsf{crs}, \mathsf{pk}_u, x, y, \sigma)$: receives the public parameters crs, a public key pk_u, an input x, a purported outcome y, and a proof σ, outputs either 0 indicating rejection or 1 indicating acceptance.

For the VRF scheme to satisfy correctness, we require that for any $v \in [n]$, for any input x, the following holds with probability 1: let $(\mathsf{crs}, \{\mathsf{pk}_u, \mathsf{sk}_u\}_{u\in[n]}) \leftarrow \mathsf{Gen}(1^\lambda)$, and let $(y, \sigma) \leftarrow \mathsf{Eval}(\mathsf{crs}, \mathsf{sk}_v, x)$, then it must be that $\mathsf{Vf}(\mathsf{crs}, \mathsf{pk}_v, x, y, \sigma) = 1$.

3.3.1 Pseudorandomness Under Selective Opening

To define pseudorandomness under selective opening, we shall consider two games. The first game is intended to capture that the evaluation outcome, i.e., the y term output by Eval, is pseudorandom even when $\mathcal{A}$ can selectively corrupt nodes and open the first component of the corrupted nodes' secret keys. The second game captures the notion that the proof σ does not reveal anything additional even under an adaptive adversary.

First Game: Pseudorandomness of the Evaluation Outcome. We consider a selective opening adversary $\mathcal{A}$ that interacts with a challenger denoted $\mathcal{C}$ in the following experiment $\mathsf{Expt}_b^{\mathcal{A}}(1^\lambda)$ indexed by the bit $b \in \{0,1\}$.

$\underline{\mathsf{Expt}_b^{\mathcal{A}}(1^\lambda)}$:

- First, the challenger $\mathcal{C}$ runs the $\mathsf{Gen}(1^\lambda)$ algorithm and remembers the secret key components $(s_1, \ldots, s_n)$ for later use. Note that $\mathcal{C}$ need not give public parameters to $\mathcal{A}$.
- Next, the adversary $\mathcal{A}$ can adaptively make queries of the following forms:
  - **Evaluate**: $\mathcal{A}$ submits a query (u, x), now $\mathcal{C}$ computes $y \leftarrow \mathsf{Eval}.E(\mathsf{crs}, s_u, x)$ and gives y to $\mathcal{A}$.
  - **Corrupt**: $\mathcal{A}$ specifies an index $u \in [n]$ to corrupt, and $\mathcal{C}$ parses $\mathsf{sk}_u := (s_u, \rho_u)$ and reveals s_u to $\mathcal{A}$.
  - **Challenge**: $\mathcal{A}$ specifies an index $u^* \in [n]$ and an input x. If $b = 0$, the challenger returns a completely random string of appropriate length. If $b = 1$, the challenger computes $y \leftarrow \mathsf{Eval}.E(\mathsf{crs}, s_{u^*}, x)$ and returns y to the adversary.

We say that $\mathcal{A}$ is compliant iff with probability 1, every challenge tuple (u^*, x) it submits satisfies the following: 1) $\mathcal{A}$ does not make a corruption query on u^* throughout the game; and 2) $\mathcal{A}$ does not make any evaluation query on the tuple (u^*, x).

If no efficient and compliant adversary can effectively distinguish $\mathsf{Expt}_0^{\mathcal{A}}(1^\lambda)$ and $\mathsf{Expt}_1^{\mathcal{A}}(1^\lambda)$, then we can be sure that the evaluation outcome of the VRF is pseudorandom even with an adaptive adversary.

Second Game: Zero-Knowledge of the Proofs. We also need to make sure that the proof part is zero-knowledge even w.r.t. an adaptive adversary. Therefore, we define another game below where the adversary $\mathcal{A}$ tries to distinguish whether it is playing in the real-world experiment or in the ideal-world experiment:

- *Real-world experiment* Real*:* In the real-world experiment, the challenger runs the $\mathsf{Gen}(1^\lambda)$ algorithm and gives the public parameters crs and all public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$ to $\mathcal{A}$, but keeps $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ to itself. Next, $\mathcal{A}$ can adaptively make the following queries:
  - **Evaluate:** $\mathcal{A}$ submits a query (u, x), now $\mathcal{C}$ computes $(y, \sigma) \leftarrow \mathsf{Eval}(\mathsf{crs}, \mathsf{sk}_u, x)$ and gives (y, σ) to $\mathcal{A}$.
  - **Corrupt:** $\mathcal{A}$ specifies an index $u \in [n]$ to corrupt. $\mathcal{C}$ reveals not only sk_u to $\mathcal{A}$, but also all the randomness used in the Eval algorithm for any earlier **Evaluate** query pertaining to u.
- *Ideal-world experiment* $\mathsf{Ideal}^{\mathcal{S}_0,\mathcal{S}_1,\mathcal{S}_2,\mathcal{S}_3}$*:* First, the challenger $\mathcal{C}$ runs a simulated setup algorithm

$$(s_1, \ldots, s_n) \leftarrow \mathcal{S}_0(1^\lambda);$$
$$(\mathsf{crs}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n, \tau) \leftarrow \mathcal{S}_1(1^\lambda);$$

  it gives the public parameters crs and all public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$ to $\mathcal{A}$, but keeps the trapdoor τ to itself.

  Next, $\mathcal{A}$ can adaptively make the following queries:
  - **Evaluate:** $\mathcal{A}$ submits a query (u, x), and now the simulator computes $y := \mathsf{Eval}.E(\mathsf{crs}, s_u, x)$, and $\sigma \leftarrow \mathcal{S}_2(\tau, \mathsf{pk}_u, x, y)$ and gives y, σ to $\mathcal{A}$.
  - **Corrupt:** $\mathcal{A}$ specifies an index $u \in [n]$ to corrupt. Let $\mathcal{I}$ denote the indices of the earlier **Evaluate** queries that correspond to the node $u \in [n]$; and moreover, for $i \in \mathcal{I}$, let the i-th query be of the form (u, x_i) and the result be of the form (y_i, σ_i).
    The challenger $\mathcal{C}$ calls $(\rho_u, \{\psi_i\}_{i\in\mathcal{I}}) \leftarrow \mathcal{S}_3(\tau, \mathsf{pk}_u, s_u, \{x_i, \sigma_i\}_{i\in\mathcal{I}})$, and returns the secret key $\mathsf{sk}_u := (s_u, \rho_u)$ as well as $\{\psi_i\}_{i\in\mathcal{I}}$ to $\mathcal{A}$.

Definition 3.3 (Pseudorandomness under selective opening)**.** We say that a VRF scheme satisfies pseudorandomness under selective opening iff:

1. for any compliant non-uniform $p.p.t.$ adversary $\mathcal{A}$, its views in $\mathsf{Expt}_0^{\mathcal{A}}(1^\lambda)$ and $\mathsf{Expt}_1^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.
2. there exist $p.p.t.$ simulators $(\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ such that the outcome of $\mathcal{S}_0(1^\lambda)$ is identically distributed as the $(s_0, \ldots, s_n)$ components generated by the real-world $\mathsf{Gen}(1^\lambda)$ algorithm, and moreover, $\mathcal{A}$'s views in the above Real and $\mathsf{Ideal}^{\mathcal{S}_0,\mathcal{S}_1,\mathcal{S}_2,\mathcal{S}_3}$ are computationally indistinguishable.

3.3.2 Unforgeability

We say that a VRF scheme satisfies unforgeability, if there exists a negligible function $\mathsf{negl}(\cdot)$ such that no non-uniform $p.p.t.$ adversary $\mathcal{A}$ can win the following game with more than $\mathsf{negl}(\lambda)$ probability:

- First, the challenger $\mathcal{C}$ runs the $\mathsf{Gen}(1^\lambda)$ algorithm and gives the public parameters crs and all public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$ to $\mathcal{A}$, but keeps $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ to itself.
- The adversary $\mathcal{A}$ can adaptively make the following queries:
  - **Evaluate:** $\mathcal{A}$ submits a query (u, x), now $\mathcal{C}$ computes $(y, \sigma) \leftarrow \mathsf{Eval}(\mathsf{crs}, \mathsf{sk}_u, x)$ and gives (y, σ) to $\mathcal{A}$.
  - **Corrupt:** $\mathcal{A}$ specifies an index $u \in [n]$ and $\mathcal{C}$ reveals sk_u to $\mathcal{A}$ as well as random coins used in earlier **Evaluate** queries pertaining to u.
- Finally, $\mathcal{A}$ outputs a tuple (u, x, y, σ). It is said to win the game if either $\mathsf{Vf}(\mathsf{crs}, \mathsf{pk}_u, x, y, \sigma) = 1$, but $y \neq y'$ where $(y', \_) := \mathsf{Eval}(\mathsf{crs}, \mathsf{sk}_u, x)$; or if u has not been corrupted before and $\mathcal{A}$ has not made any **Evaluate** query of the form (u, x).

In other words, we want that except with negligible probability, $\mathcal{A}$ cannot forge the VRF outcome and proof on behalf of any honest node on a point that has not been queried; furthermore, even for corrupted nodes, $\mathcal{A}$ cannot forge an VRF outcome and proof such that the evaluation outcome is different from the honest evaluation outcome.

Abraham et al. [ACD+19] proved the following theorem where the bilinear group assumptions needed are the same as those adopted by Groth et al. [GOS12].

Theorem 3.4 (Existence of adaptively secure VRFs [ACD+19])**.** *Assuming standard bilinear group assumptions and a trusted setup, we can construct a VRF scheme satisfying pseudorandomness under selective opening and unforgeability.*

4 Delayed-Exposure Message Distribution

4.1 Definitions

4.1.1 Syntax

We first introduce the syntax of the $\mathsf{Distribute}(1^\lambda, m_1, \ldots, m_n)$ protocol. At the beginning of the protocol, every node $u \in [n]$ is given a message $m_u \in \{0,1\}^\ell$ of length $\ell(\lambda, n)$ which is upper bounded by a fixed polynomial in λ and n. In the $\mathsf{Distribute}$ protocol, every node makes an attempt to multicast its message m_u to everyone else. At the end of R_{distr} number of rounds, everyone outputs $(m'_1, \ldots, m'_n)$ where $m'_u \in \{0,1\}^\ell \cup \{\perp\}$ denotes the message received from node $u \in [n]$, and $\perp$ indicates that nothing has been received from u.

In the following we will allow setup assumptions for constructing our $\mathsf{Distribute}$ protocol, specifically, we assume that the setup algorithm $\mathsf{Gen}(1^\lambda)$ outputs a common reference string denoted crs. Moreover, there is a public-key

infrastructure (PKI) later used for digital signatures. We assume that during the setup phase, we run the key generation algorithm of a digital signature scheme which outputs a public- and secret-key pair for every node $u \in [n]$, henceforth denoted vk_u and ssk_u, respectively. We assume that the crs and the PKI consisting of $\{\mathsf{vk}_u, \mathsf{ssk}_u\}_{u\in[n]}$ can be reused across multiple instances of the $\mathsf{Distribute}$ protocol.

4.1.2 Security

At the beginning of the $\mathsf{Distribute}$ protocol, either everyone is honest, or a subset of nodes have already been corrupted by the adversary $\mathcal{A}$. During the $\mathsf{Distribute}$ protocol, the adversary $\mathcal{A}$ can adaptively corrupt more nodes, and upon newly corrupting a node $u \in [n]$, the adversary $\mathcal{A}$ receives u's internal state.

Liveness. Liveness requires the following: let $\widetilde{H}$ denote the set of nodes that remain honest till the beginning of the second round of the $\mathsf{Distribute}$ protocol; except with negligible in λ probability, it holds that for every node $u \in \widetilde{H}$, all forever-honest nodes[1] output m_u as the message received from u.

Momentary Secrecy. Roughly speaking, we want that even when the adversary $\mathcal{A}$ may have unbounded polynomial parallelism, the honest nodes' messages remain secret to $\mathcal{A}$ till the beginning of the second round of $\mathsf{Distribute}$. Formally, we define the following $\mathsf{Expt}(1^\lambda, \{m_u^*\}_{u\in[n]})$ experiment.

Experiment $\mathsf{Expt}(1^\lambda, \{m_u^*\}_{u\in[n]})$**:** The experiment $\mathsf{Expt}(1^\lambda, \{m_u^*\}_{u\in[n]})$ is defined as follows:

- **Setup.** Run the honest setup algorithm which outputs a common reference string denoted crs and a key pair $(\mathsf{vk}_u, \mathsf{ssk}_u)$ for every $u \in [n]$. The crs and the public verification keys $\{\mathsf{vk}_u\}_{u\in[n]}$ are given to $\mathcal{A}$;
- **Query.** The query phase runs for an arbitrary polynomial amount of time. During this time, $\mathcal{A}$ may adaptively make the following queries where multiple sessions of the $\mathsf{Distribute}$ protocol are allowed to be initiated concurrently:
  - *Session.* $\mathcal{A}$ specifies a session identifier *sid*, as well as a set of input messages $\{m_u\}_{u\in H}$ where H denotes the so-far honest nodes. Now, the so-far honest nodes execute the honest $\mathsf{Distribute}$ protocol using the inputs $\{m_u\}_{u\in H}$ and session identifier *sid*, and interact with $\mathcal{A}$.
  - *Corrupt.* At any time, $\mathcal{A}$ specifies a new node $u \in [n]$ to corrupt, and at this moment ssk_u and all random coins used by node u so far in the protocol are given to $\mathcal{A}$;
- **Challenge.** Finally, $\mathcal{A}$ outputs `challenge` with a challenge session identifier sid^*: it is required that sid^* be a fresh one that has never been queried before. Let H^* denote the honest nodes at the moment. Now, compute the first-round messages denoted M^* that H^* would send in the real-world $\mathsf{Distribute}$ protocol

[1] Forever honest w.r.t. the $\mathsf{Distribute}$ protocol means that the node remains honest till the end of the protocol.

with the session identifier sid^*, and using the inputs $\{m_u^*\}_{u \in H^*}$. Output $\mathcal{A}$'s view[2] in the experiment as well as M^*.

We say that the Distribute protocol satisfies momentary secrecy iff for any non-uniform $p.p.t.$ parallel machine $\mathcal{A}$ and any non-uniform $p.p.t.$ $2\mathcal{T}_{\emptyset}$-bounded parallel distinguisher $\mathcal{D}$, there is a negligible function $\mathsf{negl}(\cdot)$ for the following to be true for any choice of λ, and any $\{m_u^*\}_{u \in [n]}$ and $\{\widetilde{m}_u^*\}_{u \in [n]}$,

$$\left|\Pr\left[\mathcal{D}(1^\lambda, \mathsf{Expt}(1^\lambda, \{m_u^*\}_{u \in [n]})) = 1\right] - \Pr\left[\mathcal{D}(1^\lambda, \mathsf{Expt}(1^\lambda, \{\widetilde{m}_u^*\}_{u \in [n]}))) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

4.2 Construction

Assumptions. In the construction below, we assume that there is a public-key infrastructure (PKI) available, and *nodes sign all messages* that they want to send. Only messages attached with valid signatures from the purported senders are considered valid, and all invalid messages are discarded. We also make an *implicit echoing* assumption: we assume that every honest node will echo every fresh message received to everyone, such that if a forever-honest node $u \in [n]$ observes some message at the beginning of round r, then every so-far honest node will have received it by the beginning of round $r + 1$.

NP Language. We will make use of a non-interactive zero-knowledge proof (NIZK) system that is secure against adaptive corruptions. The formal definition of such a NIZK system is given in Sect. A.2. We now describe the NP language used in our NIZK proofs. A statement is of the form $\mathsf{stmt} := (u, Z)$, and a witness is of the form $w := (m, \sigma, \rho)$. We assume that $(\lambda, \mathcal{T}_{\mathrm{solve}}, \mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ are global parameters and do not repeat it in the statement. A statement $\mathsf{stmt} := (u, Z)$ is in the language vouched for by a valid witness $w := (m, \sigma, \rho)$ iff there exists (m, σ, ρ) such that $Z = \mathsf{TLP.Gen}(1^\lambda, \mathcal{T}_{\mathrm{solve}}, (m, \sigma); \rho)$ and moreover $\Sigma.\mathsf{Vf}(\mathsf{vk}_u, m, \sigma) = 1$.

Protocol. Let $\mathsf{TLP} := (\mathsf{Gen}, \mathsf{Solve})$ denote a time-lock puzzle with hardness parameter ξ as defined in Sect. 3.2, and let $\mathcal{T}_{\emptyset}$ denote the duration of one synchronous round. Let $\mathsf{NIZK} := (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ denote a non-interactive zero-knowledge proof system as defined in Sect. A.2. Let $\Sigma := (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vf})$ denote a digital signature scheme. The Distribute protocol is described below.

Input: Each node $u \in [n]$ receives the input $m_u \in \{0,1\}^\ell$. Without loss of generality, henceforth we shall assume that the message m_u itself is tagged with the sender's identifier $u \in [n]$. Below, we may assume that we always prefix the message m_u with the string `inp`.

Setup: Run $\mathsf{crs}_{\mathrm{nizk}} \leftarrow \mathsf{NIZK.Gen}(1^\lambda)$ and publish $\mathsf{crs}_{\mathrm{nizk}}$. Recall that there is a PKI and nodes sign all messages they send, henceforth we use vk_u and ssk_u to denote the public- and secret-key of node u, respectively.

[2] Here, $\mathcal{A}$'s view may contain any output $\mathcal{A}$ has produced so far which might have taken an arbitrary polynomial time to compute prior to the start of the challenge phase.

Protocol:

1. **Initial round**: every node $u \in [n]$ does the following:
   - let $\sigma := \Sigma.\mathsf{Sign}(\mathsf{ssk}_u, m_u)$; call $Z_u \leftarrow \mathsf{TLP.Gen}(1^\lambda, \mathcal{T}_{\text{solve}}, (m_u, \sigma); \rho)$ where $\mathcal{T}_{\text{solve}} := 2\mathcal{T}_\emptyset/\xi$ and ρ explicitly denotes the randomness consumed by the $\mathsf{TLP.Gen}$ algorithm;
   - call $\pi_u \leftarrow \mathsf{NIZK.P}(\mathsf{crs}_{\text{nizk}}, (u, Z_u), (m_u, \sigma, \rho))$;
   - sign and multicast the tuple $(\mathsf{puz}, Z_u, \pi_u)$ to everyone.

   Henceforth, we assume that whenever an honest node receives a message of the form $(\mathsf{puz}, Z_u, \pi_u)$ signed by u, it calls $\mathsf{NIZK.V}(\mathsf{crs}_{\text{nizk}}, (u, Z_u), \pi_u)$ and if the verification fails, the message is discarded immediately without being processed. If the verification succeeds, the puzzle Z_u is considered as received and we say that it belongs to u.
2. **Solve phase**: Henceforth, every $\mathcal{T}_{\text{epoch}} := \mathcal{T}_{\text{solve}} \cdot \lceil \frac{2n}{h} \cdot \ln(\frac{16n}{h}) \cdot \log^2 \lambda \cdot (\log n + 3) \rceil + 1$ rounds is called an *epoch*: round 1 is the beginning of the first epoch; round $\mathcal{T}_{\text{epoch}} + 1$ is the beginning of the second epoch, and so on.
   Repeat the following for a total of $E = \lceil \log_2 n \rceil + 1$ epochs.
   - At the beginning of each epoch, let S denote the set of all puzzles received so far and belonging to *active* nodes. For $Z \in S$, define the puzzle's *age* $\alpha(Z)$ as follows: $\alpha(Z) := \lceil \frac{r - r'}{\mathcal{T}_{\text{epoch}}} \rceil$ where r denotes the beginning round of the epoch and $r' \le r$ denotes the first round in which Z was observed.
   - For each $Z \in S$ sequentially, perform the following: flip a random coin that comes up heads with probability $p := \min\left(\frac{2^{\alpha(Z)} \cdot \ln(16n/h)}{h}, 1\right)$; if the coin comes up heads, then solve the puzzle Z by calling $(m, \sigma) \leftarrow \mathsf{TLP.Solve}(Z)$. Once solved, multicast the solution (m, σ) to everyone[a].

   Output: at any time, upon observing a tuple (m, σ) where σ is a valid signature on m from the purported sender (henceforth denoted $v \in [n]$), if no message from v has been output yet, output m as the message received from v and mark v as *inactive*. At the end of the protocol, if no message from some $v \in [n]$ has been output, then we output a canonical message $\perp$ as the message from v.
   Detect equivocation: at any time, if multiple puzzles have been received from the same node $v \in [n]$, mark v as *inactive*[b].

[a] We may assume that if more than $\frac{2n}{h} \cdot \ln(\frac{16n}{h}) \cdot \log^2 \lambda \cdot (\log n + 3)$ number of puzzles are chosen to be solved in some epoch, the node simply aborts outputting $\mathsf{failure}$ — we will show in the proof of Lemma 4.3 later that this does not happen except with negligible probability.

[b] Both the **Output** and **Equivocation** entry points are processed at the beginning of every round before all other actions of the round.

Clearly, the total round complexity of the Distribute protocol is

$$(\lceil \log_2 n \rceil + 1) \cdot \frac{\mathcal{T}_{\text{solve}}}{\mathcal{T}_{\emptyset}} \cdot \left(\frac{2n}{h} \cdot \ln(\frac{16n}{h}) \cdot \log^2 \lambda \cdot (\log n + 3) \rceil + 1 \right) + 1$$

Assume that n is polynomially bounded in λ, then the round complexity is upper bounded by $O(\frac{n}{\xi \cdot h}) \cdot \mathsf{poly} \log \lambda$.

4.3 Proofs: Liveness

Henceforth, we use the term "m is *in honest view*" to mean that some forever-honest node has seen m.

Fact 4.1. *If some forever-honest node observes a puzzle Z at the beginning of epoch e, then all so-far honest nodes will have observed Z by the beginning of epoch $e + 1$.*

Proof. Follows directly from the implicit echoing assumption, i.e., honest nodes echo every fresh message they see. □

Fact 4.2. *Assume that the* NIZK *scheme satisfies perfect knowledge extraction, the* TLP *scheme satisfies correctness. Then, except with negligible probability, the following must hold: if any forever-honest node solves a puzzle Z belonging to $v \in [n]$ by the beginning of the last round of epoch e, then no puzzle from v will still be active in any so-far honest node's view at the beginning of epoch $e + 1$.*

Proof. If the NIZK satisfies perfect knowledge extraction, then it must be that the solved solution is a (m, σ) pair such that σ is a valid signature from v on m. Since an honest node has solved the puzzle and found the solution by the beginning of the last round of epoch e, it will multicast (m, σ) to everyone in the last round of epoch e, and by the beginning of epoch $e + 1$, every so-far honest node will have observed (m, σ) and will have marked v as inactive. □

Given Fact 4.1, we know that honest nodes' perception of a puzzle's age can differ by at most 1. We say that a puzzle Z's *minimum age* is $\alpha \geq 0$ in epoch e, iff all forever-honest nodes have observed it by the beginning of epoch $e - \alpha$, but at least one forever-honest node has not observed it by the beginning of epoch $e - \alpha - 1$.

Henceforth, if at the beginning of some epoch e, a so-far honest has seen a puzzle belonging to an active node, then the puzzle is said to be *active*. Due to the equivocation detection rule, it must be that from each active node, at most one active puzzle has been seen.

Lemma 4.3. *Let $\alpha \geq 0$ and $n \geq \log^2 \lambda$. Except with negligible in λ probability, the following holds: at the beginning of every epoch, there can be at most $n/2^{\alpha-1}$ active puzzles belonging to distinct nodes, and with minimum age α, in honest view.*

Proof. We first state some simplifying assumptions that can be made without loss of generality. We may assume that honest nodes use a puzzle's minimum age to determine the probability p with which a puzzle is selected to be solved. Note that in the real-world protocol, a node does not necessarily know the minimum age of the puzzle, but we may assume it for proving this lemma since making p smaller will only increase the probability of the bad event stated in the lemma that we care about bounding. Furthermore, let us first assume that any forever-honest node has enough time to solve all puzzles it chooses to solve during any epoch e, and not only so, they can be solved by the beginning of the last round of the epoch e—later we will show that indeed this is the case except with negligible probability.

For $\alpha = 0$, the lemma trivially holds. Henceforth, we may assume that $\alpha \geq 1$. Fix any epoch e, and let $S_{\alpha-1}$ denote all active puzzles whose minimum age is $\alpha-1 \geq 0$ at the beginning of epoch e. This means that all so-far honest nodes will choose to solve any puzzle in $S_{\alpha-1}$ with probability $p = \min(\frac{2^{\alpha-1}\cdot\ln(16n/h)}{h}, 1)$ in epoch e. We would like to upper bound the probability that at least $n/2^{\alpha}$ puzzles in $S_{\alpha-1}$ are not selected by any forever-honest node in epoch e. Due to Fact 4.2, if $\frac{2^{\alpha-1}\cdot\ln(16n/h)}{h} \geq 1$, then there cannot be any active puzzles of minimum age α left in any honest node's view at the beginning of the next epoch. Henceforth, we may also assume that $p = \frac{2^{\alpha-1}\cdot\ln(16n/h)}{h} < 1$.

Consider a fixed honest node u and an active puzzle from a fixed node v: the probability that u does not select an active puzzle from v is at most $1-p$. The probability that any fixed set of h forever-honest nodes (denoted W) all do not select an active puzzle from v is $(1-p)^h$. For any fixed set of $n/2^{\alpha-1}$ nodes denoted Γ that has a puzzle of minimum age $\alpha-1$ in honest view in epoch e, the probability that a fixed set of h forever-honest nodes' puzzle choices do not intersect with Γ is at most $(1-p)^{h\cdot n/2^{\alpha-1}}$.

The probability that there exists a choice for the set W (consisting of h forever-honest nodes), and a set $\Gamma \subset [n]$ of size $n/2^{\alpha-1}$ (who have a puzzle of age $\alpha-1$ in honest view in epoch e), such that W's puzzle choices do not intersect with Γ is upper bounded by the following expression:

$$
\begin{aligned}
&(1-p)^{h\cdot n/2^{\alpha-1}} \cdot \binom{n}{h} \cdot \binom{n}{n/2^{\alpha-1}} \\
&\leq \left(1 - \frac{2^{\alpha-1}\cdot\ln(16n/h)}{h}\right)^{hn/2^{\alpha-1}} \cdot \binom{n}{h} \cdot \binom{n}{n/2^{\alpha-1}} \\
&= \left(1 - \frac{2^{\alpha-1}\cdot\ln(16n/h)}{h}\right)^{\frac{h}{2^{\alpha-1}\cdot\ln(\frac{16n}{h})}\cdot\ln(\frac{16n}{h})\cdot n} \cdot \binom{n}{h} \cdot \binom{n}{n/2^{\alpha-1}} \\
&\leq \exp\left(-\ln\left(\frac{16n}{h}\right)\cdot n\right) \cdot \binom{n}{h} \cdot \binom{n}{n/2^{\alpha-1}} \\
&= \left(\frac{h}{16n}\right)^{n} \cdot \binom{n}{h} \cdot \binom{n}{n/2^{\alpha-1}} \qquad (*)
\end{aligned}
$$

$$
\begin{aligned}
&= \left(\frac{h}{16n}\right)^n \cdot \binom{n}{h} \cdot \binom{n}{n - n/2^{\alpha-1}} \\
&\leq \left(\frac{h}{16n}\right)^n \cdot \left(\frac{en}{h}\right)^h \cdot \left(\frac{en}{n(1-1/2^{\alpha-1})}\right)^{n(1-1/2^{\alpha-1})} \\
&\leq \left(\frac{h}{16n}\right)^n \cdot \left(\frac{en}{h}\right)^n \cdot \left(\frac{en}{n(1-1/2^{\alpha-1})}\right)^n \\
&= \left(\frac{h}{16n} \cdot \frac{en}{h} \cdot \frac{en}{n(1-1/2^{\alpha-1})}\right)^n \leq \exp(-\Theta(n))
\end{aligned}
$$

In the above derivation, if $\alpha = 1$, then the last term $\binom{n}{n/2^{\alpha-1}}$ in the expression $(*)$ is equal to 1. Therefore, in the derivation steps after the expression $(*)$, we can simply assume that $\alpha > 1$ which only makes the expression $\binom{n}{n/2^{\alpha-1}}$ larger. [To reviewer: We expanded the derivation to make it more detailed.]

So far, we have assumed that if a forever-honest node selects some puzzle to solve in some epoch e, it will actually have enough time to solve the puzzle by the beginning of the last round of epoch e. We now show that the allotted epoch duration $\mathcal{T}_{\text{epoch}} := \mathcal{T}_{\text{solve}} \cdot \lceil \frac{2n}{h} \cdot \ln(\frac{16n}{h}) \cdot \log^2 \lambda \cdot (\log n + 3) \rceil + 1$ is indeed long enough to meet this requirement except with negligible probability. Basically, if in epoch e, the number of puzzles of minimum age α left in honest view is at most $n/2^{\alpha-1}$, and an honest node selects each puzzle with probability $p = \min(\frac{2^{\alpha-1} \cdot \ln(16n/h)}{h}, 1)$, we can bound the total number of puzzles of minimum age α an honest node selects to solve with the following two cases:

- Case 1: if $\frac{2^{\alpha-1} \cdot \ln(16n/h)}{h} \geq 1$, then $2^{\alpha-1} \geq h/\ln(16n/h)$. The total number of puzzles of minimum age α selected to solve is upper bounded by $n/2^{\alpha-1} \leq \frac{n}{h} \cdot \ln(16n/h)$.
- Case 2: if $\frac{2^{\alpha-1} \cdot \ln(16n/h)}{h} < 1$, then the expected number of puzzles of minimum age α selected to solve is upper bounded by

  $$\frac{n}{2^{\alpha-1}} \cdot \frac{2^{\alpha-1} \cdot \ln(16n/h)}{h} = \frac{n}{h} \cdot \ln(16n/h).$$

  By the Chernoff bound, the probability that the number of puzzles of minimum age α selected to solve is more than

  $$\frac{n}{h} \cdot \ln(16n/h) + \sqrt{\frac{n}{h} \cdot \ln(16n/h) \cdot \log^2 \lambda} \leq \frac{n}{h} \cdot \ln(16n/h) \cdot \log^2 \lambda \qquad (**)$$

  is upper bounded by $\exp(-\Omega(\log^4 \lambda))$.

Recall that the number of ages is upper bounded by the number of epochs, that is $\lceil \log_2 n \rceil + 1$. Now, taking a union bound over all possible ages, except with $\exp(-\Omega(\log^4 \lambda))$ probability, the total number of puzzles an honest node chooses to solve in an epoch is upper bounded by

$$\frac{2n}{h} \cdot \ln(\frac{16n}{h}) \cdot \log^2 \lambda \cdot \lceil \log_2 n \rceil \leq \frac{2n}{h} \cdot \ln(\frac{16n}{h}) \cdot \log^2 \lambda \cdot (\log n + 3)$$

Finally, taking a union bound over the number of epochs which is polynomially bounded in λ, we have that except with $\exp(-\Omega(\log^4 \lambda))$ probability, the above bad event will never happen throughout all epochs.

[To reviewer: We added more steps of derivation in the above calculations.]

Thus, the allotted epoch duration $\mathcal{T}_{\text{epoch}} := \mathcal{T}_{\text{solve}} \cdot \lceil \frac{2n}{h} \cdot \ln(\frac{16n}{h}) \cdot \log^2 \lambda \cdot (\log n + 3) \rceil + 1$ is sufficiently long such that except with negligible in λ probability, an honest node has time to solve all puzzles it chooses in every epoch. □

Remark 2. The expression $(**)$ is by no means the tightest possible bound; but it is outside the scope of this paper to understand what the best possible constant c is in the $O(\log^c(\lambda, n))$ round complexity bound. With our current techniques, we only know how to achieve poly-logarithmic round complexity in the strongly adaptive setting under corrupt majority. It is an exciting open question whether we can improve the result to, say, expected constant rounds, or prove impossibility.

[To reviewer: We added an explanation why we did not focus on calculating the tightest expression.]

Theorem 4.4 (Liveness)**.** *Assume that $n \geq \log^2 \lambda$, that the* NIZK *scheme satisfies soundness, the* TLP *scheme satisfies correctness, and the signature scheme* Σ *satisfies existential unforgeability under chosen-message attack. Then, the above* Distribute *protocol satisfies liveness.*

Proof. Let $\widetilde{H}$ denote the set of nodes that remain honest till the beginning of the second round of the Distribute protocol. For every $u \in \widetilde{H}$, every so-far honest node will have received an honestly generated puzzle Z_u from u at the beginning of the second round, i.e., the beginning of the first epoch of the **Solve** phase. At the beginning of the next round after the final epoch E, Z_u, if still active, would have age E in every honest node's view, i.e., its minimum age is E. Due to Lemma 4.3, except with negligible probability, the number of active puzzles from nodes in $\widetilde{H}$ is then upper bounded by $n/2^{E-1} < 1$. Now, since honest nodes do not double sign puzzles, except with negligible probability, it must be that every forever-honest node has output a message for everyone in $\widetilde{H}$. □

4.4 Proofs: Momentary Secrecy

Experiment $\mathsf{Hyb}(1^\lambda, \{m_u^*\}_{u\in[n]})$**.** The hybrid experiment $\mathsf{Hyb}(1^\lambda, \{m_u^*\}_{u\in[n]})$ is defined almost identically as $\mathsf{Expt}^{\mathcal{A}}(1^\lambda, \{m_u^*\}_{u\in[n]})$, except with the following modifications:

- Run the simulated NIZK setup algorithm $\mathsf{Gen}_0(1^\lambda)$ which outputs a crs and a trapdoor τ;
- Whenever an honest node u is supposed to compute a NIZK proof by calling

$$\pi_u \leftarrow \mathsf{NIZK.P}(\mathsf{crs}_{\text{nizk}}, \mathsf{stmt} = (u, Z_u), w = (m_u, \sigma, \rho); \mathsf{coins}),$$

now we instead call the simulated prover

$$\pi_u \leftarrow \mathsf{NIZK.P}_0(\mathsf{crs}_{\mathrm{nizk}}, \tau, \mathsf{stmt} = (u, Z_u); \mathsf{coins}).$$

Note that P_0 uses the trapdoor τ but does not use the witness to output a simulated proof;

- Whenever a node u newly becomes corrupt and the experiment needs to explain the random coins used earlier by u, it calls the NIZK's Explain algorithm, that is,

$$\mathsf{NIZKcoins} \leftarrow \mathsf{NIZK.Explain}\,(\mathsf{crs}_{\mathrm{nizk}}, \tau, \mathsf{stmt} = (u, Z_u), w = (m_u, \sigma, \rho); \mathsf{coins})$$

to output an explanation of the coins used in generating the earlier NIZK proofs. These coins are returned to $\mathcal{A}$ along with all other random coins consumed by the newly corrupted node u earlier.
[To reviewer: Indeed, there is no VRF here, the VRF was an editorial typo due to historical reasons, since we changed our protocol completely at some point. Thank you for spotting this. We also polished the entire paragraph to make it more clear.]

For details of the NIZK syntax and security definitions (including the NIZK.Explain algorithm which is part of the NIZK's security definition), please refer to Appendix A.2.

Claim 4.5. *Assume that the* NIZK *scheme satisfies non-erasure computational zero-knowledge. Then, the outputs of the experiments* $\mathsf{Expt}(1^\lambda, \{m_u^*\}_{u\in[n]})$ *and* $\mathsf{Hyb}(1^\lambda, \{m_u^*\}_{u\in[n]})$ *are computationally indistinguishable.*

Proof. Follows directly from the computational zero-knowledge of the NIZK system. □

Experiment $\mathsf{Ideal}(1^\lambda)$**.** The ideal experiment $\mathsf{Ideal}(1^\lambda)$ is almost identical to $\mathsf{Hyb}(1^\lambda, \_)$, except with the following modification:

- At the beginning of the challenge phase, let H^* denote the so-far honest nodes. We compute the first-round message $(\mathsf{puz}, Z_u, \pi_u)$ for every $u \in H^*$ as below: call $Z_u \leftarrow \mathsf{TLP.Gen}(1^\lambda, \mathcal{T}_{\mathrm{solve}}, (0, \sigma))$ where $\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{ssk}_u, 0)$. Further, call the NIZK's simulated prover $\widetilde{\mathsf{P}}$ which uses τ but not the witness to generate a simulated proof π_u.

Claim 4.6. *Assume that the* TLP *scheme satisfies* ξ*-hardness. Then, for any* $\{m_u^*\}_{u\in[n]}$*, no non-uniform parallel* $2\mathcal{T}_\emptyset$*-bounded distinguisher* $\mathcal{D}$ *can distinguish the outputs of the experiments* $\mathsf{Ideal}(1^\lambda)$ *and* $\mathsf{Hyb}(1^\lambda, \{m_u^*\}_{u\in[n]})$ *except with negligible probability.*

Proof. We can consider a sequence of hybrids for $i \in [0, h]$, such that in the i-th hybrid, during the challenge session, the first $\min(i, |H^*|)$ nodes in H^* (by lexicographical ordering) will use the input $(0, \sigma)$ where $\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{ssk}_u, 0)$

to compute a puzzle. If there exists a non-uniform parallel $2\mathcal{T}_{\emptyset}$-bounded distinguisher $\mathcal{D}$ that can distinguish the outputs of the experiments $\mathsf{Ideal}(1^\lambda)$ and $\mathsf{Hyb}(1^\lambda, \{m_u^*\}_{u\in[n]})$ with more than negligible probability, by the hybrid argument, there must exist a pair of adjacent hybrids indexed j and $j+1$ that $\mathcal{D}$ can distinguish with non-negligible probability.

We can construct a non-uniform $p.p.t.$ parallel machine $\mathcal{B}$ which breaks the ξ-hardness of the TLP scheme. $\mathcal{B}$ simulates the experiment for $\mathcal{A}$ and let u be the $(j+1)$-th node in H^* at the beginning of the challenge session. At the beginning of the challenge session, for every v that is among the first j nodes in H^*, $\mathcal{B}$ computes their puzzles using the input $(0, \sigma)$ where $\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{ssk}_v, 0)$; and for everyone else in H^* that is not among the first $j+1$ nodes, $\mathcal{B}$ will compute their puzzles using the input (m_v^*, σ') where $\sigma' \leftarrow \Sigma.\mathsf{Sign}(\mathsf{ssk}_v, m_v^*)$.

After this computation is done, $\mathcal{B}$ now computes the first-round message for the $(j+1)$-th node in H^*. To do so, $\mathcal{B}$ interacts with a TLP challenger which either returns a puzzle either for the string $(0, \sigma)$ where $\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{ssk}_u, 0)$, or for the string (m_u^*, σ') where $\sigma' \leftarrow \Sigma.\mathsf{Sign}(\mathsf{ssk}_u, m_u^*)$. This answer will be used as the puzzle for the $(j+1)$-th node in H^*.

At this moment, $\mathcal{B}$ gives the view of $\mathcal{A}$, including all random coins consumed by $\mathcal{A}$ and all outputs of $\mathcal{A}$ so far, as well as the first-round messages of H^* in the challenge session to the distinguisher $\mathcal{D}$, and in at most $2\mathcal{T}_{\emptyset}$ time, it outputs the same answer as $\mathcal{D}$. □

At this moment, by the hybrid argument, we have that no non-uniform $p.p.t.$ $2\mathcal{T}_{\emptyset}$-bounded parallel machine $\mathcal{D}$ can distinguish the outputs of $\mathsf{Expt}^{\mathcal{A}}(1^\lambda, \{m_u^*\}_{u\in[n]})$ and $\mathsf{Ideal}(1^\lambda)$ with more than negligible probability. By a symmetric argument, the same holds for $\mathsf{Expt}^{\mathcal{A}}(1^\lambda, \{\widetilde{m}_u^*\}_{u\in[n]})$ and $\mathsf{Ideal}(1^\lambda)$. Thus, we can conclude that no non-uniform $p.p.t.$ $2\mathcal{T}_{\emptyset}$-bounded parallel machine $\mathcal{D}$ can distinguish the outputs of $\mathsf{Expt}^{\mathcal{A}}(1^\lambda, \{m_u^*\}_{u\in[n]})$ and $\mathsf{Expt}^{\mathcal{A}}(1^\lambda, \{\widetilde{m}_u^*\}_{u\in[n]})$ with more than negligible probability.

5 Byzantine Broadcast Protocol

5.1 Protocol

Without loss of generality, we may assume that $u = 1$ is the designated sender for the Byzantine Broadcast. Our protocol will make use of a VRF scheme which is defined in Sect. 3.3, and will rely on the $\mathsf{Distribute}$ protocol that is defined and constructed in Sect. 4.

Vote. A vote from a node $u \in [n]$ for the bit $b \in \{0,1\}$ is a tuple of the form $(\mathtt{vote}, b, u, D, \sigma)$ such that $\mathsf{VRF.Vf}(\mathsf{crs}_{\mathrm{vrf}}, \mathsf{pk}_u, b, D, \sigma) = 1$, and moreover, it must be that either $D < D_p$ or $u = 1$. Here D_p denotes a *difficulty parameter* whose choice will be specified shortly.

Batch of Votes. An r-batch of votes for a bit $b \in \{0,1\}$ is a collection of valid votes from r distinct nodes, and moreover, it must be that one of these votes comes from the designated sender.

Protocol. Our Byzantine Broadcast protocol is described below. Recall that $h = n - f$ denotes the number of honest nodes.

Initially, every node u's $\mathsf{Extracted}_u$ set is set to $\emptyset$. The designated sender $u = 1$ computes and records a vote for its input bit b by computing $(D, \sigma) \leftarrow \mathsf{VRF.Eval}(\mathsf{crs}_{\mathrm{vrf}}, \mathsf{sk}_1, b)$.

Parameters. Let ℓ be the length of the first term of the VRF's evaluation outcome. The difficulty parameter D_p is set such that the probability that a random string of length ℓ is less than D_p with probability $p \in (\frac{\log^2 \lambda}{h}, \frac{3\log^2 \lambda}{h}) \cap (0, 1)$. The number of phases $R := 6\log^2 \lambda \cdot \frac{n}{h}$.

Setup. We use two instances of the Distribute protocol, denoted $\mathsf{Distribute}^0$ and $\mathsf{Distribute}^1$ respectively, and each instance is used by nodes to distribute batches of votes for the bit 0 and 1, respectively. For $b \in \{0, 1\}$, call the setup of Distribute which outputs $(\mathsf{crs}^b_{\mathrm{distr}}, \{\mathsf{vk}^b_u, \mathsf{ssk}^b_u\}_{u\in[n]})$. Call $(\mathsf{crs}_{\mathrm{vrf}}, \{\mathsf{pk}_u, \mathsf{sk}_u\}_{u\in[n]}) \leftarrow \mathsf{VRF.Gen}(1^\lambda)$. Now, publish the public parameters $(\mathsf{crs}^0_{\mathrm{distr}}, \mathsf{crs}^1_{\mathrm{distr}}, \mathsf{crs}_{\mathrm{vrf}})$ and give each node u the secret keys ssk^0_u, ssk^1_u, and sk_u.

Phase $r \in [1 \ldots R]$**.** Each phase consists of $R_{\mathrm{distr}} + 1$ rounds where R_{distr} denotes the round complexity of the Distribute protocol.

1. In the first round, every node u performs the following: for each bit $b \in \{0, 1\}$, if node u has seen a valid r-batch of votes for b and $b \notin \mathsf{Extracted}_u$, then it multicasts any such r-batch for b to everyone, and sets $\mathsf{Extracted}_u \leftarrow \mathsf{Extracted}_u \cup \{b\}$.
2. The next step lasts for R_{distr} rounds. Each node $u \neq 1$ does the following: for each bit $b \in \{0, 1\}$, it invokes a new session of the $\mathsf{Distribute}^b$ protocol with a new session identifier r to distribute either an $(r + 1)$-batch of votes or a dummy message:
   - If it has recorded a valid r-batch of votes for b and node u has never computed a vote for b before, then it attempts to vote for b by computing $(D, \sigma) \leftarrow \mathsf{VRF.Eval}(\mathsf{crs}_{\mathrm{vrf}}, \mathsf{sk}_u, b)$. If $D < D_p$, then execute the following:
     - let $\mathsf{Extracted}_u \leftarrow \mathsf{Extracted}_u \cup \{b\}$;
     - invoke $\mathsf{Distribute}^b$ to distribute a valid $(r + 1)$-batch of votes for b, possibly by adding its own vote $(\mathtt{vote}, b, u, D, \sigma)$.
   - Else if the node did not decide to distribute an $(r+1)$-batch of votes for b in the above, then invoke $\mathsf{Distribute}^b$ to distribute a dummy message $\perp$ which is encoded as a string of the same length as an $(r + 1)$-batch of votes for b.

At any time, if any valid vote is received over the network or output by the $\mathsf{Distribute}^0$ or $\mathsf{Distribute}^1$ protocols, the vote is then recorded by the node.

Output. At the end of the R phases, if $|\mathsf{Extracted}_u| = 1$ node u outputs the unique bit in $\mathsf{Extracted}_u$; else output the default bit 0.

Round Complexity. The total round complexity of the above protocol is upper bounded by $R \cdot R_{\text{distr}} = \left(\frac{n}{h}\right)^2 \cdot \frac{1}{\xi} \cdot \mathsf{poly} \log \lambda$. As a special case, in the case 99% or any arbitrarily large constant fraction of nodes are corrupt, and assuming that the hardness parameter ξ is a constant, the round complexity of the protocol is $\mathsf{poly} \log \lambda$.

6 Proofs for Our Byzantine Broadcast Protocol

6.1 Additional Terminology

For convenience, we will use the following terminology.

- We say that an execution satisfies consistency for the bit $b \in \{0, 1\}$, iff the following holds: if some forever-honest node u has b in its $\mathsf{Extracted}_u$ set at the end, then every forever-honest node v have b in its $\mathsf{Extracted}_v$ set at the end, too. To show consistency, we just have to prove that except with negligible probability over the choice of the randomized execution, consistency holds for $b = 0$ as well as $b = 1$.
- For convenience, we say that a node u *mines* a vote for $b \in \{0, 1\}$ if it calls $(D, \sigma) \leftarrow \mathsf{VRF.Eval}(\mathsf{crs}_{\text{vrf}}, \mathsf{sk}_u, b)$ to attempt to compute a vote for b, and recall that whether a mining attempt is successful depends on whether the outcome D is less than the difficulty parameter D_p. All honest mining attempts are made in the second round of some phase, i.e., the first round of the $\mathsf{Distribute}$ protocol of that phase.
- We say that a node $u \in [n]$ is in the 0-committee iff the first term in the output of $\mathsf{VRF.Eval}(\mathsf{crs}_{\text{vrf}}, \mathsf{sk}_u, 0)$ is smaller than D_p. Members of the 1-committee is similarly defined.

We will consider two types of bad events. We will later prove that if, except with negligible probability, neither type of bad event happens for both bits, then consistency is respected except with negligible probability.

- *Type A bad event for the bit b*: In the second round of some phase r, all so-far honest nodes have made attempts to mine a vote for b (in the second round of some phase $r' \leq r$); and yet, the adversary $\mathcal{A}$ manages to corrupt every member of the b-committee either before it even made a mining attempt for b, or by the beginning of the third round of the phase in which it makes a mining attempt for b.
- *Type B bad event for the bit b*: Either at least R nodes are members of the b-committee, or no node is a member of the b-committee.

6.2 Proof Overview: Challenges and Intuition

We would like to use probablistic reasoning to argue that the above two types of bad events do not happen except with negligible probability. The probabilistic reasoning could be accomplished using standard measure concentration bounds if

all the cryptography we employ were "ideal". Unfortunately the cryptography we employ is far from ideal. One problem we encounter is that our delayed-exposure distribution primitive $\mathsf{Distribute}$ guarantees secrecy against only an adversary who is restricted to run in a small number of parallel steps. An adversary who can take more parallel steps can completely break the secrecy of $\mathsf{Distribute}$ by solving the time-lock puzzles. For our final protocol, of course, we want to prove security against any *parallel p.p.t. adversary* who is allowed to take an *unbounded* number of parallel steps.

Partly due to this reason, the most natural proof strategy completely fails: we are not able describe an ideal protocol (without cryptography), and show that the real-world protocol *securely emulates* the ideal protocol by a standard, *simulation-based* notion. What comes to the rescue is that we only need to prove that certain security properties hold in the real-world protocol; and proving these properties eventually boils down to showing that certain bad events (as defined above) happen with negligible probability. Therefore, instead of proving that the real-world protocol *securely emulates* some ideal protocol, our strategy is to prove that the probability of bad events is not higher in the real-world protocol than in the ideal protocol (barring negligible differences). To do this, we will define a polynomially long sequence of hybrids, starting from the real-world protocol, all the way to the ideal protocol which does not have any cryptography: we will prove that for every pair of adjacent hybrids, the probability of bad events in the former is not higher than the probability of bad events in the latter (barring negligible differences).

We now elaborate on our blueprint—below in our proof overview, we mainly focus on how we bound the probability of Type-A bad events since this is the most technical part of the proof. First, we make several modifications to the real-world protocol and obtain a hybrid called $\mathsf{Hyb}'_\star$. $\mathsf{Hyb}'_\star$ *is no longer a consensus protocol, it should simply be viewed as a game in which the adversary* $\mathcal{A}$*is trying to cause Type-A bad events to happen.* The modifications we made ensure that the probability of Type-A bad events can only increase in $\mathsf{Hyb}'_\star$ in comparison with the real-world protocol. Importantly, in $\mathsf{Hyb}'_\star$, we introduce a *final guessing phase*: if at the end of the protocol, $\mathcal{A}$ has corrupted $f' < f$ number of nodes, i.e., it has more corruption budget left, we give $\mathcal{A}$ an extra opportunity to guess who, among the remaining honest nodes that have not made a mining attempt for b, are members of the b-committee. If $\mathcal{A}$ can correctly guess all remaining honest b-committee members in at most $f - f'$ tries, we also declare that $\mathcal{A}$ wins, i.e., a Type-A bad event has happened.

At this moment, it is not clear why we introduce the final guessing phase yet. This will become clear in the next hybrid $\mathsf{Hyb}_\star$. In $\mathsf{Hyb}_\star$, we modify the final guessing phase, such that the remaining honest nodes who have not made a mining attempt for b yet would use true random coins rather than VRFs to determine if they are a member of the b-committee. In this way, the game becomes ideal (i.e., without cryptography) after the final guessing phase starts. Partly relying on the security of VRF, one can show that any parallel $p.p.t.$ $\mathcal{A}$

cannot cause Type-A bad events to happen more often in $\mathsf{Hyb}'_\star$ than in $\mathsf{Hyb}_\star$ (barring negligible differences).

Now, in the remainder of the proof, our idea is to start from the end of the experiment, and make each phase "ideal" one by one. In other words, in each hybrid, we will make the final guessing phase start one phase earlier, until at the very end, the final guessing phase starts upfront and therefore the whole game becomes ideal (i.e., no cryptography). In the process, we make sure that $\mathcal{A}$'s probability of causing bad events does not decrease (barring negligible differences).

At this moment, it is a good time to revisit how we can overcome the aforementioned problem where the overall adversary is a parallel machine running in unbounded parallel steps but our Distribute primitive only gives secrecy against an adversary who is restricted to run in a small number of parallel steps. With the above proof strategy, informally speaking, at some point, we need to compare the probability of Type-A bad event in the following two adjacent hybrids—henceforth let r^* be the phase immediately preceding the final guessing phase:

1. in the first hybrid, in phase r^*, honest nodes run the real Distribute protocol using real inputs;
2. in the second hybrid, in phase r^*, honest nodes run the Distribute protocol using input $\mathbf{0}$.

In both of these hybrids, the adversary $\mathcal{A}$ can win the game if it either wins in the final guessing phase, or if $\mathcal{A}$ can guess, by the beginning of the third round of phase r^*, which honest nodes successfully made mining attempts for b in phase r^*. To succeed in the latter, $\mathcal{A}$ might try to gain some leverage by attacking the Distribute protocol of phase r^*, but because of the short-fuse deadline $\mathcal{A}$ must make the guess by, the Distribute protocol in phase r^* is unhelpful to $\mathcal{A}$ due to the momentary secrecy property. Even though after phase r^*, $\mathcal{A}$ may completely break the secrecy of the Distribute protocol of phase r^*, recall that the games becomes ideal immediately after phase r^*. Therefore, breaking the secrecy of the phase-r^* Distribute protocol no longer helps $\mathcal{A}$ after phase r^*.

Last but not the least, besides the aforementioned technicalities, yet another is that $\mathcal{A}$ is adaptive, and we need to handle the adaptivity with particular care in our proofs. Now, without further ado, we present our formal proofs.

6.3 Bounding the Probability of Type-A Bad Events

Let Real denote an execution of the real-world protocol in which the adversary $\mathcal{A}$'s goal is to cause a Type-A bad event to happen. In the remainder of this section, we will consider a sequence of hybrid experiments starting with Real such that for each pair of adjacent experiments, the probability of a Type-A bad event in the latter is an upper bound of the probability of a Type-A bad event in the former (ignoring negligible differences). In the end, we will upper bound the probability of a Type-A bad event in the final experiment called Hyb_1. Hyb_1 essentially gets rid of the cryptography and therefore we can upper bound the

probability of a Type-A bad event in Hyb_1 with a simple information-theoretic, probabilistic argument.

In the following we fix an arbitrary $b \in \{0,1\}$, and we care about bounding Type-A bad events for the bit b. Henceforth whenever we say Type-A bad event, it means a Type-A bad event for the bit b. Also, recall that we use the notation $f = n - h$ to denote the maximum number of corruptions allowed.

6.3.1 Experiment $\mathsf{Hyb}'_\star$

Since we only care about bounding Type-A bad events for the bit b, we make some simplifications to the protocol without decreasing the probability of a Type-A bad event. We therefore define a hybrid experiment $\mathsf{Hyb}'_\star$:

1. At the beginning of the protocol, for each $u \in [n]$, we compute $(D_u, \sigma_u) \leftarrow \mathsf{VRF.Eval}(\mathsf{crs}_{\mathrm{vrf}}, \mathsf{sk}_u, 1-b)$ and disclose to $\mathcal{A}$ the pair (D_u, σ_u) which is the evaluation outcome and proof for the bit $1-b$. Of course, upon the new corruption of some node v, we now need to explain to $\mathcal{A}$ the coins in the above evaluation for v too.
2. During the protocol, in each phase, we only run the $\mathsf{Distribute}^b$ protocol but not the $\mathsf{Distribute}^{1-b}$ protocol; similarly, we need not run the setup for $\mathsf{Distribute}^{1-b}$ either.
3. If some honest node u tried to call $\mathsf{Distribute}^b$ to send a valid $(r+1)$-batch of votes for b in the second round of phase r and u remains honest till the beginning of the third round of phase r, the experiment declares that $\mathcal{A}$ has failed to cause a Type-A bad event, and simply aborts outputting $\mathsf{adv\text{-}fail}$.
4. Immediately after the second round of phase R, for every honest node u who has already made a mining attempt for b, we disclose its VRF secret key sk_b to $\mathcal{A}$ even if u has not been corrupted by $\mathcal{A}$ (and note that these nodes do not count towards the corruption budget).
   Now, we allow $\mathcal{A}$ an extra **final guessing phase**, in which $\mathcal{A}$ can adaptively specify nodes to corrupt one by one; all nodes specified must not have made a mining attempt for b yet. Every time $\mathcal{A}$ specifies a new node u to corrupt, it learns its VRF secret key sk_u. The experiment stops when $\mathcal{A}$ has made f corruption queries in total. At this moment, if A has corrupted all members of the b-committee who have not made a mining attempt for b at the beginning of the final guessing phase, then declare that a Type-A bad event has happened.

Claim 6.1. *If for some non-uniform p.p.t. parallel machine $\mathcal{A}$, a Type-A bad event happens with probability μ in the real-world experiment* Real*, then there exists a non-uniform p.p.t. parallel machine $\mathcal{A}'$ such that a Type-A bad event happens with probability at least μ in* $\mathsf{Hyb}'_\star$.

Proof. We can add these modifications one by one and in each step argue that if there is a non-uniform *p.p.t.* parallel machine $\mathcal{A}$ in the previous experiment that can cause a Type-A bad event to occur with probability μ, then there is a non-uniform *p.p.t.* parallel machine $\mathcal{A}'$ in the modified experiment that can cause a Type-A bad event to occur with probability at least $\mu - \mathsf{negl}(\lambda)$.

First, we can add the modification 4. It is not hard to see that this phase only gives the adversary more opportunities in causing a Type-A bad event. In some sense, the final guessing phase is saying, at the end of the protocol, if not so-far honest nodes have made mining attempts for b (in this case a Type-A bad event cannot happen), we will pretend as if all of them made mining attempts for b and give $\mathcal{A}$ another opportunity to guess who the b-committee members are.

With modification 4, we can essentially imagine that all so-far honest nodes will have made mining attempts for b by the end of the protocol. Therefore, modification 3 is cosmetic, it basically checks for Type-A bad events constantly in the background and does not change the probability of Type-A bad event.

Next, we can add modifications 1 and 2. It is not hard to see that if there is a non-uniform $p.p.t.$ parallel machine $\mathcal{A}$ in the previous experiment that can cause a Type-A bad event to occur with probability μ, then we can construct a non-uniform $p.p.t.$ parallel machine $\mathcal{A}'$ in the modified experiment that can cause a Type-A bad event to occur with probability at least μ. Basically $\mathcal{A}'$ simply simulates the $\mathsf{Distribute}^{1-b}$ instances for $\mathcal{A}$ using its knowledge of all the VRF evaluations and proofs for $1-b$. Whenever $\mathcal{A}$ corrupts some v, $\mathcal{A}'$ learns the explanation of the coins that contributed towards v's VRF proofs for $1-b$, in this way, $\mathcal{A}'$ can provide the necessary explanation to $\mathcal{A}$ too. □

6.3.2 Experiment $\mathsf{Hyb}_\star$

Experiment $\mathsf{Hyb}_\star$ is almost the same as $\mathsf{Hyb}'_\star$, except that when the final guessing phase starts, every so-far honest node who has not made a mining attempt for b yet chooses a random D from an appropriate domain instead of using the honest VRF outcome to determine whether it is a member of the b-committee. Furthermore, during the final guessing phase, when $\mathcal{A}$ corrupts any node, we no longer disclose the node's VRF key to $\mathcal{A}$.

Lemma 6.2. *Assume that the VRF scheme satisfies pseudorandomness under selective opening (see Definition 3.3). Then, suppose that there is a non-uniform $p.p.t.$ parallel machine $\mathcal{A}$ that can cause a Type-A bad event to happen in $\mathsf{Hyb}'_\star$ with probability μ, then there is a non-uniform $p.p.t.$ parallel machine $\mathcal{A}'$ that can cause a Type-A bad event to happen in $\mathsf{Hyb}_\star$ with probability at least $\mu - \mathsf{negl}(\lambda)$ for some appropriate negligible function $\mathsf{negl}(\cdot)$.*

Proof. We prove this lemma through a sequence of intermediate hybrids described below.

Hybrid $\widetilde{\mathsf{H}}_\star$. The experiment $\widetilde{\mathsf{H}}_\star$ is defined in almost the same way as $\mathsf{Hyb}'_\star$ except with the following modifications:

- During the setup, we will replace the VRF's setup with the simulated setup algorithms $\mathcal{S}_0$ which generates the $(s_1, \ldots, s_n)$ part of the secret keys, and $\mathcal{S}_1$ which generates $(\mathsf{crs}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n, \tau)$. The adversary $\mathcal{A}$ is given the public components $\mathsf{crs}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n$.

- Whenever an honest node u needs to evaluate the VRF on input b', we compute the VRF outcome y honestly using sk_u, but call $\mathcal{S}_2(\tau, \mathsf{pk}_u, b', y)$ instead to compute a simulated proof using the trapdoor τ.
- Whenever an honest node u gets corrupted, we call the $\mathcal{S}_3$ algorithm which returns ρ_u and all the randomness u used in earlier VRF evaluations. Now return $\mathsf{sk}_u := (s_u, \rho_u)$ to $\mathcal{A}$. If this is before the final guessing round, also return the random coins output by $\mathcal{S}_3$ to $\mathcal{A}$, as well as randomness the newly corrupted node used in the $\mathsf{Distribute}$ protocol instances so far. Immediately after the second round of phase R, call $\mathcal{S}_3$ for every honest node v who has already made a mining attempt for b, and return s_u and the term ρ_v output by $\mathcal{S}_3$ to $\mathcal{A}$.

Claim 6.3. *Assume that the VRF scheme satisfies pseudorandomness under selective opening (see Definition 3.3). Then, $\mathcal{A}$'s views in $\widetilde{\mathsf{H}}_\star$ and $\mathsf{Hyb}'_\star$ are computationally indistinguishable.*

Proof. Follows directly from the second part of the definition of pseudorandomness under selective opening. □

Hybrid $\widetilde{\mathsf{H}}_f$. The experiment $\widetilde{\mathsf{H}}_f$ is almost the same as $\widetilde{\mathsf{H}}_\star$ except with the following modification: when the last node u becomes corrupt during the final guessing phase, for u and all remaining honest nodes who have not made a mining attempt for b, we choose a random number of appropriate length to determine whether the node is in the b-committee. Moreover, for the last corruption u in the final guessing stage, we do not disclose u's secret key or coins to $\mathcal{A}$.

Claim 6.4. *Assume that the VRF scheme satisfies pseudorandomness under selective opening. Then, if there exists a non-uniform p.p.t. parallel machine $\mathcal{A}$ that can cause a Type-A bad event to happen in $\widetilde{\mathsf{H}}_\star$ with probability μ, then there is a non-uniform p.p.t. parallel machine $\mathcal{A}'$ that can cause a Type-A bad event to happen in $\widetilde{\mathsf{H}}_f$ with probability at least $\mu - \mathsf{negl}(\lambda)$ for some negligible function $\mathsf{negl}(\cdot)$.*

Proof. Whenever $\mathcal{A}$ makes the last corruption query during the final guessing phase, whether a Type-A bad event has occurred is already determined no matter whether we disclose the secret key of the newly corrupt node to $\mathcal{A}$. Therefore, henceforth we simply assume that nothing is disclosed to $\mathcal{A}$ upon the last corruption during the final guessing phase.

Basically we define $\mathcal{A}'$ to be the same as $\mathcal{A}$ and it runs $\mathcal{A}$ till it makes the last corruption query during the final guessing phase. We show that if $\mathcal{A}$ can cause a Type-A bad event to happen in $\widetilde{\mathsf{H}}_\star$ with more than negligibly higher probability than $\mathcal{A}'$ in experiment $\widetilde{\mathsf{H}}_f$, we can construct a reduction $\mathcal{B}$ to break the first game in the definition of pseudorandomness under selective opening.

Essentially $\mathcal{B}$ interacts with its challenger $\mathcal{C}$ as defined in the first game in the definition of pseudorandomness under selective opening. Moreover, $\mathcal{B}$ simulates the experiment $\widetilde{\mathsf{H}}_\star$ to $\mathcal{A}$ right till the moment $\mathcal{A}$ makes the last corruption query in the final guessing phase.

- During setup, $\mathcal{B}$ asks its challenger $\mathcal{C}$ to run the setup who generates $(s_1, \ldots, s_n)$. $\mathcal{B}$ now runs $\mathcal{S}_1$ to generate $\mathsf{crs}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n, \tau$ and it gives the terms $\mathsf{crs}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n$ to $\mathcal{A}$.
- Whenever the experiment needs to evaluate the first term of the VRF outcome, $\mathcal{B}$ instead forwards the query to its challenger $\mathcal{C}$, and then it simulates the proof part by calling $\mathcal{S}_2$ just like in $\widetilde{\mathsf{H}}_\star$.
- Whenever $\mathcal{A}$ corrupts an honest node u (except for the last corruption query in the final guessing phase), $\mathcal{B}$ issues a corruption query to $\mathcal{C}$, learns s_u, and then calls the $\mathcal{S}_3$ algorithm which returns ρ_u and all the randomness u used in earlier VRF evaluations. Now return $\mathsf{sk}_u := (s_u, \rho_u)$ to $\mathcal{A}$, and if this is before the final guessing round, also return the random coins output by $\mathcal{S}_3$ to $\mathcal{A}$, as well as the coins u used in $\mathsf{Distribute}$ protocol instances so far.
- Immediately after the second round of phase R, for every honest node v who has already made a mining attempt for b, $\mathcal{B}$ issues a corruption query to $\mathcal{C}$ for v, learns the s_v, and then calls $\mathcal{S}_3$ to obtain ρ_v. It returns the pair (s_v, ρ_v) to $\mathcal{A}$.
- When $\mathcal{A}$ makes the last corruption query during the final guessing stage, $\mathcal{B}$ sends multiple challenge queries on the input b to $\mathcal{C}$ to obtain the evaluation outcomes for the last corrupted node, as well as all remaining honest nodes who have not made a mining attempt for b.

Besides the above, $\mathcal{B}$ simulates the rest of the $\widetilde{\mathsf{H}}_\star$ faithfully.

These evaluation outcomes are used to determine whether a Type-A event has happened at this moment. Note that if $\mathcal{C}$ returned random answers, the experiment is the same as $\mathcal{A}'$ interacting with $\widetilde{\mathsf{H}}_f$; otherwise it is the same as $\mathcal{A}$ interacting with $\widetilde{\mathsf{H}}_\star$.

We stress that in the above, we are using a multi-challenge version of the first-game of the pseudorandomness under selective opening notion, where in the challenge phase, the adversary can specify multiple challenge queries rather than a single one. As argued in Chan et al. [ACD+19], the multi-challenge version is equivalent to the single challenge version by a standard hybrid argument. □

Hybrid $\widetilde{\mathsf{H}}_{f-1}$. The experiment $\widetilde{\mathsf{H}}_{f-1}$ is almost the same as $\widetilde{\mathsf{H}}_f$ except with the following modification: when the second to last corruption query u is made in the final guessing phase (or if fewer than 2 corruption queries are made in the final guessing phase, then for all of them):

- we do not return anything to $\mathcal{A}$ upon the corruption query; and
- we use a random number for the node u as well as all remaining honest nodes who have not made a mining attempt for b, to determine if the corresponding node is a member of the b-committee.

Claim 6.5. *Assume that the VRF scheme satisfies pseudorandomness under selective opening. Then, if there exists a non-uniform p.p.t. parallel machine $\mathcal{A}$ that can cause a Type-A bad event to happen in $\widetilde{\mathsf{H}}_f$ with probability μ, then there is a non-uniform p.p.t. parallel machine $\mathcal{A}'$ that can cause a Type-A bad event to happen in $\widetilde{\mathsf{H}}_{f-1}$ with probability at least $\mu - \mathsf{negl}(\lambda)$ for some negligible function $\mathsf{negl}(\cdot)$.*

Proof. Basically, $\mathcal{A}'$ runs $\mathcal{A}$ till it makes the second to last corruption query u in the final guessing phase. $\mathcal{A}'$ also makes the same corruption query u as the second to last query, but for the last corruption query, it just chooses to corrupt an arbitrary honest node that has not made a mining attempt for b.

We can construct a reduction $\mathcal{B}$ in a similar way as in the proof of Claim 6.4, except that now $\mathcal{B}$ stops at the second to last corruption query (for the node u) in the final guessing stage, and then $\mathcal{B}$ asks $\mathcal{C}$ for the evaluation outcome on the input b for u as well as any remaining honest node who has not made a mining attempt for b. The returned evaluation outcomes are used to decide whether the corresponding node is a member of the b-committee. It is not hard to see that if $\mathcal{C}$ returns random answers, the experiment above would have the same probability of causing a Type-A bad event as in $\widetilde{\mathsf{H}}_{f-1}$; else it has the same probability of causing a Type-A bad event as in $\widetilde{\mathsf{H}}_f$. □

Hybrid $\widetilde{\mathsf{H}}_1$. We can define a sequence of hybrids $\widetilde{\mathsf{H}}_f$, $\widetilde{\mathsf{H}}_{f-1}$, …, $\widetilde{\mathsf{H}}_1$, until eventually we arrive at $\widetilde{\mathsf{H}}_1$, which is almost the same as $\mathsf{Hyb}_\star$ except that we are using the simulated setup and simulated VRF proofs in $\widetilde{\mathsf{H}}_1$. Specifically, in experiment $\widetilde{\mathsf{H}}_i$, when $\mathcal{A}$ is making the last but $(f-i+1)$-th corruption query in the final guessing phase, we switch to using random outcomes for all remaining honest nodes who have not made a mining attempt for b (including the one being corrupted right now), to determine if the corresponding node is in the b-th committee; and moreover at this moment we do not disclose anything more to $\mathcal{A}$.

Claim 6.6. *Assume that the VRF scheme satisfies pseudorandomness under selective opening. Then, if there exists a non-uniform p.p.t. parallel machine $\mathcal{A}$ that can cause a Type-A bad event to happen in $\widetilde{\mathsf{H}}_i$ with probability μ, then there is a non-uniform p.p.t. parallel machine $\mathcal{A}'$ that can cause a Type-A bad event to happen in $\widetilde{\mathsf{H}}_{i-1}$ with probability at least $\mu - \mathsf{negl}(\lambda)$ for some negligible function $\mathsf{negl}(\cdot)$.*

Proof. The proof is essentially identical to that of Claim 6.5. □

Claim 6.7. *Assume that the VRF scheme satisfies pseudorandomness under selective opening. Then, $\mathcal{A}$'s views in $\widetilde{\mathsf{H}}_1$ and $\mathsf{Hyb}_\star$ are computationally indistinguishable.*

Proof. Directly follows from the second part of the pseudorandomness under selective opening notion. □

With the above sequence of hybrid experiments, we have concluded the proof of Lemma 6.2. □

6.3.3 Experiments Hyb'_r and Hyb_r

We define a sequence of hybrids below $\{\mathsf{Hyb}'_r, \mathsf{Hyb}_r\}_{r\in[1,R]}$.

Experiment Hyb'_r. Experiment Hyb'_r is almost identical as $\mathsf{Hyb}_\star$ except the following modifications:

- In phase r of the protocol, we pretend instead that so-far honest nodes use the inputs $\mathbf{0}$ for the phase-r $\mathsf{Distribute}^b$ protocol, and compute their first-round messages (denoted M) of the $\mathsf{Distribute}^b$ protocol. We give M to $\mathcal{A}$, and let it run till the beginning of the third round of phase r (i.e., the second round of the phase-r $\mathsf{Distribute}$ protocol). $\mathcal{A}$ outputs, among other terms, a set of new nodes to corrupt by the beginning of the third round of phase R.
- At this moment, every remaining honest node who has not yet made a mining attempt for b will use a random string of appropriate length to determine if it is a member of the b-committee. We now let $\mathcal{A}$ engage in a final guessing phase as defined before.

The definition of a Type-A bad event in Hyb_r is the same as in $\mathsf{Hyb}_\star$.

Experiment Hyb_r. Experiment Hyb_r is almost the same as $\mathsf{Hyb}_\star$ except that at the beginning of the second round of phase r, the experiment discloses all honest nodes' secret keys for the $\mathsf{Distribute}^b$ instance to $\mathcal{A}$. $\mathcal{A}$ now enters the final guessing phase, in which all remaining honest nodes who have not yet made mining attempts for b switch to using random coins to determine if they are a member of the b-committee.

Claim 6.8. *Assume that the* $\mathsf{Distribute}$ *protocol satisfies momentary secrecy. Then, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds: if for some non-uniform p.p.t.* $\mathcal{A}$*, Type-A bad events happen with probability* μ *in* $\mathsf{Hyb}_\star$*, then there must exist a non-uniform p.p.t.* $\mathcal{A}'$ *such that Type-A bad events happen with probability at least* $\mu - \mathsf{negl}(\lambda)$ *in* Hyb'_R*.*

Proof. Consider the following experiment Expt^β in which a reduction $\mathcal{B}$ interacts with a challenger $\mathcal{C}$ as well as the adversary $\mathcal{A}$.

- For the $\mathsf{Distribute}^b$ instance, it will embed the public parameters passed to it by $\mathcal{C}$.
- Whenever $\mathcal{B}$ needs to play on behalf of honest nodes in $\mathsf{Distribute}^b$ protocols (not including in phase R), it forwards the query to $\mathcal{C}$ instead providing the inputs of the so-far honest nodes, and acts as a relay between $\mathcal{C}$ and $\mathcal{A}$ for messages of the $\mathsf{Distribute}^b$ protocol.
- Whenever some honest node is corrupted by $\mathcal{A}$, it forwards the corruption query to $\mathcal{C}$, and forwards the internal states returned by $\mathcal{C}$ to $\mathcal{A}$; besides this, $\mathcal{B}$ also gives $\mathcal{A}$ the secret keys and random coins pertaining to the VRF of the newly corrupt node.
- Finally, during phase R, $\mathcal{B}$ invokes a challenge session with $\mathcal{C}$. Depending on the bit β, $\mathcal{C}$ will either use the honest nodes' real inputs in the challenge $\mathsf{Distribute}^b$ protocol if $\beta = 0$; else if $\beta = 1$, it will use the vector $\mathbf{0}$ as honest inputs to the challenge $\mathsf{Distribute}^b$ protocol. Let M^β be the first-round messages of the challenge $\mathsf{Distribute}^b$ protocol computed by $\mathcal{C}$. Let view^β be the joint view of $\mathcal{A}$ and $\mathcal{B}$ at this point.
- ($\diamond$): Now, give M^β to $\mathcal{A}$, run it till the beginning of the next round, and $\mathcal{A}$ outputs, among other terms, a set K^β of nodes to corrupt.

- At this moment, any remaining honest node who has not made a mining attempt for b uses random coins to decide if it is a member of the b-committee, and we let $\mathcal{A}$ engage in the final guessing phase.

Besides the above, $\mathcal{B}$ simply runs the experiment $\mathsf{Hyb}_\star$ faithfully. Observe that if $\beta = 0$, the experiment is the same as $\mathsf{Hyb}_\star$ till the beginning of the third round of phase R (i.e., second round of the phase-R Distribute protocol); else if $\beta = 1$, the experiment is the same Hyb'_R till the beginning of the third round of phase R.

Let X denote the total number of honest nodes who have not made a mining attempt for b by the beginning of the third round of phase R, let Y denote the total number of nodes corrupted by the beginning of the third round of phase R, and let Z be a bit indicating whether at the beginning of the third round of phase R, adv-fail has occurred—recall that if the set K does not contain all honest b-committee members who made a mining attempt for b in phase R, then adv-fail would occur.

Recall that after the beginning of the third round of phase R, the experiment enters a final guessing phase in which all honest nodes who have not made a mining attempt for b yet uses random coins to decide if they are members of the b committee. To prove that the probability of Type-A bad events in Hyb'_R can must be at least as high as in $\mathsf{Hyb}_\star$ barring negligible differences, it suffices to show that for any $x \in [n]$, $0 \leq y \leq x$, and $z \in \{0, 1\}$,

$$\left| \Pr_{\mathsf{Expt}^0} [X = x, Y = y, Z = z] - \Pr_{\mathsf{Expt}^1} [X = x, Y = y, Z = z] \right| \leq \mathsf{negl}(\lambda)$$

Suppose not. Then, there must exist $x \in [n]$, $0 \leq y \leq x$, and $z \in \{0, 1\}$, such that $\Pr_{\mathsf{Expt}^0} [X = x, Y = y, Z = z]$ and $\Pr_{\mathsf{Expt}^1} [X = x, Y = y, Z = z]$ differ by a non-negligible amount. Now, we can construct a non-uniform $p.p.t.$ parallel $2\mathcal{T}_\emptyset$-bounded distinguisher $\mathcal{D}$ that can distinguish (M^0, view^0) and (M^1, view^1) with more than negligible probability. Basically $\mathcal{D}$ takes M^β and view^β, and runs whatever $\mathcal{A}$ runs in the $(\diamond)$ step for exactly one round which is $\mathcal{T}_\emptyset$ amount of time. At this moment, among $\mathcal{A}$'s output, there is an additional set K of nodes to corrupt. Finally, $\mathcal{D}$ tallies the counts X, Y, and the bit Z; here the tallying includes the set K. $\mathcal{D}$ outputs 1 if $(X, Y, Z) = (x, y, z)$; else it outputs 0. The tallying can be computed in logarithmic in n parallel time. Due to our assumption on the duration of an round, that is, an honest node must be able to process all n received messages within a round (see Sect. 3.1), the tallying can be computed in a single round, that is, at most $\mathcal{T}_\emptyset$ time. Therefore, $\mathcal{D}$ runs in at most $2\mathcal{T}_\emptyset$ time in total.

Remark 3. We stress that $\mathcal{A}$'s views are NOT computationally indistinguishable in the two hybrids since $\mathcal{A}$ can run in unbounded parallel time. We are merely arguing that the probability of Type-A bad events are not decreased by switching the phase-R Distribute messages. It is NOT true that $(M^0, \mathsf{view}^0, K^0)$ and $(M^1, \mathsf{view}^1, K^1)$ are computationally indistinguishable, because (M^0, K^0)

and (M^1, K^1) can potentially be distinguished by an adversary running in sufficiently long time. However, *any property on* $(M^0, \mathsf{view}^0, K^0)$ *or* $(M^1, \mathsf{view}^1, K^1)$ *that can be checked in small parallel runtime should happen with almost the same probability* regardless of the choice of β. □

Claim 6.9. *Assume that the VRF scheme satisfies pseudorandomness under selective opening. Then, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds: if for some non-uniform p.p.t.* $\mathcal{A}$*, Type-A bad events happen with probability* μ *in* Hyb'_R*, then there must exist a non-uniform p.p.t.* $\mathcal{A}'$ *such that Type-A bad events happen with probability at least* $\mu - \mathsf{negl}(\lambda)$ *in* Hyb_R*.*

Proof. First, disclosing all honest secret keys for the $\mathsf{Distribute}^b$ protocol at the beginning of the second round of phase R discloses strictly more information to $\mathcal{A}$ than in the earlier Hyb'_R. Next, we can repeat the same argument as in the proof of Lemma 6.2, that we can switch to using random coins to decide whether the following nodes are members of the b-committee: the nodes that remain honest at the beginning of the second round of phase R and have not yet made any mining attempts for b. □

Claim 6.10. *Assume that the* $\mathsf{Distribute}$ *protocol satisfies momentary secrecy, and that the VRF scheme satisfies pseudorandomness under selective opening. Then, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds: if for some non-uniform p.p.t.* $\mathcal{A}$*, Type-A bad events happen with probability* μ *in* $\mathsf{Hyb}_\star$*, then there must exist a non-uniform p.p.t.* $\mathcal{A}'$ *such that Type-A bad events happen with probability at least* $\mu - \mathsf{negl}(\lambda)$ *in* Hyb_1*.*

Proof. The proof works through a sequence of hybrids from Hyb_R to Hyb'_{R-1}, to Hyb_{R-1}, to Hyb'_{R-2} and so on, where to argue each adjacent pair of hybrids we use either the same proof as Claim 6.8 or the same proof as Claim 6.9. □

Lemma 6.11. *Let* b *be an arbitrary bit. For any non-uniform p.p.t. parallel machine* $\mathcal{A}$*, the probability of a Type-A bad event for* b *in* Real *is negligibly small.*

Proof. Notice that in Hyb_1, even for an unbounded adversary making $f = n - h$ corruptions, the expected number of forever-honest nodes that belong to the b-committee is $\Theta(\log^2 \lambda)$. By the Chernoff bound, the probability that $\mathcal{A}$ succeeds in guessing and corrupting all members of the b-committee is upper bounded by $\mathsf{negl}(\lambda)$.

Now, the earlier sequence of hybrids established that the probability of a Type-A bad event for b in Real must be upper bounded by the probability of a Type-A bad event in Hyb_1 against an unbounded adversary plus $\mathsf{negl}(\lambda)$. □

6.4 Consistency Proofs

The following lemma bounds the probability of a Type-B bad event for either $b = 0$ or $b = 1$.

Lemma 6.12. *Fix an arbitrary $b \in \{0,1\}$. Assume that the VRF scheme satisfies pseudorandomness under selective opening. Then, Type-B bad events do not happen except with negligible probability.*

Proof. If we used random coins to decide if each node is a member of the b-committee, then the probability that at least $R = 6\log^2 \lambda \cdot \frac{n}{h}$ nodes are members of the b-committee is upper bounded by a negligible function in λ by the Chernoff bound. Similarly, the probability that no node is a member of the b-committee is also negligibly small in λ. Now, rather than true randomness, we are using the VRF which gives pseudorandomness, and because the function that determines how many nodes are in the b-committee is a polynomial function on the outcomes of the VRF, it holds that the same holds when the true random coins are replaced with pseudorandom ones. □

So far, we have concluded that neither Type-A nor Type-B bad events happen except with negligible probability, for either bit $b \in \{0,1\}$. To prove consistency, it suffices to show that if neither types of bad events happen for both bits except with negligible probability, then consistency follows. This is stated and proven in the following theorem.

Theorem 6.13 (Consistency)**.** *Assume that $n \geq \log^2 \lambda$, that the VRF scheme satisfies pseudorandomness under selective opening as well as unforgeability, and that the* Distribute *protocol satisfies liveness and momentary secrecy. Then, the Byzantine Broadcast protocol defined earlier in this section satisfies consistency.*

Proof. Due to Lemmas 6.11 and 6.12, the liveness of the Distribute protocol, as well as the unforgeability of the VRF, it suffices to prove that the following hold in some execution, then the execution satisfies consistency.

1. for either $b = 0$ or $b = 1$, neither Type-A nor Type-B bad events happen for b;
2. the liveness property of Distribute is never broken;
3. for either $b = 0$ or $b = 1$, if any so-far honest node u has not made a mining attempt for b, then there is no valid vote from u for b in any honest node's view.

Observe that an inconsistency can only take place if some forever-honest node u includes a bit b in its $\mathsf{Extracted}_u$ set but some other honest node v does not include b in its $\mathsf{Extracted}_v$ set. We now consider the following cases:

- *Case 1*: u first added the bit b to its $\mathsf{Extracted}_u$ set in some phase r but *not* in the first round of phase r. According to our protocol, in phase r, u must have observed an r-batch of votes for b, made a successful mining attempt for the bit b, and moreover, it must have tried to distribute an $(r+1)$-batch of votes for b. Since Type-B bad events do not happen and votes are not forged, it must be that $r < R$.
  Since u is forever honest, by the liveness property of Distribute, it must be that by the end of the phase-r Distribute protocol, all forever-honest nodes will have observed the $(r+1)$-batch of votes from u. Therefore, every forever-honest node v will have added b to its $\mathsf{Extracted}_v$ set in the first round of phase $r + 1 \leq R$.

- *Case 2*: u first added the bit b to its $\mathsf{Extracted}_u$ set in some phase r but in the first round of phase r. This means that u has observed an r-batch of votes for b in the first round of phase r. Since Type-B bad events do not happen and votes are not forged, it must be that $r < R$.
  Because u is forever honest, it must be that at the beginning of the second round of phase r, all so-far nodes have observed the same r-batch of votes for b that u saw. Now, all so-far honest nodes will make a mining attempt for b if they have not done so already. Now, since Type-A and Type-B bad events do not happen, there must exist a so-far honest node v that made a successful mining attempt for the bit b in the second round of some phase $r' \leq r$, and moreover, the adversary did not yet corrupt v at the beginning of the third round of phase r'. Now, by liveness of the Distribute protocol, by the beginning of phase $r'+1 \leq R$, all forever-honest nodes will have observed the $(r'+1)$ batch of votes for b that v tried to distribute in phase r', and therefore, by the end of the first round of phase $r'+1$, every forever-honest node w will have added the bit b to its $\mathsf{Extracted}_w$ set. □

6.5 Validity Proofs

Theorem 6.14 (Validity)**.** *Suppose that $n \geq \log^2 \lambda$, that the VRF scheme satisfies pseudorandomness under selective opening as well as unforgeability, and that the* Distribute *protocol satisfies liveness and momentary secrecy. Suppose also that the designated sender is forever-honest and its input is $b' \in \{0,1\}$. Then, except with negligible probability, if any forever-honest node outputs b at the end of the protocol, it must be that $b = b'$.*

Proof. If the designated sender $u = 1$ is forever-honest, let b be its input bit, then node $u = 1$ must distribute a valid 1-batch of votes for b in the first round of the first phase. Thus, by the beginning of the second round of the first phase, all so-far honest nodes will have made a mining attempt for b. Because Type-A and Type-B bad events do not happen except with negligible probability, it must be that except with negligible probability, at least one node u successfully mines a vote for b in phase 1 and the node u remains honest till at least the beginning of the third round of the first phase. By the liveness property of Distribute, it must be that except with negligible probability, by the beginning of the second phase, every so-far honest node v will have observed a valid 2-batch of votes for b, and will have added the bit b to its $\mathsf{Extracted}_v$ set. Finally, validity follows by observing that due to the unforgeability of the VRF, no valid batch of votes for $1-b$ can appear in any honest node's view except with negligible probability. □

7 Conclusion and Open Questions

Our work is the first to show a sublinear-round Byzantine Broadcast protocol secure in the presence of *corrupt majority* and *strongly adaptive* corruptions.

Our work leaves open several exciting directions for future work:

- Recall that Garay et al. [GKKO07] show an $\Omega(\frac{n}{n-f})$ lower bound even for randomized protocols and static corruptions. Our round complexity is $\left(\frac{n}{n-f}\right)^2 \mathsf{poly}\log\lambda$ assuming that the puzzle's hardness parameter ξ is a constant. Although our round complexity is only polylogarithmic even with, say, 99% corruption, it is an intriguing question whether we can match the elegant lower bound of Garay et al. [GKKO07].
- Another interesting and natural direction is whether we can get rid of cryptographic assumptions such as time-lock puzzles.
- In the honest majority setting, it is long known how to construct expected constant-round protocols even for strongly adaptive adversaries [FM97, KK09, ADD+19]. Therefore, an interesting question is whether we can attain expected constant-round protocols in the corrupt majority setting under strongly adaptive corruptions.
- Finally, as mentioned earlier, it would be interesting to understand how general our current Distribute primitive is, and whether one can devise a general compiler that upgrades any weakly adaptive protocol to a strongly adaptive one while preserving its security.

Acknowledgment. We are extremely grateful to the TCC reviewers for their detailed and insightful comments and suggestions that helped greatly in improving the paper. We are especially grateful to our shepherd Ran Cohen who spent a significant amount of time to help us improve the paper. We acknowledge helpful technical discussions with Kai-Min Chung, Ilan Komargodski, Hoeteck Wee, and Rafael Pass about time-lock puzzles. This work is in part supported by an NSF grant under the award number CNS-1561209, and a Packard Fellowship.

A Additional Preliminaries

A.1 The Decisional Linear Assumption

Suppose that $\mathcal{G}(1^\lambda)$ is a group generator that samples a bilinear group $(\mathbb{G}, \mathbb{G}_T)$ of prime order p, along with a pairing operation $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and a random generator $g \in \mathbb{G}$. The decisional linear assumption posits that the following two probability ensembles are computationally indistinguishable:

1. Run $\mathcal{G}(1^\lambda)$ to generate a bilinear group $(\mathbb{G}, \mathbb{G}_T)$ of prime order p, along with a pairing operation $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and a random generator $g \in \mathbb{G}$. Sample random x, y, r, s at random from $\mathbb{Z}_p$. Output the tuple $(g, g^x, g^y, g^{xr}, g^{ys}, g^{r+s})$ as well as the group description.
2. Run $\mathcal{G}(1^\lambda)$ to generate a bilinear group $(\mathbb{G}, \mathbb{G}_T)$ of prime order p, along with a pairing operation $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and a random generator $g \in \mathbb{G}$. Sample random x, y, r, s, d at random from $\mathbb{Z}_p$. Output the tuple $(g, g^x, g^y, g^{xr}, g^{ys}, g^d)$ as well as the group description.

A.2 Adaptively Secure Non-interactive Zero-Knowledge Proofs

We use $f(\lambda) \approx g(\lambda)$ to mean that there exists a negligible function $\nu(\lambda)$ such that $|f(\lambda) - g(\lambda)| < \nu(\lambda)$.

A non-interactive proof system henceforth denoted NIZK for an NP language $\mathcal{L}$ consists of the following algorithms.

- $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{L})$: Takes in a security parameter λ, a description of the language $\mathcal{L}$, and generates a common reference string crs.
- $\pi \leftarrow \mathsf{P}(\mathsf{crs}, \mathsf{stmt}, w)$: Takes in crs, a statement stmt, a witness w such that $(\mathsf{stmt}, w) \in \mathcal{L}$, and produces a proof π.
- $b \leftarrow \mathsf{V}(\mathsf{crs}, \mathsf{stmt}, \pi)$: Takes in a crs, a statement stmt, and a proof π, and outputs 0 (reject) or 1 (accept).

Perfect Completeness. A non-interactive proof system is said to be perfectly complete, if an honest prover with a valid witness can always convince an honest verifier. More formally, for any $(\mathsf{stmt}, w) \in \mathcal{L}$, we have that

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{L}),\ \pi \leftarrow \mathsf{P}(\mathsf{crs}, \mathsf{stmt}, w) : \mathsf{V}(\mathsf{crs}, \mathsf{stmt}, \pi) = 1\right] = 1.$$

Non-erasure Computational Zero-Knowledge. Non-erasure zero-knowledge requires that under a simulated CRS, there is a simulated prover that can produce proofs without needing the witness. Further, upon obtaining a valid witness to a statement a-posteriori, the simulated prover can explain the simulated NIZK with the correct witness.

We say that a proof system $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ satisfies non-erasure computational zero-knowledge iff there exist a probabilistic polynomial-time algorithms $(\mathsf{Gen}_0, \mathsf{P}_0, \mathsf{Explain})$ such that

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda), \mathcal{A}^{\mathsf{Real}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1\right] \approx \Pr\left[(\mathsf{crs}_0, \tau_0) \leftarrow \mathsf{Gen}_0(1^\lambda), \mathcal{A}^{\mathsf{Ideal}(\mathsf{crs}_0,\tau_0,\cdot,\cdot)}(\mathsf{crs}_0) = 1\right],$$

where $\mathsf{Real}(\mathsf{crs}, \mathsf{stmt}, w)$ runs the honest prover $\mathsf{P}(\mathsf{crs}, \mathsf{stmt}, w)$ with randomness r and obtains the proof π, and then outputs (π, r); $\mathsf{Ideal}(\mathsf{crs}_0, \tau_0, \mathsf{stmt}, w)$ runs the simulated prover $\pi \leftarrow \mathsf{P}_0(\mathsf{crs}_0, \tau_0, \mathsf{stmt}, \rho)$ with randomness ρ and without a witness, and then runs $r \leftarrow \mathsf{Explain}(\mathsf{crs}_0, \tau_0, \mathsf{stmt}, w, \rho)$ and outputs (π, r).

Perfect Knowledge Extraction. We say that a proof system $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ satisfies perfect knowledge extraction, if there exists probabilistic polynomial-time algorithms $(\mathsf{Gen}_1, \mathsf{Extr})$, such that for all (even unbounded) adversary $\mathcal{A}$,

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda) : \mathcal{A}(\mathsf{crs}) = 1\right] = \Pr\left[(\mathsf{crs}_1, \tau_1) \leftarrow \mathsf{Gen}_1(1^\lambda) : \mathcal{A}(\mathsf{crs}_1) = 1\right],$$

and moreover,

$$\Pr\left[(\mathsf{crs}_1, \tau_1) \leftarrow \mathsf{Gen}_1(1^\lambda); (\mathsf{stmt}, \pi) \leftarrow \mathcal{A}(\mathsf{crs}_1); w \leftarrow \mathsf{Extr}(\mathsf{crs}_1, \tau_1, \mathsf{stmt}, \pi) : \begin{array}{l}\mathsf{V}(\mathsf{crs}_1, \mathsf{stmt}, \pi) = 1 \\ \text{but } (\mathsf{stmt}, w) \notin \mathcal{L}\end{array}\right] = 0.$$

Theorem A.1 (Instantiation of NIZK [GOS12])**.** *Assume that the decisional linear assumption holds in suitable bilinear groups. Then, there exists a proof system that satisfies perfect completeness, non-erasure computational zero-knowledge, and perfect knowledge extraction.*

References

[ACD+19] Abraham, I., et al.: Communication complexity of Byzantine agreement, revisited. In: PODC (2019)

[ADD+19] Abraham, I., Devadas, S., Dolev, D., Nayak, K., Ren, L.: Synchronous Byzantine agreement with optimal resilience, expected $o(n^2)$ communication, and expected o(1) rounds. In: Financial Cryptography and Data Security (FC) (2019)

[BBBF18] Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 757–788. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_25

[BGJ+16] Bitansky, N., Goldwasser, S., Jain, A., Paneth, O., Vaikuntanathan, V., Waters, B.: Time-lock puzzles from randomized encodings. In: Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science (ITCS), pp. 345–356 (2016)

[Can00] Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptol. **13**(1), 143–202 (2000)

[CCGZ16] Cohen, R., Coretti, S., Garay, J., Zikas, V.: Probabilistic termination and composability of cryptographic protocols. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 240–269. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_9

[CHM+19] Cohen, R., Haitner, I., Makriyannis, N., Orland, M., Samorodnitsky, A.: On the round complexity of randomized Byzantine agreement. In: 33rd International Symposium on Distributed Computing, DISC 2019, Budapest, Hungary, 14–18 October 2019, pp. 12:1–12:17 (2019)

[CL99] Castro, M., Liskov, B.: Practical Byzantine fault tolerance. In: OSDI (1999)

[CMS89] Chor, B., Merritt, M., Shmoys, D.B.: Simple constant-time consensus protocols in realistic failure models. J. ACM **36**(3), 591–614 (1989)

[CPS19a] Hubert Chan, T.-H., Pass, R., Shi, E.: Consensus through herding. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 720–749. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_24

[CPS19b] Hubert Chan, T.-H., Pass, R., Shi, E.: Sublinear-round byzantine agreement under corrupt majority (2019). Online full version of this paper. https://eprint.iacr.org/2019/886

[CPS20] Chan, T.-H.H., Pass, R., Shi, E.: Sublinear-round byzantine agreement under corrupt majority. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 246–265. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_9

[CS20] Chan, B.Y., Shi, E.: Streamlet: textbook streamlined blockchains. Cryptology ePrint Archive, Report 2020/088 (2020). https://eprint.iacr.org/2020/088

[DPS16] Daian, P., Pass, R., Shi, E.: Snow white: robustly reconfigurable consensus and applications to provably secure proofs of stake. Cryptology ePrint Archive, Report 2016/919 (2016)

[DS83] Dolev, D., Raymond Strong, H.: Authenticated algorithms for Byzantine agreement. SIAM J. Comput. - SIAMCOMP **12**(4), 656–666 (1983)

[EFL17] Eckey, L., Faust, S., Loss, J.: Efficient algorithms for broadcast and consensus based on proofs of work. Cryptology ePrint Archive, Report 2017/915 (2017). https://eprint.iacr.org/2017/915

[Fel88] Felman, P.N.: Optimal algorithms for Byzantine agreement. Ph.D. dissertation, MIT (1988)

[FM97] Feldman, P., Micali, S.: An optimal probabilistic protocol for synchronous Byzantine agreement. SIAM J. Comput. **26**, 873–933 (1997)

[FN09] Fitzi, M., Nielsen, J.B.: On the number of synchronous rounds sufficient for authenticated Byzantine agreement. In: Keidar, I. (ed.) DISC 2009. LNCS, vol. 5805, pp. 449–463. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04355-0_46

[GKKO07] Garay, J., Katz, J., Koo, C.-Y., Ostrovsky, R.: Round complexity of authenticated broadcast with a dishonest majority. In: 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS), November 2007

[GKKZ11] Garay, J.A., Katz, J., Kumaresan, R., Zhou, H.-S.: Adaptively secure broadcast, revisited. In: Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, PODC 2011, pp. 179–186. ACM, New York (2011)

[GKL15] Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_10

[GOS12] Groth, J., Ostrovsky, R., Sahai, A.: New techniques for noninteractive zero-knowledge. J. ACM **59**(3), 11:1–11:35 (2012)

[HZ10] Hirt, M., Zikas, V.: Adaptively secure broadcast. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 466–485. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_24

[KK09] Katz, J., Koo, C.-Y.: On expected constant-round protocols for Byzantine agreement. J. Comput. Syst. Sci. **75**(2), 91–112 (2009)

[KMS14] Katz, J., Miller, A., Shi, E.: Pseudonymous secure computation from time-lock puzzles. IACR Cryptology ePrint Archive 2014:857 (2014)

[KY] Karlin, A., Yao, A.C.-C.: Probabilistic lower bounds for byzantine agreement. Manuscript

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

[LPS17] Lin, H., Pass, R., Soni, P.: Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In: 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, 15–17 October 2017, pp. 576–587 (2017)

[LSP82] Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. ACM Trans. Program. Lang. Syst. **4**(3), 382–401 (1982)

[MVR99] Micali, S., Vadhan, S., Rabin, M.: Verifiable random functions. In: FOCS (1999)

[PS17a] Pass, R., Shi, E.: Hybrid consensus: efficient consensus in the permissionless model. In: DISC (2017)

[PS17b] Pass, R., Shi, E.: Rethinking large-scale consensus. In: 30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, 21–25 August 2017, pp. 115–129 (2017)

[PS17c] Pass, R., Shi, E.: Rethinking large-scale consensus (invited paper). In: CSF (2017)

[PS17d] Pass, R., Shi, E.: The sleepy model of consensus. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10625, pp. 380–409. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70697-9_14

[PSS17] Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 643–673. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_22

[RSW96] Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Technical report, USA (1996)

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

[WXSD20] Wan, J., Xiao, H., Shi, E., Devadas, S.: Expected constant round byzantine broadcast under dishonest majority. Cryptology ePrint Archive, Report 2020/590 (2020). https://eprint.iacr.org/2020/590

[YMR+19] Yin, M., Malkhi, D., Reiter, M.K., Gueta, G.G., Abraham, I.: HotStuff: BFT consensus with linearity and responsiveness. In: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC), pp. 347–356. Association for Computing Machinery, New York (2019)