Efficient Asynchronous Byzantine Agreement without Private Setups

Yingzi Gao†‡§, Yuan Lu*, Zhenliang Lu†, Qiang Tang†, Jing Xu* and Zhenfeng Zhang‡

*Institute of Software, Chinese Academy of Sciences, Beijing, China
†School of Computer Science, The University of Sydney, Sydney, Australia
‡State Key Laboratory of Cryptology, Beijing, China
§University of Chinese Academy of Sciences, Beijing, China

{yingzi2019,luyuan}@iscas.ac.cn, zhlu9620@uni.sydney.edu.au, qiang.tang@sydney.edu.au,{xujing,zhenfeng}@iscas.ac.cn

Abstract—Efficient asynchronous Byzantine agreement (BA) protocols were mostly studied with private setups, e.g., pre-setup threshold cryptosystem. Challenges remain to reduce the large communication in the absence of such setups. Recently, Abraham et al. (PODC’21) presented the first asynchronous validated BA (VBA) with expected \( O(n^3) \) messages and \( O(1) \) rounds, relying on only public key infrastructure (PKI) setup, but the design still costs \( O(\lambda n^3 \log n) \) bits. Here \( n \) is the number of parties, and \( \lambda \) is a cryptographic security parameter.

In this paper, we reduce the communication of private-setup free asynchronous BA to expected \( O(\lambda n^2) \) bits. At the core of our design, we give a systematic treatment of common randomness protocols in the asynchronous network, and proceed as:

- We give an efficient reasonably fair common coin protocol in the asynchronous setting with only PKI setup. It costs only \( O(\lambda n^2) \) bit and \( O(1) \) rounds, and ensures that with at least 1/3 probability, all honest parties can output a common bit that is as if randomly flipped. This directly renders more efficient private-setup free asynchronous binary agreement (ABA) with expected \( O(\lambda^2) \) bits and \( O(1) \) rounds.
- Then, we lift our common coin to attain perfect agreement by using a single ABA. This gives us a reasonably fair random leader election protocol with expected \( O(\lambda n^2) \) communication and expected constant rounds. It is pluggable in all existing VBA protocols (e.g., Cachin et al., CRYPTO’01; Abraham et al., PODC’19; Lu et al., PODC’20) to remove the needed private setup or distributed key generation (DKG). As such, the communication of private-setup free VBA is reduced to expected \( O(\lambda n^2) \) bits while preserving fast termination in expected \( O(1) \) rounds. Moreover, our result paves a generic path to private-setup free asynchronous BA protocols, as it is not restricted to merely improve Abraham et al.’s specific VBA protocol (PODC’21).

Our results and techniques could be found useful and interesting for a broad array of applications such as asynchronous DKG and DKG-free asynchronous random beacon that is friendly for dynamic participation and reconfiguration.

Index Terms—asynchronous protocols, Byzantine agreement, coin tossing, random leader election, common randomness

I. INTRODUCTION

Recently, following the unprecedented demand of deploying BFT protocols on the Internet for robust and highly available decentralized applications, renewed attentions are gathered to implement more efficient asynchronous Byzantine agreements [1–6]. Nevertheless, asynchronous protocols have to rely on randomized executions to circumvent the seminal FLP “impossibility” result [7]. In particular, to quickly decide the output in expected constant rounds of interactions, many asynchronous protocols [1–6, 8–16] essentially need common randomness, given which for “costless”, one can construct asynchronous Byzantine agreement (BA) protocols with expected \( O(n^2) \) messages, expected \( O(1) \) rounds and optimal resilience.

However, efficient ways to implement asynchronous common randomness in practice mostly rely on varieties of private setups. For example, initiated by M. Rabin [17], it assumes that a trusted dealer directly uses secret sharing to distribute a large number of random secrets among the participating parties before the protocol starts. Later, Cachin et al. [11] presented how to set up a non-interactive threshold pseudorandom function (tPRF) by assuming that a trusted dealer can faithfully share a short tPRF key, which now is widely used by most practical asynchronous BFT protocols including [1–6].

These private setups might cause unpleasant deployment hurdles, preventing asynchronous protocols from being widely used in broader scenarios. Hence, it becomes critical to reduce the setup assumptions for easier real-world deployment.

Existing efforts on reducing setups. There are a few known approaches to private-setup free asynchronous BA but most are costly or even prohibitively expensive.

Back to 1993, Canetti and Rabin [9] presented the first asynchronous common coin framework (CR93) centered around asynchronous verifiable secret sharing (AVSS), from which an expected constant-round asynchronous binary agreement (ABA) can be realized. Here, AVSS is a two-phase protocol: a dealer can distributively “commit” a confidential secret in a sharing phase, and then the secret can be collectively recovered in another reconstructing phase. The CR93 common coin protocol can return a reasonably fair 1-bit randomness, which means there exists a constant probability that the honest parties output the common 0/1, though they might output differently sometimes. But CR93 exchanges tremendous \( O(n^6) \) messages and \( O(\lambda n^8 \log n) \) bits, where \( \lambda \) is the cryptographic security parameter and \( n \) is the number of parties. The huge complexities of CR93 are dominated by its expensive AVSS. Since then, many private-setup free AVSS protocols [18, 22–24] were proposed and can directly improve it. For example, Cachin et
al. [18] gave an AVSS to share \( n \) secrets with only \( O(n^2) \) messages and \( O(\lambda n^3) \) bits, but the resulting common coin and ABA (CKLS02) still incur \( O(n^3) \) messages and \( O(\lambda n^4) \) bits, which remains expensive and exists an \( O(n) \) gap between the message and the communication complexities.

Recently, Kokoris-KOGias et al. [19] (KMS20) presented a new path to constructing common coin primitive from AVSS.\(^1\) It costs \( O(\lambda n^4) \) bits and \( O(n) \) asynchronous rounds to flip a single random coin, which is seemingly worse than CKLS02. Nonetheless, once being bootstrapped, it can continually generate coins at a lower cost of \( O(\lambda n^2) \) bits and \( O(1) \) rounds per coin, thus being cheaper in an amortized way. In a recent breakthrough, Abraham et al. [16] gave an asynchronous validated Byzantine agreement (VBA) protocol (AJM+21) without private setup. It costs expected \( O(n^3) \) messages, \( O(1) \) rounds, and \( O([m/n]^2 + \lambda n^3 \log n) \) bits for \( |m| \)-bit input,\(^2\) and only assumes the presence of a bulletin PKI that aids in the registration of public keys. At the core of AJM+21, it strengthens reasonably fair common coin to a new random proposal election primitive, such that with a constant probability, the honest parties can randomly decide a common value proposed by some non-corrupted party. As such, a certain VBA protocol called No-Waiting’ HotStuff was tailored to cater for this special random election primitive. However, this election notion is too specific to be used in other existing VBA constructions [1, 2, 12] due to the imperfect of necessary agreement. Still, AJM+21 VBA costs \( O(\lambda n^3 \log n) \) bits, and leaves room for further reducing the communication cost asymptotically by removing the \( \log n \) factor.

A recent study from Das et al. [20] (concurrent to our results) proposed a more efficient reliable broadcast (RBC) protocol for large input, and thus can reduce the communication cost of AJM+21 VBA to expected \( O(\lambda n^3) \) bits, but it is unclear how to apply Das et al.’s new RBC protocol in other existing VBA protocols (e.g., [1, 2, 12, 28, 29]) to remove the needed private setups.

Bearing the state-of-the-art, it is needed to systematically treat asynchronous Byzantine-fault tolerant (BFT) protocols for higher efficiency with fewer setup. In particular, can we design efficient asynchronous common randomness protocols with fewer setup assumptions, thus reducing the expected communication cost of asynchronous Byzantine agreements (e.g., ABA and VBA) to \( O(\lambda n^3) \) bits?

\(^{1}\)KMS20 requires AVSS with high-threshold secrecy: the adversary cannot learn the secret before \( n - 2f \) honest parties start to reconstruct (where \( f \) is the number of corrupted parties). In contrast, the classic AVSS notion [9] only preserves secrecy before the first honest party activates reconstruction.

\(^{2}\)Throughout the paper, we consider the input size \( |m| \) of VBA at most \( \lambda n \) bits, so the \( |m|^2 \)-term does not dominate and thus ignored. For larger input, it can be an orthogonal problem to push the \( |m|^2/n^2 \) term to \( |m|/n \), as discussed by many “extension” protocols [2, 25–27] for multi-valued BA.

\section{Our contribution}

We give an affirmative answer to the above question. At the core of our solution, we develop a set of new techniques to empower an efficient private-setup free construction for a reasonably fair common coin that is also pluggable in many existing ABA protocols [8–10]; more interestingly, we put forth to and construct an efficient (reasonably fair) leader election notion with perfect agreement by lifting our common coin protocol to be always agreed. This leader election primitive can be directly plugged in all existing VBA protocols [1, 2, 12, 28, 29] to remove their reliance on private setups.

In greater detail, our technical contribution is three-fold:

- At the core of our solution, we design an efficient private-setup free common coin with \( O(\lambda n^3) \)-bit communication in the PKI setting. It is reasonably fair, and can be pluggable in many existing ABA protocols [8–10]. The resulting private-setup free ABA reduces the communication complexity of CKLS02 [18] by an \( O(n) \) factor, and preserve other benefits such as the maximal \( n/3 \) resilience and fast termination in expected constant rounds. Even comparing with AJM+21 [16] whose VBA construction

\section{Comparison of private-setup free asynchronous BA protocols}

|                  | ABA/Coin | VBA/Election |
|------------------|----------|--------------|
|                  | Comm. Round | Comm. Round | Adaptive Security? | Cryptographic Assumption | Setup Assumption |
| CKLS02 [18] \(^\dagger\) | \( O(\lambda n^3) \) | \( O(1) \) | Yes | Diag-hash | global param \(^*\) |
| KMS20 [19] \(^\dagger\) | \( O(\lambda n^4) \) | \( O(1) \) | No \(^*\) | RO+DHH \(^\dagger\) | PKI \(^\dagger\) |
| AJM+21 [16] \(^\dagger\) | \( O(\lambda n^3 \log n) \) | \( O(n) \) | No | RO+SXDHD | PKI |
| AJM+21 [16] + RBC in [20] | \( O(\lambda n^3) \) | \( O(1) \) | No | RO+SXDHD | PKI |
| This paper | \( O(\lambda n^3) \) | \( O(1) \) | Yes | RO+DHH | PKI, 1-time rnd \(^\dagger\) |

\(^\dagger\)Global parameters capture some minimal setups such as an agreed group description and group generators. For some schemes relying on collision-resistant hash [18, 19], a one-time common random string is also needed to key the hash functions.

\(^\dagger\) CKLS02 [18] did not construct VBA or leader election. We also do not realize any complexity-preserving reductions to it.

\(^\dagger\)KMS20 states that it might be adaptively secure by using the pairing-based adaptively secure threshold signature [21], and this might cause it rely on SXDH assumption instead of only DDH assumption.

\(^\dagger\)The PKI setup in KMS20 can be removed by recent high-threshold AVSS presented in [22].

\(^\dagger\)Note that KMS20 and AJM+21 did not present an explicit construction for random leader election (Election). Nevertheless, they gave asynchronous distributed key generation protocols (ADKG) that can bootstrap threshold verifiable random function and thus can set up Election (and also common coin) schemes via ADKG.

\(^\dagger\)AJM+21 only presents an explicit VBA construction but does not construct ABA. However, VBA implies ABA with some complexities, because there is a simple complexity-preserving reduction from ABA to VBA in the PKI setting, cf. [12].

\(^\dagger\)The communication of AJM+21 can be reduced to \( O(\lambda n^3) \) by a recent reliable broadcast protocol [20], but this only applies to the specific AJM+21 VBA construction, while our result is generic and can be adapted to all existing VBA protocols.

\(^\dagger\)1-time rnd means a one-time common random string can be published after PKI registration but before protocol execution.
improves ABA as a by-product, our approach still saves a log \( n \) factor in communication.

- We further present an efficient random leader election with perfect agreement lifted from our common coin by adding an ABA to clean possible disagreement. It replaces the counterpart relying on private setups in existing VBA protocols \([1, 2, 12]\). The resulting VBAs would terminate in expected constant rounds and cost expected \( O(\lambda n^3) \) bits when the input size \( |n| \leq \lambda n \). As an immediate application, it improves AJM+21 asynchronous distribution key generation (ADKG) to attain cubic communication, preserving expected constant-round termination.

- Along the way, we develop a set of crucial pertinent techniques that could be of independent interests. For example, we give an efficient AVSS construction satisfying the classic CR93 notion \([9]\) in the PKI setting under the discrete logarithm assumption, and it costs only \( O(\lambda n^2) \) bits when sharing a \( \lambda \)-bit secret. Prior art with the same communication complexity either relies on private setup \([30–32]\) or incurs at least \( O(\lambda n \log n) \) bits \([18, 22, 23, 33]\).

**B. Challenge and our techniques**

**Remaining efficiency hurdles.** To flip coins, both CR93 and AJM+21 let each party commit an unbiased secret gathered from sufficient parties. In CR93, everyone plays a role of “delegate” for each party to share a random secret through AVSS, then everyone picks and commits \( n - f \) secrets from distinct “delegates”, and thus the aggregation of these selected secrets is uniformly distributed. In AJM+21, every party combines \( n - f \) aggregatable public verifiable secret sharing (PVSS) from distinct parties, and then reliably broadcasts the \( O(\lambda n) \)-bit aggregated PVSS script (which can be analogous to AVSS, though no explicit AVSS invocation). After enough parties commit the gathered secrets, it selects a core-set of these unknown secrets. That means all parties decide a set of indices corresponding to some indeed committed secrets, and more importantly, the honest parties’ decisions have a large enough intersection (called core-set). Hence, a simple trick to toss the coin can be imagined: everyone reconstructs the committed secrets, then the lowest bits of the largest secret become the coin. This works because the largest secret has a constant probability to appear in the core-set. Usually, such a core-set can be made via several reliable broadcasts \([9, 15, 34]\), and these reliable broadcasts’ input is a set of \( O(n) \) indices.

As such, further reducing the communication of the CR93 and AJM+21 frameworks seems challenging because every party reliably broadcasts and/or verifiably shares at least \( O(\lambda n) \) bits. While, on the other side, although KMS20 gives a workaround to the above steps and reduces the number of secrets to share by an \( O(n) \) order, it on the contrary causes slow termination of \( O(n) \) rounds.

**Techniques to efficient common coin from PKI.** We present a collection of new techniques to circumvent the efficiency hurdles lying in the phases of committing secrets and selecting core-set as shown in Fig. 1. In greater detail, we proceed as: Constructing efficient AVSS from PKI. If we aim at common coin with cubic communication, the underlying AVSS must cost at most quadratic bits. We realize such AVSS by lifting Pedersen’s verifiable secret sharing \([35]\) to asynchronous network, through exploiting the wisdom of hybrid secret sharing \([36]\). First, the dealer just collects \( n - f \) signatures from distinct parties for the same Pedersen’s polynomial commitment \([35]\). These signatures ensure that at least \( f + 1 \) honest parties receive the same commitment binding a unique encryption key. Then, the dealer encrypts its actual secret, and leverages the \( n - f \) signatures to convince the honest parties to participate into a reliable broadcast for the ciphertext.

**Weakening Core-Set selection for more efficient construction.** We further tackle the problem of getting core-set efficiently. Our main observation is that if the confidentially shared secrets come with verifiability, e.g., VRFs, we might slightly weaken the primitive. I.e., only \( f + 1 \) honest parties obtain the core-set instead of all honest parties. This works because if the largest VRF appears in this weaker core-set, \( f + 1 \) parties can reconstruct this VRF, and then multicast it to let all parties accept it as the largest. The weaker notion can be easily designed in the PKI setting: each party multicasts its input set, and then waits for \( n - f \) signatures returned by distinct parties indicating that at least \( f + 1 \) honest parties have a superset of a common core. This becomes an efficient workaround to avoid \( O(n) \) reliable broadcasts in the conventional core-set selection \([9, 15, 16, 34]\).

A broadcast version “coin” to patch VRF. With more efficient AVSS and core-set selection at hand, it is enticing to let every party confidentially share its own VRF evaluation with
proof via AVSS (instead of gathering an unbiased secret from many parties). Then, the weak core-set selection can ensure that at least \( f + 1 \) honest parties get the core of \( n - f \) shared VRFs. As such, the largest VRF seemingly can appear in the core-set with a constant probability, and its least significant bits would naturally become the common coin.

However, the above seemingly appealing idea does not work in the bulletin PKI setting because if VRFs are evaluated on deterministic seeds (e.g., one-time common random string), corrupted parties can generate VRF keys maliciously to bias their own VRFs, for example: just run key generation for polynomial times, and register at PKI with the verification key of the most favorable key pair. So the VRFs evaluated by corrupted parties gain a great advantage to be the largest.

We therefore set forth a notion called reliable broadcasted seeding (Seeding) to provide VRF an unpredictable seed and construct it from aggregatable PVSS [37]. To some extent, Seeding can be viewed as a broadcast version common coin, which reliably “broadcasts” an unpredictable VRF seed. It is led by the party who evaluates VRF. So if a corrupted Seeding leader blocks or slows down the protocol, no honest party can verify its VRF, which actually “harms” the corrupted leader itself, because no honest party will solicit it into the core-set.

Putting everything together, we give a novel Coin framework illustrated in Fig. 2. A weak core-set can gather \( n - f \) AVSSes that share VRFs (patched by Seeding). With 1/3 probability, the largest VRF appears in the core and is also evaluated by some honest party, and its lowest bits become the flipped coin.

Lifting agreement for Leader Election. Although common coin is a powerful tool to enable asynchronous Binary agreement, it faces a few barriers for implementing the interesting class of VBA. The main issue stems from the fact that most existing VBAs [1, 2, 12] require a leader election with at least one significant strengthening relative to common coin, i.e., ensuring agreement all the time.

We use a single ABA along with a set of voting rules to “detect” the possible disagreement when the largest VRF does not appear in the core-set. In particular, everyone reliably broadcasts the speculative largest VRF obtained at the end of Coin execution, and then waits for \( n - f \) such broadcasts. If there exists a majority VRF that is also the largest in \( n - f \) received VRFs, they vote 1 to ABA; otherwise, they vote 0. When ABA returns 0, a default leader is elected. In case ABA outputs 1, at least an honest party receives a maximal VRF from \( f + 1 \) parties, and this VRF must be seen by any party that does cross-check \( n - f \) VRFs, so the VRF is unique and everyone can decide the same leader.

C. Other Related Work

Note on a concurrent work. A concurrent work from Das et al. [20] presented the technique of asynchronous data dissemination (ADD) to improve the efficiency of reliable broadcast and relevant protocols such as AVSS. It can reduce the communication of the specific AJM+21 VBA protocol to \( O(n^3) \) by replacing the reliable broadcast building block. It also has applications to reduce the communication of AJM+21 ADKG to cubic. Though Das et al.’s reliable broadcast and some of their proposed AVSS protocols can be adaptively secure, their applications to VBA and ADKG are also in the random oracle model against only static corruptions.

Different from Das et al. that focused on improving the broadcast components, we present a set of different techniques to simplify the protocol structure, which are the basis protocols of fast-terminating Byzantine agreement. In particular, our common coin and leader election can be directly plugged into any BA protocols that requires such a building block to improve the efficiency of their private-setup free variants, while ADD only explicitly helps the specific AJM+21 VBA (e.g., if without our results or future studies on asynchronous common randomness). In addition, our leader election protocol can be adapted into a reconfiguration-friendly random beacon protocol without DKG, while Das et al.’s results can only bootstrap random beacon protocol through DKG. Moreover, it might be interesting to explore the new design space provided by the combination of ADD and our techniques towards practical private-setup free asynchronous protocols.

Other relevant studies. Earlier content has carefully investigated the major efforts in the line of research for private-setup free asynchronous BA protocols. For space limit, we defer else relevant work to our online full version [38].

II. SYSTEM/THREAT MODELS AND PRELIMINARIES

Asynchronous system without private setup. There are \( n \) designated parties, each of which has a unique identity (i.e., \( P_1 \) through \( P_n \)) known by everyone. Moreover, we consider the following asynchronous message-passing model with Byzantine corruptions and bulletin public key infrastructure (PKI):

- **Bulletin PKI.** There exists a PKI functionality that can be viewed as a bulletin board, through which each party can register some public keys (e.g., the public key of digital signature) bounded to its identity. Once public keys are registered, all parties can receive them from the PKI.
- **Computing model.** Following modern cryptographic practices, we let the \( n \) parties and the adversary \( A \) be probabilistic polynomial-time interactive Turing machines.
- **Byzantine corruptions.** The adversary can choose up to \( f \) parties to fully control before the protocol execution. Through the paper, we stick with the optimal resilience \( f = \lfloor (n-1)/3 \rfloor \). The adversary can also control corrupted parties to register maliciously generated public keys at PKI. The adversary can also control the corrupted parties to exploit advantages while registering public keys at PKI.
- **Fully asynchronous network.** There is an asynchronous secure (private) channel between any two parties. The adversary can arbitrarily delay and reorder messages, but for messages sent between honest parties, it cannot modify/drop them or learn any information except their lengths. Such secure asynchronous channels can be obtained by running TLS over TCP connections in practice.

**Performance metrics.** We consider the standard quantitative metrics to characterize “efficiency”, including communication
complexity (the number of exchanged bits) and round complexity (the number of asynchronous rounds [9]).

Note on “private-setup free”. Our private-setup free model admits a bulletin PKI with some parameter systems that are just group descriptions and random group generators. The parameters are indeed one-time setup (of a global system instead of just for ours), and are usually ignored in the literature [16, 20]. Except that, we consider other structured common reference strings such that of KZG polynomial commitment [32] fall into the category of private setups.

Note on adaptive/static adversary. Some of our results (e.g., our AVSS protocol) can be secure against an adaptive adversary that can corrupt up to \( f \) parties while the protocol is running. While our common coin and random leader election protocols are secure in the static model, in which the adversary is restricted to corrupt parties before the protocol starts. However, this is only because the existing aggregatable PVSS is not proven to be adaptively secure (which actually is the same reason why AJM+21 [16] is statically secure). To demonstrate that, we can introduce a one-time online common random string assumption, thus avoiding the broadcasted seeding protocol that relies on PVSS, and then show that our common coin (as well as random leader election) become adaptively secure. Namely, we can assume a trusted one-time randomness that is announced after PKI registration but before protocol execution and adaptive security can be realized by our protocols in the setting, as PVSS is no longer needed, cf. more detailed discussions in our online version [38].

Moreover, the assumption of adaptively secure private channels can be easily realized by existing techniques (e.g., in the erasure model [39]).

**Reliable broadcast (RBC)** [40] allows a party called sender to send a value to all parties. It satisfies: (i) Agreement: the honest parties output the same value. (ii) Totality: if an honest party outputs, then all honest parties would output. (iii) Validity: if an honest sender inputs \( v \), then all honest parties output \( v \).

**Digital signature** consists of a tuple of algorithms (KenGen, Sign, SigVerify). We require it to be existentially unforgeable under adaptive chosen-message attack. Due to PKI, each party is bounded to a unique public key for signature verification.

**Verifiable random function** (VRF) [41] is a pseudorandom function with public verifiability. It consists three algorithms (VRF.Gen, VRF.Eval, VRF.Verify), and satisfies unpredictability, verifiability and uniqueness [41]. Everyone can register a unique VRF verification key via PKI. However, traditional VRF notion [41] does not capture malicious key generation. To consider such threat, we require a stronger unpredictability under malicious key generation due to David et al. [42], so VRF can perform like a random oracle, even if the key generation is malicious. Such VRF can be achieved in the random oracle model under CDH assumption [42].

### III. WARM-UP: AVSS AND WEAKER CORE SET FROM PKI

This Section focuses on the preparing building blocks—AVSS and core-set selection. Our AVSS protocol attains \( O(\lambda n^2) \) bits in the PKI setting.\(^3\) Then, we put forth to and construct a new weak core-set selection primitive, which can be used to ensure \( f + 1 \) honest parties (instead of all) to get some superset of a \( (n - f) \)-sized core-set.

#### A. Efficient Private-Setup Free AVSS

Though there exist varieties of strengthened AVSS notions, we focus on the hereinbelow classic AVSS notion defined by Canetti and Rabin in their seminal work [9]. Later, we will see this is still a powerful building block to empower our efficient common coin and leader election protocols.

**Definition 1 (Asynchronous Verifiable Secret Sharing).** AVSS consists of a tuple of protocols (AVSS-Sh, AVSS-Rec) and its syntax and properties can be defined as follows.

**Syntax.** In each AVSS-Sh instance with a session identifier ID, a designated dealer \( \mathcal{D} \) inputs a secret and each party outputs a string (e.g., a share of input secret). In the corresponding AVSS-Rec instance, the parties input their outputs of AVSS-Sh to collectively reconstruct the shared secret.

**Properties.** AVSS shall satisfy the following security properties except with negligible probability:

- **Totality.** If an honest party outputs in the AVSS-Sh instance associated to ID, then all honest parties activated to run the AVSS-Sh instance will output.

- **Commitment.** When an honest party outputs in the AVSS-Sh instance for ID, there exists a fixed value \( m^* \), such that when all honest parties are activated to run the corresponding AVSS-Rec instance, all of them can reconstruct the same value \( m^* \).

- **Correctness.** If the dealer is honest and inputs secret \( m \) in AVSS-Sh, then:
  - If all honest parties start to run AVSS-Sh on ID, all honest parties would output in the AVSS-Sh instance;
  - If any honest party reconstructs some value \( m^* \) in the corresponding AVSS-Rec instance, \( m^* = m \).

- **Secrecy.** While running an AVSS-Sh instance, if the dealer is honest, the adversary shall not learn any information about the input secret before the first honest party starts to run the corresponding AVSS-Sh instance.

**Constructing AVSS without private setups.** The rationale behind our AVSS construction is straightforward. The intuitions behind our AVSS construction are simple. Inspired by the hybrid approach of secret sharing [36], our sharing sub-protocol is split into two steps: it first confidentially commits to the polynomial \( A(x) \) with using \( B(x) \) for hiding. Here \( c_j = g_1^{a_j} g_2^{b_j} \), and \( a_j \) and \( b_j \) are the \( j \)-th coefficients of

\(^3\)Through the paper, the input secret to AVSS is assumed as \( O(\lambda) \) bits.
\(A(x)\) and \(B(x)\), respectively. Then, the dealer sends a KeyShare message to each party \(P_j\) containing \(C_i\), \(A(j)\) and \(B(j)\). If \(P_j\) receives KeyShare from the dealer, it checks \(C\) commits \(A(j)\), and returns a signature for \(C\) to dealer.

2) Cipher broadcast. After receiving \(n - f\) valid signatures for \(C\) from distinct parties, the dealer multicasts a Cipher message containing the commitment \(C\), a quorum proof \(\Pi\) made of \(n - f\) signatures for \(C\), and cipher = key \(\oplus m\). Here \(m\) is the input secret. The remaining process is similar to a Bracha’s reliable broadcast [40] for cipher, except that a party would not “echo” cipher if not yet receiving valid \(\Pi\) for \(C\). At the end of the phase, each party can output (cipher, \(A(i), B(i), C\)), where \(A(i), B(i)\) and \(C\) can be \(\perp\).

Then, the AVSS-Rec protocol can be constructed as follows:

1) Key recovery. \(P_i\) multicasts a KeyRec message containing \(A(i)\) and \(B(i)\), in case these variables are not \(\perp\). Then, it can eventually solicit \(f + 1\) valid shares of the polynomial committed to \(C\) through KeyRec messages, and then interpolate the shares to reconstruct \(A(x)\) and compute key = \(A(0)\).

2) Key amplification. After a party obtains decryption key, it multicasts key via a Key message. All honest parties can receive at least \(f + 1\) Key messages containing the same key, and then compute and output \(m = key \oplus cipher\).

Analyzing AVSS. The securities of the above AVSS construction can be asserted by the following theorem:

**Theorem 1.** The above AVSS-Sh and AVSS-Rec protocols realize AVSS as defined in Definition 1, in the asynchronous message-passing model with \(n/3\) adaptive Byzantine corruption and bulletin PKI assumption without private setups, conditioned on the hardness of Discrete Log problem and EUF-CMA security of digital signature.

**Proof Sketch.** The security intuition is straightforward: the totality is because of using Bracha broadcast [40] to distribute the ciphertext encrypting secret; the secrecy follows the information theoretic argument of Pedersen [35] about his verifiable secret sharing; the commitment is ensured by the binding of Pedersen commitment and unforgeability of signatures. Regarding complexities, both AVSS-Sh and AVSS-Rec terminate in constant asynchronous rounds, and they exchange \(O(n)\) messages with \(O(\lambda n)\) bits and \(O(n^2)\) messages with \(O(\lambda)\) bits, so the overall communication is \(O(\lambda n^2)\). For the sake of completeness, we describe the detailed security proof (along with formal protocol description) in our full version [38] due to space limit.

**Remark on Adaptive Security.** Our AVSS protocol is adaptively secure. Recall that our AVSS-Sh sub-protocol is similar to that of [18], which use Pedersen’s polynomial commitment in combination of Shamir’s secret sharing to consistently distribute the secret shares (of a symmetric encryption key) to the participating parties, which avoids the shortage of using static cryptographic primitives such as non-interactive PVSS [20]. While the major difference between [18] and our AVSS-Sh is that we concatenate \(n - f\) EUF-CMA secure digital signatures to form a quorum proof attesting that enough honest parties have received the consistent secret shares. Nevertheless, this doesn’t sacrifice adaptive security, because the quorum proof ensures that there must exist \(f + 1\) honest parties that can never be corrupted and also received the consistent secret shares, and hence these \(f + 1\) forever honest parties can help all other honest parties to recover the same encryption key during the AVSS-Rec protocol.

**B. Weak Core Set Selection**

Core-set selection is another critical component while flipping a coin in the asynchronous network [9]. It allows each party to output a set of indices representing some completed RBCs or AVSSes, and it ensures the intersection of all honest parties’ outputs corresponds to a large enough core-set.

Instead of this widely known approach, we introduce a weakened core-set selection, in which probably only \(f + 1\) honest parties can receive the core-set. It can be constructed very efficiently in the PKI setting, and remains to be an expressive notion for coin flipping. The idea is to use it to select a core-set of AVSSes that hides some VRVs. With a constant probability, the largest VRF can luckily appear in the core-set, so at least \(f + 1\) honest parties can reconstruct this largest VRF and multicast it to the whole network, thus still ensuring all honest parties get the same largest VRF.

Let us focus on the cautiously weakened notion and its efficient construction. More formally, the weak core-set selection can be defined as follows.

**Definition 2 (Weak Core-Set Selection).** A weak core-set selection protocol among \(n\) parties with up to \(f\) Byzantine corruptions can be defined as follows.

**Syntax.** For each protocol instance with session identifier ID, every party \(P_i\) inputs a set of indices \(S_i\) s.t. \(|S_i| \geq n - f\) and each index in \(S_i\) belongs to \([n]\). Note that each honest party’s input set \(S_i\) can be found in some honest party’s input set.

**Properties.** It satisfies the next properties except with negligible probability:

- **Termination.** If each index in any honest party’s input set can eventually appear in all honest parties’ input sets, every honest party would output.
- **(f + 1)-Supporting Core-Set.** Once the first honest party outputs, there exists a core-set \(S^*\) consisting of at least \(n - f\) distinct indices, and \(S^*\) must be the intersection of at least \(f + 1\) honest parties’ output sets.
- **Validity.** Every index in any honest party’s output set can be found in some honest party’s input set.

Intuitively, the above definition captures our purpose that after each party conducts reliable broadcast or verifiable secret sharing, each party can invoke the primitive to output a set of indices representing which reliable broadcasts or AVSSes are indeed completed. More importantly, the output sets of at
least \( f + 1 \) honest parties share a \((n - f)\)-sized intersection, representing that all these honest parties have output in these reliable broadcasts or AVSSes.

**Constructing WCS without private setup.** The WCS protocol can be simply constructed as follows:

1) Once an honest party \( P_i \) receives an input local set \( S \) which contains \( n - f \) values, it takes a “snapshot” \( \tilde{S} \) of \( S \) and multicasts \( \tilde{S} \) to all parties. Note that \( P_i \)'s local \( S \) can increase monotonically after the multicast, as new indices might be added to \( S \). Then, if receiving some \( \tilde{S}_j \) sent from some party \( P_j \), the party \( P_i \) checks \( |\tilde{S}_j| = n - f \), and waits for that its local \( \tilde{S} \) eventually becomes a superset of \( \tilde{S}_j \), after which, it returns a signature for \( \tilde{S}_j \) to \( P_j \).

2) Eventually, \( P_i \) might collect \( n - f \) distinct signatures for its multicasted “snapshot” \( \tilde{S} \), which corresponds to a quorum proof \( \Sigma \) for \( \tilde{S} \). Finally, \( P_i \) multicasts \( \Sigma \) and \( \tilde{S} \) to all parties. After receiving a valid quorum proof \( \Sigma_j \) for \( \tilde{S}_j \) from some party \( P_j \), the party \( P_i \) can immediately output its current local set \( S \) (without halt).

**Analyzing WCS.** The security guarantees of WCS can be formalized as the following theorem:

**Theorem 2.** The above construction realizes WCS against \( n/3 \) adaptive Byzantine corruptions in the asynchronous message-passing model, conditioned on that the underlying digital signature scheme is EUF-CMA secure.

**Proof Sketch.** The proof is trivial from the security of digital signature, because if a valid quorum proof is formed, there must be at least \( f + 1 \) honest parties sign the same core-set. For complexities, all parties can terminate after three rounds, and each honest party would send \( 3n \) messages with \( O(\lambda n) \) bits. So the overall communication protocol is provided in our full version [38].

**Remark on adaptive security.** We concatenate \( n - f \) digital signatures from distinct parties to attest the existence of a core set, and therefore, whenever any honest party outputs, there must exist \( f + 1 \) honest parties that have signed the core set and can never be corrupted by the adaptive adversary, and at least these \( f + 1 \) forever honest parties would share a \((n - f)\)-sized intersection in their output sets. These facts prevent the adaptive adversary from corrupting parties posteriorly to mimic the core-set which is received by less than \( f + 1 \) honest parties.

**IV. Backbone: Common Coin in the PKI Setting**

The backbone of our results is an efficient private-setup free common coin (Coin) protocol that costs only \( O(\lambda n^3) \) communicated bits and terminates in constant rounds. Formally, we consider the common coin notion defined as follows:

**Definition 3** \((n, f, f+k, \alpha)\)-Common Coin. A Coin protocol among \( n \) parties with up to \( f \) Byzantine corruptions shall have the next syntax and properties.

**Syntax.** For all executions of each protocol instance with session identifier \( ID \), every party takes the system’s public knowledge (i.e., \( \lambda \) and all public keys) and its own private keys as input, and outputs a single bit.

**Properties.** It satisfies the next properties except with negligible probability:

- **Termination.** If all honest parties are activated on \( ID \), every honest party will output a bit for \( ID \).

- **Reasonably fair bit-tossing.** Prior to that \( k \) honest parties \((1 \leq k \leq f+1)\) are activated on \( ID \), the adversary \( A \) cannot fully guess the output. More precisely, consider the predication game: \( A \) guesses a bit \( b^* \) before \( k \) honest parties activated on \( ID \), if \( b^* \) equals to some honest party’s output for \( ID \), we say that \( A \) wins; we require \( Pr[A \text{ wins}] \leq 1 - \alpha/2 \).

Here \( \alpha \) represents the lower-bound probability that all honest parties would output the same bit that is as if uniformly distributed over \( \{0, 1\} \), while \( 1 - \alpha \) captures the possibility that the adversary might predicate/bias the output (which also captures the case that the honest parties output differently).

Intuitively, with a constant probability \( \alpha \), the output would be uniformly flipped. If \( \alpha = 1 \), the Coin is said to be perfect. Many optimal-resilient ABA protocols [8–10] do not necessarily need perfect Coin, and can terminate in expected constant rounds, if \( \alpha \) is a non-zero constant. Also, in some ABA protocols, \( k = f + 1 \) can further clip the asynchronous adversary’s power, so we design a \((n, f, 2f + 1, 1/3)\)-Coin to maximize the strength of our result.

**Tackling the seed of VRF.** Our Coin relies on VRF to let each party commit a uniformly random secret. It is worth mentioning an issue of this primitive: in the bulletin PKI setting, the keys of VRF are generated by each party itself. So if VRFs are evaluated on some pre-determined seeds, a compromised party can use maliciously chosen VRF keys to register at PKI, and biases the distribution of its VRF output.

**Trusted nonce from “genesis”.** In many settings, this might not be an issue, since there could be a trusted nonce generated after all parties have registered their VRF keys (e.g., the same rationale behind the “genesis block” in some Proof-of-Stake blockchains [42]), which can be naturally used as the VRF seed. Such a functionality was earlier formalized by David, Gazi, Kiayias and Russell in [42] as an initialization functionality to output an unpredictable VRF seed.

**Generating VRF seed on the fly.** Nevertheless, we might still expect fewer setup assumptions to get rid of the trusted “genesis block”. Intuitively, we need to handle the issue by generating unpredictable VRF seeds on the fly during the protocol execution. To this end, we put forth a new notion called **reliable broadcasted seeding** (Seeding). Intuitively, the notion can be understood as a “broadcast” version of common coins. That means, it might not terminate if encountering malicious leader, but the honest parties either all receive the common seed, or all receive nothing.

Formally, the Seeding protocol can be defined as follows.
Definition 4 (Reliable Broadcasted Seeding). Seeding is a protocol with two successive committing and revealing phases, and can be defined as follows.

SYNTAX. For each protocol instance with an identifier ID, it has a designated party called leader $P_L$, and is executed among $n$ parties with up to $f$ Byzantine corruptions. Each party takes as input the system’s public knowledge and its private keys, and then sequentially executes the committing phase and the revealing phase, at the end of which it outputs a $\lambda$-bit string seed.

PROPERTIES. It satisfies the next properties with all but negligible probability:

- **Totality.** If some honest party outputs in the Seeding instance associated to ID, then every honest party activated on ID would output.
- **Correctness.** For all ID, if the leader $P_L$ is honest and all honest parties are activated on ID, then all honest parties would output for ID.
- **Committing.** Upon any honest party completes the committing phase and starts to run the revealing phase on session ID, there exists a fixed value seed, such that if any honest party outputs for ID, then it outputs seed.
- **Unpredictability.** Prior to that $k$ honest parties $(1 \leq k \leq f + 1)$ are activated to run the protocol’s revealing phase on session ID, the adversary $A$ cannot predicate the output seed. Namely, $A$ guesses a value seed" before $k$ honest parties are activated on ID, then the probability that seed" = seed shall be negligible, where seed is the output of some honest party for ID.

The totality property ensures: if an honest party gets seed, then all honest parties will do so, despite malicious leader. Moreover, the committing and unpredictability together can ensure that no one can predicate the output before the seed to output is already fixed. This guarantees that VRFs evaluated on the output seed cannot be biased. Intuitively, if the adversary still can bias its own VRF’s output, it must can query the VRF oracle (which performs like a random oracle [42]) with the right seed in a number of polynomial queries (before seed is fixed), which raises a contradiction to break unpredictability.

**Lemma 1.** In the asynchronous message-passing model with bulletin PKI assumption, there exists a Seeding protocol among $n$ parties that can tolerate up to $f < n/3$ static Byzantine corruptions, and terminate in constant asynchronous rounds with $O(n^2)$ messages and $O(\lambda n^2)$ bits, assuming EUF-CMA secure signature and SXDH assumption.

As the above Lemma asserts, we actually can easily construct a Seeding protocol that is centered around the aggregatable non-interactive PVSS scheme that was recently proposed by Gurkan et al. [37] based on SXDH assumption. Here a non-interactive PVSS scheme allows a (possibly malicious) dealer to facilitate verifiable secret sharing via one single script, such that the other parties can decrypt their own secret shares once receiving the PVSS script, and more importantly, they can verify that all parties’ shares are consistent and can recover a unique value. A typical construction of non-interactive PVSS was the Scrape PVSS scheme from Cascudo and David [43]. Gurkan et al. lifted Scrape PVSS to further allow the aggregation of different dealers’ PVSS scripts, such that one can verify that a certain PVSS script indeed commits a secret that is aggregated from which parties’ PVSS scripts. We refer the unfamiliar readers to [37, 43] and our full version [38] for more detailed explanation of PVSS.

So we can employ the aggregatable non-interactive PVSS scheme to implement our Seeding protocol: (i) let each party send an aggregatable PVSS script carrying a random secret to the leader, so the leading party can aggregate them to produce an aggregated PVSS script committing an unpredictable nonce contributed by enough parties (e.g., $2f + 1$); (ii) Then, before recovering the secret hidden behind the aggregated PVSS script, the leader must send it to at least $2f + 1$ parties to collect enough digital signatures to form a “certificate” to prove that
the nonce is fixed and committed to the PVSS script; (iii) Only after seeing such proof, each party would decrypt its corresponding share from the committed PVSS script, thus ensuring the unpredictability and commitment properties. For space limit, the formal description and proofs for our Seeding construction is deferred to our online full version [38].

Constructing private-setup free Coin. With Seeding at hand, we are ready to construct our Coin protocol (as illustrated in Fig. 2) that has four main steps:

1) VRFs sharing. Each party activates a Seeding process as leader and participants in all other Seeding processes to get the seeds. A party $P_i$ activates its own AVSS-Sh instance as dealer to share its VRF evaluation-proof after obtaining its own VRF seed $seed_i$; once obtaining $seed_j$ besides $seed_i$, $P_i$ also activates the corresponding AVSS-Sh instance as a participant.

2) Core-set selecting. Each party $P_i$ records a local set $S_i$ of indices representing the completed AVSS-Sh instances. Once the local set of $P_i$ contains $n - f$ indices, it activates WCS taking $S_i$ as input.

3) VRFs revealing. Once WCS outputs $\hat{S}$, an honest party $P_i$ starts to reconstruct AVSS associated to the indices in $\hat{S}$. After reconstructions, it might get some valid VRF evaluation-proof, and then multicasts the VRF with maximum evaluation to all parties.

4) Largest-VRF amplifying. After receiving $n - f$ valid VRF evaluation-proof pairs, $P_i$ selects the largest VRF evaluation $\hat{r}_i$ and outputs its lowest bit.

Analyzing Coin. The security of the above Coin protocol can be stated by the next main theorem:

Theorem 3. In the bulletin PKI setting and random oracle model, our Coin protocol (formally described in the full version) realizes $(n, f, 2f + 1, 1/3)$-Coin against $n/3$ static Byzantine corruptions in the asynchronous network, conditioned on that the underlying primitives are all secure.

Proof Sketch. This security theorem can be bridged to the hereinafter two key lemmas: one about termination, and one giving the lower-bound possibility of the good cases in which the maximal VRF is evaluated by some honest party and also solicited by the core-set.

Lemma 2. (Termination) If all honest parties are activated on ID, every honest party will decide a speculative largest VRF evaluation $\hat{r}_i$ with a valid proof $\pi_i$, and all honest parties can eventually receive $seed_i$ that $\hat{r}_i$ is evaluated on.

Lemma 3. (Good-case bound) Let $\text{Event}_{\text{good}}$ denote the case in which the $(f+1)$-supporting core-set solicits an honest party’s VRF evaluation that is also largest among all parties’ VRF. Then, $\Pr[\text{Event}_{\text{good}}] \geq \alpha = 1/3$, under the ideal functionality of VRF in [42] (realizable in the RO model with CDH assumption).

Lemma 2 states that every party can get a speculative largest VRF evaluation and thus output since that at least $n - f$ Seedings and $n - f$ AVSSes must complete to ensure the completeness of WCS. While, Lemma 3 bounds the probability of good case and states that with a constant probability $\alpha$, all honest parties could decide the same speculative largest VRF evaluation $\hat{r}_i$ that is also evaluated by some honest party. This implies that with constant probability, the output bit is as uniformly flipped.

In addition, the overall communication complexity of Coin is $O(\lambda n^3)$, which is dominated by WCS, $n$ AVSSes, and $n$ Seedings. Also, the Coin protocol can terminate in constant rounds, because all underlying building blocks would output in constant rounds deterministically. For the sake of completeness, the detailed proof (along with formal description) for the Coin protocol is provided in our online full version [38].

Resulting ABA without private setup. Given our new Coin protocol, one can instantiate efficient asynchronous binary agreement (ABA, cf. [38] for formal definition) with expected constant rounds and cubic bits. We refrain from reintroducing the ABA designs in details, as we only have to plug in Coin to instantiate the needed building block in the ABA protocols from [8] and [10]. More formally, the following theorem holds:

Theorem 4. Given our Coin protocol, the BA protocols in [8] and [10] still can implement ABA in the asynchronous message-passing model with $n/3$ static Byzantine corruption and PKI setup, and they cost expected constant asynchronous rounds, expected $O(n^3)$ messages and expected $O(\lambda n^3)$ bits.

Proof Sketch. The proofs can be found in [8] and [10], respectively. The complexities of resulting ABAs would be dominated by our $(n, f, 2f + 1, 1/3)$-Coin protocol. So both [8] and [10] can terminate in expected constant rounds and exchange only $O(\lambda n^3)$ bits, after adopting our Coin protocol.

V. AUGMENT: TO LEADER ELECTION WITH AGREEMENT

Election in our context aims to randomly elect some party’s identity [1, 2]. The primitive can prevent the adversary from fully predicting which party would be elected. In case it fails, the adversary might schedule message deliveries and cause the higher level protocol to never stop (or slow termination). For example, AMS19 VBA [1] requires Election to luckily draw a completed input broadcast to terminate. So if the adversary can always predicate the Election result in advance, it can delay the to-be-elected broadcast to make it not completed when Election is invoked, and therefore cause VBA grind to a halt.

To this end, we aim at a reasonably fair random leader Election primitive that can be formally defined as:

Definition 5 ((n, f, f + k, \alpha)-Leader Election). A protocol is said to be $(n, f, f + k, \beta)$-Election, if it is among $n$ parties with up to $f$ static Byzantine corruptions, and has syntax and properties as follows.

Syntax. For each protocol instance with session identifier ID, every party takes its own private keys and the system’s public knowledge (i.e., $\lambda$ and all public keys) as input, and outputs an index $l \in [n]$. Properties. Besides the above syntax, Election satisfies the following properties except with negligible probability:
Termination. If all honest parties are activated on ID, every honest party would output some value.

Agreement. For any two honest parties $P_i$ and $P_j$ that output $\ell_i$ and $\ell_j$ for ID, respectively, there is $\ell_i = \ell_j$.

Reasonably fair index-election. Before $k$ honest parties ($1 \leq k \leq f + 1$) are activated on ID, the adversary $A$ cannot always predicate the elected leader. More precisely, consider the predication game: $A$ guesses $\ell^*$ before $k$ honest parties activated on ID, if $\ell^*$ equals to some honest party’s output for ID, we say that $A$ wins; we require $\Pr[A \text{ wins}] \leq 1 - \alpha - \alpha/n$.

Here $\alpha$ represents the lower-bound probability that the output is as if uniformly distributed over $[n]$, while $1 - \alpha$ captures the possibility that the adversary might predicate/bias the output.

Necesity of Agreement. Different from Coin that only realizes agreement with some probability (e.g., 1/3), Election always has agreement. This is necessary in many VBA protocols, because Election is used to decide which party’s input becomes the output, so lacking agreement in Election might directly violate the agreement of VBA. This is the crux to make our Election primitive applicable in all exiting VBA protocols.

Constructing Election. As Fig. 3 illustrates, the starting point of constructing Election is our Coin protocol, at the end of which each party gets a speculative largest VRF. Recall Lemmas 2 and 3 and asserting the essential properties of Coin. Lemma 3 states that with a constant probability $\alpha$, the speculative largest VRF of all honest parties is same. Thus, lifting agreement boils down to the problem of cleaning the possible disagreement in the else $1 - \alpha$ worse cases. Furthermore, Lemma 2 brings to light that all honest parties can get the VRF seeds needed to verify each other’s speculative largest VRF. This hints us all parties can cross-check their local speculative VRFs and then vote on whether a common largest VRF exists.

Taking the above key observations constructively in mind, Election can be designed as follows:

1) Committing the largest VRF for cross-check. Each party firstly runs the code of Coin protocol, and obtains the speculate largest VRF evaluation $r^\ell$ and also $\pi_j$, where $\ell$ represents that this speculative largest VRF is evaluated by which party, and $r^\ell$ and $\pi_j$ are the VRF evaluation and proof, respectively. After that, each party uses RBC to reliably broadcasts $\text{rn}_{\text{max}} = (\ell, r^\ell, \pi_j)$, thus committing this speculative largest VRF evaluation.

2) Voting to clean disagreement. A party can eventually output in $n - f$ RBC, each of which can return a valid VRF evaluation-proof pair (which can be verified because the needed seed can be waited from Seeding protocols). Then, the party checks: if there exists a received VRF output, s.t. its evaluation is the largest and also the majority of all $n - f$ received VRFs. If such a VRF exists, the party inputs 1 to ABA (which can be obtained from section IV), otherwise, it inputs 0 to ABA.

3) Output decision. If ABA outputs 1, then each party waits for that there exists a $(n - f)$-sized subset $G^*$ of VRFs received from RBCs, s.t. some $r^\ast$ is the largest and majority VRF evaluation among all VRFs in $G^*$, then it outputs $(r^\ast \mod n) + 1$. If ABA outputs 0, all parties output a default index, e.g., 1.

Analyzing Election. The security of our Election protocol can be formalized by the next main theorem:

Theorem 5. In the bulletin PKI setting, the above Election construction can realize $(n, f, 2f + 1, 1/3)$-Election (as defined in Definition 5) in the asynchronous message-passing model against $n/3$ static Byzantine corruptions, conditioned on that all underlying primitives are secure.

Proof Sketch. Since all honest parties can obtain the needed seeds for VRF verifications (cf. Lemma 2), they can cross-check each other’s speculative largest VRF, so they can start ABA and then output. When ABA outputs 1, there is a unique VRF satisfying voting rules and all parties can elect the common leader. When ABA outputs 0, it is trivial as the
default leader is elected. In addition, the voting rules do not harm the reasonable fairness of election. Recall that with 1/3 probability, all honest parties have the same speculative largest VRF evaluated by some honest party (cf. Lemma 3). In such good case, ABA returns 1, and the leader is uniformly elected.

Regarding complexity, Election exchanges $O(\lambda n^2)$ bits on average, because each RBC costs $O(\lambda n^2)$ bits and ABA spends expected $O(\lambda n^2)$ bits. Moreover, the Election protocol can terminate in expected $O(1)$ rounds, which is mainly dominated by ABA. We present the detailed proof (as well as the formal pseudocode description) for the Election protocol in the full version [38] for the sake of completeness.

**Resulting VBA without private setup.** Existing VBA protocols (cf. [38] for formal VBA definition) [1, 2] rely on non-interactive threshold PRF (tPRF) [11] to implement a random leader election module. Alternatively, our Election protocol is pluggable in these VBA protocols [1, 2, 12] to replace tPRF, thus removing the possible unpleasant private setup of it. More formally, we have the next theorem for private-setup free thus preserving its constant round complexity. For message complexity, it becomes dominated by the $O(n^3)$ messages of Election. For communication complexity, it is worth noticing that non-interactive threshold signature scheme is used to form short quorum certificates in the 4-staged provable-broadcast (PB) protocols; nevertheless, such instantiation of quorum certificate can be replaced by trivially concatenating digital signatures from $n - f$ distinct parties in the bulletin PKI setting, which only adds an $O(n)$ factor to the size of quorum certificates, thus causing $O(\lambda n^2)$ communication complexity to this private-setup free VBA instantiation (for $\lambda n$-bit input).

VI. APPLICATION SCENARIOS

**Application to asynchronous DKG.** The resulting VBA protocols can be plugged in AJM+21 ADKG [16] to reduce the communication to $O(\lambda n^2)$ bits, with preserving fast termination in expected $O(1)$ rounds and optimal $n/3$ resilience. The basic idea (cf. Section 7.5 in [16]) lets each party multicast an aggregatable PVSS hiding a random secret. Then, everyone gathers and combines $n - f$ PVSS from different parties. So they can input the aggregated PVSS to one VBA instance (with external validity specified to check the input is indeed PVSS aggregated by $n - f$ parties’ contributions). Finally, all honest parties get the same PVSS script returned by VBA and can decrypt to get the key shares. The communication cost now is $O(\lambda n^3)$ as all PVSS scripts are $O(\lambda n)$-bit.

**Application to random beacon w/o DKG.** Our Election protocol can be slightly adapted to realize an asynchronous random beacon service that all participating parties can proceed by consecutive epochs and continually output an unbiased and unpredictable value in each epoch. Here unbiased means the output is uniformly distributed [44, 45] and the unpredictable means that the adversary cannot tell the random output of the next epoch better than guessing, unless $f + 1$ honest parties already output in the current epoch.

To implement asynchronous random beacon, we let all parties execute a sequence of Election with the following minor changes: (i) when ABA returns 0 and thus no largest VRF is agreed, the honest parties directly move into the next Election instance; (ii) instead of returning a short index belonging [n], the parties output the lowest $O(\lambda)$ bits of the selected largest VRF.

These adaptions ensure that a non-default value can output with overwhelming probability after running $k$ Election instances. For bias-resistance, according to the committing and unpredictability of Seeding, the adversary cannot manipulate the generation of VRF seeds so that they cannot bias the VRFs evaluated on the seed or immediately break the unpredictability of VRF. The unpredictability is similar because before $f + 1$ honest parties invoke Seeding protocols, the adversary cannot predate the output VRF seeds.

VII. CONCLUSION

We for the first time systematically study on how to generate common randomness in the asynchronous network with only PKI setup. Efficient common coin and random leader election protocols are designed with only $O(\lambda n^2)$ bits and $O(1)$
rounds in the absence of private setups. These results provide basis to obtain more efficient private-setup free asynchronous Byzantine agreement and might have appealing applications in asynchronous DKG and asynchronous random beacon.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments. Yingzi and Jing were supported in part by the National Key R&D Program of China (No. 2020YFB1005801) and NSFC under Grant 62127396. Yuan was supported in part by the Youth Innovation Promotion Association CAS and NSFC under Grant 62102404. Qiang was supported in part by two Stellar Research Awards and a research gift from Protocol Labs.

REFERENCES

[1] I. Abraham, D. Malkhi, and A. Spiegelman, “Asymptotically optimal validated asynchronous byzantine agreement,” in Proc. PODC 2019, pp. 337–346.
[2] Y. Lu, Z. Lu, Q. Tang, and G. Wang, “Dumbo-ivha: Optimal multi-valued validated asynchronous byzantine agreement, revisited,” in Proc. PODC 2020, pp. 129–138.
[3] I. Keidar, E. Kokoris-Kogias, O. Naor, and A. Spiegelman, “All you need is dag,” in Proc. PODC 2021.
[4] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, “The honey badger of bft protocols,” in Proc. CCS 2016, pp. 31–42.
[5] B. Guo, Z. Lu, Q. Tang, J. Xu, and Z. Zhang, “Dumbo: Faster asynchronous bft protocols,” in Proc. CCS 2020, pp. 803–818.
[6] L. Yang, S. J. Park, M. Alizadeh, S. Kannan, and D. Tse, “DisperseLedger: High-Throughput byzantine consensus on variable bandwidth networks,” in Proc. NSDI 22.
[7] M. J. Fischer, N. A. Lynch, and M. S. Paterson, “Impossibility of distributed consensus with one faulty process,” Massachusetts Inst of Tech Cambridge lab for Computer Science, Tech. Rep., 1982.
[8] A. Mostéfaoui, H. Moumen, and M. Raynal, “Signature-free asynchronous binary consensus with $t < n/3$, $O(n^2)$ messages, and $O(1)$ expected time,” JACM, vol. 62, no. 4, p. 31, 2015.
[9] R. Canetti and T. Rabin, “Fast asynchronous byzantine agreement with optimal resiliency,” in Proc. STOC 1993, pp. 42–51.
[10] T. Crain, “Two more algorithms for randomized signature-free asynchronous binary consensus with $t < n/3$ and $O(n^2)$ messages and $O(1)$ round expected termination,” arXiv preprint arXiv:2002.08768, 2020.
[11] C. Cachin, K. Kursawe, and V. Shoup, “Random oracles in constantino-ple: practical asynchronous byzantine agreement using cryptography,” in Proc. PODC 2000.
[12] C. Cachin, K. Kursawe, P. Petzold, and V. Shoup, “Secure and efficient asynchronous broadcast protocols,” in Annual International Cryptology Conference. Springer, 2001, pp. 524–541.
[13] S. Cohen, I. Keidar, and A. Spiegelman, “Not a coincidence: Sub- quadratic asynchronous byzantine agreement whp,” in 34th International Symposium on Distributed Computing (DISC 2020), 2020.
[14] E. Blum, J. Katz, C.-D. Liu-Zhang, and J. Loss, “Asynchronous byzantine agreement with subquadratic communication,” in Theory of Cryptography Conference, 2020, pp. 353–380.
[15] I. Abraham, M. K. Aguilera, and D. Malkhi, “Fast asynchronous consensus with optimal resiliency,” in International Symposium on Distributed Computing, 2010, pp. 4–19.
[16] I. Abraham, P. Jovanovic, M. Maller, S. Meiklejohn, G. Stern, and A. Tomascu, “Reaching consensus for asynchronous distributed key generation,” in Proc. PODC 2021.
[17] M. O. Rabin, “Randomized byzantine generals,” in 24th Annual Symposium on Foundations of Computer Science (sfc 1983). IEEE, 1983, pp. 403–409.
[18] C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Strobl, “Asynchronous verifiable secret sharing and proactive cryptosystems,” in Proc. CCS 2002, pp. 88–97.
[19] E. Kokoris Kogias, D. Malkhi, and A. Spiegelman, “Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures,” in Proc. CCS 2020, pp. 1751–1767.