Breaking the $O(\sqrt{n})$-Bit Barrier: Byzantine Agreement with Polylog Bits Per Party

Elette Boyle
elette.boyle@idc.ac.il
IDC
Herzliya, Israel

Ran Cohen
rancohen@ccs.neu.edu
Northeastern University
Boston, USA

Aarushi Goel
aarushig@cs.jhu.edu
Johns Hopkins University
Baltimore, USA

ABSTRACT

Byzantine agreement (BA), the task of n parties to agree on one of their input bits in the face of malicious agents, is a powerful primitive that lies at the core of a vast range of distributed protocols. Interestingly, in BA protocols with the best overall communication, the demands of the parties are highly *unbalanced*: the amortized cost is $\tilde{O}(1)$ bits per party, but some parties must send $\Omega(n)$ bits. In best known *balanced* protocols, the overall communication is sub-optimal, with each party communicating $\tilde{O}(\sqrt{n})$.

In this work, we ask whether asymmetry is inherent for optimizing total communication. In particular, is BA possible where *each* party communicates only $\tilde{O}(1)$ bits? Our contributions in this line are as follows:

- We define a cryptographic primitive—*succinctly reconstructed distributed signatures* (SRDS)—that suffices for constructing $\tilde{O}(1)$ balanced BA. We provide two constructions of SRDS from different cryptographic and Public-Key Infrastructure (PKI) assumptions.
- The SRDS-based BA follows a paradigm of boosting from "almost-everywhere" agreement to full agreement, and does so in a single round. Complementarily, we prove that PKI setup and cryptographic assumptions are necessary for such protocols in which every party sends $o(n)$ messages.
- We further explore connections between a natural approach toward attaining SRDS and average-case succinct non-interactive argument systems (SNARGs) for a particular type of NP-Complete problems (generalizing Subset-Sum and Subset-Product).

Our results provide new approaches forward, as well as limitations and barriers, towards minimizing per-party communication of BA. In particular, we construct the first two BA protocols with $\tilde{O}(1)$ balanced communication, offering a tradeoff between setup and cryptographic assumptions, and answering an open question presented by King and Saia (DISC'09).

PODC '21, July 26–30, 2021, Virtual Event, Italy

ACM ISBN 978-1-4503-8548-0/21/07…$15.00
https://doi.org/10.1145/3465084.3467897

CCS CONCEPTS

• **Security and privacy → Cryptography**; • **Theory of computation → Computational complexity and cryptography**.

KEYWORDS

cryptographic protocols; Byzantine agreement; communication complexity

ACM Reference Format:
Elette Boyle, Ran Cohen, and Aarushi Goel. 2021. Breaking the $O(\sqrt{n})$-Bit Barrier: Byzantine Agreement with Polylog Bits Per Party. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC '21), July 26–30, 2021, Virtual Event, Italy*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3465084.3467897

1 INTRODUCTION

The problem of *Byzantine agreement (BA)* [50, 55] asks for a set of n parties to agree on one of their input bits, even facing malicious corruptions. BA is a surprisingly powerful primitive that lies at the core of virtually every interactive protocol tolerating malicious adversaries, ranging from other types of consensus primitives such as broadcast [50, 55] and blockchain protocols (e.g., [22]), to secure multiparty computation (MPC) [5, 21, 39, 56, 60]. In this work, we study BA in a standard context, where a potentially large set of n parties runs the protocol within a synchronous network, and security is guaranteed facing a constant fraction of statically corrupted parties.

Understanding the required communication complexity of BA as a function of n is the subject of a rich line of research. For the relaxed goal of *almost-everywhere agreement* [34], i.e., agreement of all but $o(1)$ fraction of the parties, the full picture is essentially understood. The influential work of King et al. [48] showed a solution roughly ideal in every dimension: in which each party speaks to $\tilde{O}(1)$ other parties (i.e., polylog degree of communication graph, a.k.a. communication *locality* [13]), and communicates a total of $\tilde{O}(1)$ bits throughout the protocol, in $\tilde{O}(1)$ rounds;[1] further, the solution does not require cryptographic and/or trusted setup assumptions and is given in the full-information model. The main challenge in BA thus becomes extending almost-everywhere to full agreement.

In this regime, our current knowledge becomes surprisingly disconnected. While it is known how to employ cryptography and setup assumptions to compute BA with $\tilde{O}(1)$ locality [11, 13, 20], the number of *bits* that must be communicated by each party is large,

[1] We follow the standard practice in large-scale cryptographic protocols, where $\tilde{O}$ hides polynomial factors in $\log n$ and in the security parameter κ, see e.g., [29, 30].

$\Omega(n)$.[2] BA with *amortized* $\tilde{O}(1)$ per-party communication (and computation) can be achieved [1, 15, 22]; however, the structure of these protocols is wildly unbalanced: with some parties who must each communicate with $\Theta(n)$ parties and send $\Omega(n)$ bits. The existence of "central parties" who communicate a large amount facilitates fast convergence in these protocols. When optimizing per-party communication, the best BA solutions degrade to $\tilde{\Theta}(\sqrt{n})$ bits/party, with suboptimal $\tilde{O}(n^{3/2})$ overall communication [45, 47].

This intriguing gap leads us to the core question studied in this paper: Is such an imbalance inherent? More specifically:

Is it possible to achieve Byzantine agreement with (balanced) *per-party communication of* $\tilde{O}(1)$*?*

Before addressing our results, it is beneficial to consider the relevant lower bounds. It is well known that any *deterministic* BA protocol requires $\Omega(n^2)$ communication [33] (and furthermore, the connectivity of the underlying communication graph must be $\Omega(n)$ [32, 35]). This result extends to randomized BA protocols, in the special case of very *strong adversarial* (adaptive, strongly rushing[3]) capabilities [1]. Most closely related is the lower bound of Holtby et al. [41], who showed that without trusted setup assumptions, at least one party must send $\Omega(\sqrt[3]{n})$ messages.[4] But, the bound in [41] applies only to a restricted setting of protocols with *static message filtering*, where every party decides on the set of parties it will listen to before the beginning of each round (as a function of its internal view at the end of the previous round). We note that while the almost-everywhere agreement protocol in [48] falls into the static-filtering model, all other scalable BA protocols mentioned above crucially rely on *dynamic message filtering* (which is based on incoming messages' content). This leaves the feasibility question open.

1.1 Our Results

We perform an in-depth investigation of boosting from almost-everywhere to full agreement with $\tilde{O}(1)$ communication per party. Motivated by the $\tilde{O}(1)$-locality protocol of Boyle et al. [13], we first achieve an intermediate step of *certified almost-everywhere agreement*, where almost all of the parties reach agreement, and, in addition, hold a certificate for the agreed value. [13] showed how to boost certified almost-everywhere agreement to full agreement in a single round, where every party communicates with $\tilde{O}(1)$ parties.

Our initial observation is that the protocol from [13] achieves low communication aside from one expensive piece: the distributed generation of the certificate, which is of size $\Theta(n)$, and its dissemination. We thus target this step and explore.

Our contributions can be summarized as follows.

- **SRDS and balanced BA.** We define a minimal ad-hoc cryptographic primitive whose existence implies $\tilde{O}(1)$ balanced BA: *succinctly reconstructed distributed signatures* (SRDS). We provide two constructions of SRDS, each based on a different flavor of a public-key infrastructure (PKI): (1) from one-way functions in a "trusted-PKI" model, and (2) from collision-resistant hash functions (CRH) and a strong form of cryptographic succinct non-interactive arguments of knowledge (SNARKs)[5] in a model with a "bare PKI" and a common random string (CRS). Roughly, trusted-PKI setup assumes that parties' keys are generated properly, whereas bare PKI further supports the case where corrupt parties may generate keys maliciously. We elaborate on the difference between the PKI models in Section 1.2.
- **Necessity of setup for one-shot "boost."** Our SRDS-based BA follows a paradigm of boosting from almost-everywhere to full agreement, and does so in a single communication round. Complementarily, we prove two lower bounds for any such one-shot boost in which every party sends $o(n)$ messages. The first shows that some form of PKI (or stronger setup, such as correlated randomness[6]) is *necessary* for this task. The second shows that given only PKI setup (as opposed to stronger, correlated-randomness setup), then *computational assumptions* (namely, at least one-way functions) are additionally required.
  In contrast to prior lower bounds (e.g., [1, 41]), this holds even against a static adversary, and where parties can exercise dynamic filtering (i.e., without placing limitations on how parties can select to whom to listen).
- **Connections to succinct arguments.** We further explore connections between a natural approach toward attaining SRDS in weaker PKI models and *average-case* succinct non-interactive argument (SNARG) systems[7] for a particular type of NP-Complete problems (generalizing Subset-Sum and Subset-Product). This can be interpreted as a barrier toward this approach for constructing SRDS without heavy "SNARG-like" tools.

Collectively, our results provide an initial mapping for the feasibility landscape of BA with $\tilde{O}(1)$ per-party communication, including new approaches forward, as well as limitations and barriers. Our approach yields two BA protocols with $\tilde{O}(1)$ communication per party, offering a tradeoff between the setup assumptions and the cryptographic assumptions. These results answer an open question presented by King and Saia [46], asking whether cryptography can be used to construct BA with $o(\sqrt{n})$ communication per party. Our BA results are summarized in Table 1 alongside other almost-everywhere to everywhere agreement protocols.

1.2 Technical Overview

We now proceed to present our results in greater detail.

Succinctly reconstructed distributed signatures. Our first contribution is identifying and formalizing a cryptographic primitive that enables boosting from almost-everywhere agreement to full agreement on a value, with low per-party communication.

[2] In fact, the constructions in [11, 13, 20] are for MPC protocols that enable secure computation of any function with $\tilde{O}(1)$ locality; these protocols are defined over point-to-point networks, and so also provide a solution for the specific task of BA.

[3] A *strongly rushing* adversary in [1] can adaptively corrupt a party that has sent a message m and replace the message with another m', as long as no honest party received m.

[4] The lower bound in [41] easily extends to a public setup such as a common reference string.

[5] A SNARK [6, 53] is a proof system that enables a prover holding a witness w to some public NP statement x to convince a verifier that it indeed knows w by sending a single message. The proof string is succinct in the sense that it is much shorter than the witness w, and knowledge is formalized via an efficient extractor that succeeds extracting w from a malicious prover P^* with roughly the same probability that P^* convinces an honest verifier.

[6] In the *correlated-randomness* model a trusted dealer samples n secret strings from a joint distribution and delivers to each party its corresponding secret string, e.g., a setup for threshold signatures.

[7] Similarly to a SNARK, a SNARG allows a prover holding a witness w to some public NP statement x to convince a verifier that x belongs to the language; however, as opposed to a SNARK, here the prover does not prove that it knows w (only that such a witness exists), hence there is no requirement to extract the witness from a cheating prover.

protocol	rounds	max com. per party	setup	cryptographic assumptions	message filtering	adversarial corruptions	remark
HKK'08 [41]		$\Omega(\sqrt[3]{n})$	crs		static	static	lower bound
KS'09 [46]	$O(1)$	$\tilde{O}(n \cdot \sqrt{n})$	-	-	dynamic	static	
KS'11 [47]	polylog(n)	$\tilde{O}(\sqrt{n})$	-	-	dynamic	adaptive	
KLST'11 [45]	polylog(n)	$\tilde{O}(\sqrt{n})$	-	-	dynamic	static	
BGH'13 [15]	$O(1)$	$\tilde{O}(n)$	-	-	dynamic	static	
BGT'13 [13]	1	$\tilde{O}(n)$	pki	owf	dynamic	static	
CM'19 [22]†	Exp $O(1)$	$\tilde{O}(n)$	trusted-pki	RO+unique-sig	dynamic	adaptive	
ACD^{+}'19 [1]†	Exp $O(1)$	$\tilde{O}(n)$	trusted-pki	bilinear maps	dynamic	adaptive	
CKS'20 [26]†	Exp $O(1)$	$\tilde{O}(n)$	trusted-pki	vrf	dynamic	adaptive	asynchronous
BKLL'20 [8]†	Exp $O(1)$	$\tilde{O}(n)$	trusted-pki	fhe+nizk	dynamic	adaptive	asynchronous
	1	$\Omega(n)$	crs		dynamic	static	lower bound
This work	1	$\tilde{O}(1)$	pki+crs	snarks*+crh	dynamic	static	
	1	$\tilde{O}(1)$	trusted pki	owf	dynamic	static	

Table 1: Comparison of protocols boosting from almost-everywhere to full agreement, tolerating $(1/3 - \epsilon) \cdot n$ corruptions. The $\tilde{O}$ notation hides polynomial terms in the security parameter κ and in $\log n$. *crs* stand for a common random string, *pki* stands for bare pki, and *trusted pki* stands for honestly generated pki. By *snarks** we refer to SNARKs with linear extraction, i.e., where the size of the extractor is linear in the size of the prover. *RO* stands for random oracle and *unique-sig* for unique signatures. *vrf* stand for verifiable pseudorandom functions, *fhe* for fully homomorphic encryption, and *nizk* for non-interactive zero-knowledge proofs. Static corruptions are done before the protocol begins but can be a function of the trusted setup; adaptive corruptions can occur during the course of the protocol. $^{(\dagger)}$ The protocols from [1, 8, 22, 26] reach agreement from scratch (hence also from almost-everywhere agreement) with amortized $\tilde{O}(1)$ communication per party; the expected round complexity is constant and termination is guaranteed in polylog(n) rounds.

The primitive—*succinctly reconstructed distributed signatures* (SRDS)—is a new type of a distributed signature scheme, with a natural motivation: allowing a set of parties to jointly produce a signature on some message m, which can serve as a succinct certificate for proving that a *majority* of the parties agree on m. Interestingly, this task does not seem to be attained by existing distributed signature notions, such as *multi-signatures* [42], *aggregate signatures* [10], or *threshold signatures* [31]. For example, while multi-signatures (and, similarly, aggregate signatures) can succinctly combine signatures of many parties, to *verify* the signature, the (length-$\Theta(n)$!) vector of contributing-parties identities must also be communicated.[8] Threshold signatures are implied by SRDS but also do not suffice: while identities of the signers are no longer needed to verify a combined signature, this information is necessary to *reconstruct* the combined signature in the first place (even within specific existing schemes, e.g., [9, 37]).

An SRDS scheme is based on a PKI for signatures, where every party is set with a secret signing key and a public verification key.[9] The parties may receive additional setup information that may contain, for example, public parameters for the signature scheme or a common random string (CRS), depending on the actual construction. Given a message m, every party can locally generate a signature on m, and signatures on the same message can be succinctly aggregated into a new signature. The new aspect is that given a combined signature and a message m, it is possible to verify whether is was aggregated from a "large" number of "base" signatures on m, and both aggregation and verification can be done *succinctly*.

[8] Indeed, the verification algorithm of multi-signatures (and aggregate signatures) must receive the set of parties who signed the message. This is precisely the culprit for the large $\tilde{\Theta}(n)$ per-party communication within the low-locality protocol of [13].

[9] As mentioned, we will distinguish between a *bare PKI*, where every party locally chooses its keys and corrupted parties can set their keys as a function of all verification keys (and any additional public information), and a *trusted PKI*, which is honestly generated (either locally or by a trusted party) and where corrupted parties cannot change their verification keys. See further discussion below.

Three properties are required from an SRDS scheme: *robustness* means that an adversary cannot prevent the honest parties from generating an accepting signature on a message; *unforgeability* prevents an adversary controlling a minority from forging a signature; and *succinctness* requires that the "final" signature (including *all* information needed for verification) is short (of size $\tilde{O}(1)$) and can be incrementally reconstructed from "base" signatures in small batches of size polylog(n).[10] An SRDS scheme is *t-secure* if it satisfies the above properties even facing t colluding adversarial parties. As mentioned earlier, we present two constructions of SRDS in Section 2.2, offering a tradeoff between setup assumptions and cryptographic assumptions.

Balanced BA from SRDS. We demonstrate how to attain $\tilde{O}(1)$-balanced BA against βn corruptions (for $\beta < 1/3$) given black-box access to any βn-secure SRDS scheme. We begin by presenting a distilled version of the "certified almost-everywhere agreement" approach from [13] that we tailor for Byzantine agreement, where only correctness matters and privacy is not required.[11]

(1) The parties execute the almost-everywhere agreement protocol of King et al. [48]; this establishes a polylog(n)-degree communication tree (which is essentially a sparse overlay network) in which each node is assigned with a committee of polylog(n) parties. The guarantees are that the polylog(n)-size *supreme committee* (i.e., the committee assigned to the root) has a $2/3$ honest majority and almost all of the parties are connected to the supreme committee via the communication tree.
(2) The supreme committee executes a BA protocol on their inputs to agree on the output y, and, in addition, runs a coin-tossing

[10] polylog(n) denotes $\log^{c}(n)$ for some constant $c > 1$.

[11] The focus of [13] was on MPC and required stronger assumptions and additional rounds; in particular, a naïve use of their MPC protocol *cannot* lead to communication-balanced BA as it requires all parties to send information to a designated polylog(n)-size set, the so-called *supreme committee*.

protocol to agree on a random seed s. Next, the supreme committee propagates the pair (y, s) to *almost* all of the parties.

(3) Once a party receives the pair (y, s), the party signs it (in [13], using a multi-signature scheme), and sends the signature back to the supreme committee that aggregates all the signatures. The aggregated signature attesting to (y, s) is then distributed to *almost* all of the parties.

Once this form of *certified* almost-everywhere agreement on (y, s) is reached, full agreement can be obtained in one round using the same approach as in [13] (see Section 3 for details). The protocol from [13] achieves $\tilde{O}(1)$ locality. However, recall that even though the size of a multi-signature might itself be "small," the verification algorithm additionally requires a list of contributing parties, where the description size of this list will need to be proportional to n. Hence, the effective size of the aggregated signature, and thus per-party communication, is stuck at $\Theta(n)$.

At this point the new notion of SRDS comes into the picture. We use the *succinctness* property of SRDS combined with the communication tree established by the protocol from [48] to bound the size of the aggregated signatures by $\tilde{O}(1)$. In essence, the parties aggregate the signatures in a recursive manner up the communication tree such that in each step at most $\text{polylog}(n)$ signatures are aggregated.

This technique introduces additional subtleties that must be addressed. For example, since the partially aggregated signature can no longer afford to describe the set of contributing parties, it is essential to make sure that the same "base" signature is not aggregated multiple times (this may allow the adversary to achieve more influence on the final aggregated signature than its proportional fraction of "base" signatures).

Theorem 1.1 (balanced BA, informal). *Let $\beta < 1/3$ be a constant. Assuming the existence of βn-secure SRDS, there exists an n-party, βn-resilient BA protocol that terminates after $\text{polylog}(n)$ rounds, and where every party communicates $\text{polylog}(n) \cdot \text{poly}(\kappa)$ bits.*

We note that our BA protocol is the first to establish a $\text{polylog}(n)$-degree communication graph where *every* party has an "honest path" to a 2/3-honest committee, such that the per-party communication required for establishing it is $\tilde{O}(1)$. Such a communication graph can be used to design a broadcast protocol where the total communication is just $\tilde{O}(1)$. It can also be used to design secure multiparty computation protocols (MPC) that scale well with the number of parties using fully homomorphic encryption (encryption schemes that allow computation over ciphertexts). As a result, we can obtain the following corollaries (we defer the proof to the full version of our paper [12]).

Corollary 1.2 (informal). *Let $\beta < 1/3$ be a constant. Assuming the existence of βn-secure SRDS:*

(1) **Broadcast***: There exists a βn-resilient 1-bit broadcast protocol such that ℓ protocol executions (potentially with different senders) require $\ell \cdot \text{polylog}(n) \cdot \text{poly}(\kappa)$ bits of communication per party.*

(2) **MPC***: Assuming fully homomorphic encryption, a function $f : (\{0,1\}^{\ell_{\text{in}}})^n \to \{0,1\}^{\ell_{\text{out}}}$ can be securely computed with guaranteed output delivery tolerating a static, malicious βn-adversary, such that the total communication complexity (of all parties) is $n \cdot \text{polylog}(n) \cdot \text{poly}(\kappa) \cdot (\ell_{\text{in}} + \ell_{\text{out}})$ bits.*

One remark regarding the corruption model is in place. In this work we consider *static* adversaries that choose the set of corrupted parties before the beginning of the protocol. As mentioned above, our constructions are based on some form of trusted setup, which, as we prove below, is necessary. We emphasize that (as standard) we avoid trivialized settings, e.g., where the trusted setup determines a $\text{polylog}(n)$-degree communication tree for achieving *full* agreement, by considering the adversarial model where the adversary can corrupt the parties adaptively during the setup phase *given* the setup information of the corrupted parties and any public setup information. During the online phase the adversary is static and cannot corrupt additional parties.

Necessity of PKI for single-round boost of almost-everywhere agreement. Our SRDS-based BA protocol (Theorem 1.1) shows how to boost almost-everywhere agreement to full agreement in a single round with small communication. Both our constructions crucially rely on a public-key infrastructure (PKI) that enables each party to publish its verification key on a bulletin board. We show that this setup assumption is necessary for this task. That is, given *only* public setup—i.e., the common reference string model—this task is not possible.

We note that the lower bound of Holtby et al. [41] does not translate to our setting, as it considers *static* message filtering, where every party chooses to whom to listen in a given round based on its view prior to that round. The lower bound in [41] shows that *dynamic* filtering, i.e., where filtering can also be based on the content of received messages, is required (at least in the CRS model). We present the first such lower bound in the dynamic-filtering model.

Theorem 1.3 (no single-shot boost in CRS model, informal). *There is no single-round protocol from almost-everywhere to everywhere agreement in the CRS model where every party sends sublinear (i.e., $o(n)$) many messages.*

Recall that almost-everywhere agreement guarantees that all parties agree on the common output aside from a $o(n)$-size set of *isolated* parties, whose identities are unknown to the remaining honest parties. In the setting of static filtering, one can prove continued isolation of these parties for any low-communication protocol in a relatively clean manner [41]: The probability that an honest party P_i will send messages to an honest isolated P_j is *independent* of the event that P_j will choose to process messages from P_i in this round, thus placing a birthday-type bound on information successfully being conveyed. With dynamic filtering, however, P_j may process messages *dependent* on some property of this message, e.g., whether it contains particular authentication, which may only be contained in honest messages.[12] In such case, there is strong bias toward accepting honest messages, and one must work harder to ensure that isolated parties do not reach agreement.

We refer the reader to the full version of our paper [12] for the proof of this theorem.

On the different PKI models. As discussed above, SRDS implies a single-round boost of almost-everywhere to full agreement, which in turn (by Theorem 1.3) requires some form of private-coin setup. Given this, one of our goals is to minimize the trust assumptions in

[12] In general, message filtering should be via a simple and "light" test, e.g., counting how many messages arrived, or verifying a signature. We refer to [11] for a discussion on message filtering in protocols over incomplete graphs.

the setup phase. Our SNARK-based construction offers the minimal setup requirement—a bare PKI—where every party locally generates its own signature keys and publishes the verification key on a bulletin board. The adversary can adaptively corrupt parties and change their keys as a function of all the public setup information (including the honest parties' verification keys and the CRS, in case it exists). This is the prevalent PKI model that has appeared in, e.g., [16, 17, 43, 44].

Our OWF-based construction, on the other hand, assumes an honestly generated PKI, where the adversary cannot alter the corrupted parties' keys. Such a setup assumption is normally captured by a trusted party who samples the keys for all the parties, and provides each party with its secret key as well as all public keys; see, e.g., [1, 18, 19, 25, 51]. We note that our trusted-PKI setup is *weaker* than a full blown trusted party in two aspects: First, the distribution from which the trusted party samples the values is a *product distribution*, i.e., parties' keys are independent and second, we consider *public-coin* sampling in the sense that the sampling coins are revealed to the corresponding party (i.e., intermediate key-generation values are not kept hidden).

Necessity of OWF for single-round boost in PKI model. Theorem 1.3 states the necessity of private-coin setup for single-round protocols (from almost-everywhere agreement to full agreement) where every party sends $o(n)$ messages. In the PKI model, where the public/private keys of each party are independently generated, we further prove that *cryptographic assumptions* are necessary. Intuitively, if one-way functions (OWF) do not exist, an adversary can invert the PKI algorithm with noticeable probability to find a preimage for each public key. In this case, the adversary can carry out the attack for the CRS model, discussed above. We prove the following theorem in our full version [12].

Theorem 1.4 (OWF needed for single-shot boost in PKI model, informal). *If OWF do not exist, there is no single-round protocol from almost-everywhere to everywhere agreement in the trusted PKI model where every party sends sublinear many messages.*

Connection to succinct arguments. Our SRDS construction from CRH and SNARKs works with minimal setup requirements, but relies on relatively undesirable cryptographic assumptions (in particular, SNARKs are a non-falsifiable [38] assumption). On the other hand, our construction from one-way functions uses light computational assumptions, but (as with many other works in this area, e.g., [1, 18, 19, 25]) requires a stronger assumption of trusted PKI. A clear goal is to obtain SRDS from better computational assumptions within a better setup model, ultimately reducing to bare PKI, or even more fine-grained intermediate models such as *registered PKI*[13] (see [9, 52] and a discussion in [4]). A natural approach toward doing so is to build upon one of the closest existing relatives within this setting: *multi-signatures*.

Recall that multi-signatures *almost* provide the required properties of SRDS in this setting, in that they support succinct aggregation of signatures, with the sole issue that multi-signature verification requires knowledge of the set of parties who contributed to it—information that requires $\Theta(n)$ bits to describe. Multi-signatures have been constructed from (standard) falsifiable assumptions in the registered-PKI model, e.g., [52]. A natural approach toward constructing SRDS within this model is thus to simply augment a multi-signature scheme with some method of succinctly convincing the verifier that a given multi-signature is composed of signatures from sufficiently many parties. In the full version [12], we demonstrate challenges toward such an approach, by showing that in some cases this *necessitates* a form of cryptographic succinct non-interactive arguments.

[13] In the registered PKI, every party can arbitrarily choose its public key (just like in bare PKI), but in order to publish it, the party must prove knowledge of a corresponding secret key.

2 SUCCINCTLY RECONSTRUCTED DISTRIBUTED SIGNATURES

In this section, we introduce a new notion of a distributed signature scheme for n parties, which can be used to obtain low-communication BA. As discussed earlier, every party has signing/verification keys based on some form of PKI, and the parties may receive additional setup information consisting of public parameters for the underlying signature scheme and potentially a common random string (CRS). We allow the adversary to adaptively corrupt a subset of the parties before the protocol begins, based on the setup information and all n verification keys. We consider two PKI models; a *bare PKI*, where the adversary can choose the corrupted parties' keys, and a *trusted PKI*, where the keys are honestly generated and cannot be changed. We do not permit adaptive corruptions once the parties start signing messages.

In Section 2.1, we define the new signature scheme and the security requirements. Later, in Section 2.2, we present two constructions of this primitive.

2.1 Definition

We start by presenting the syntax of the definition, and later, define the required properties from the scheme: succinctness, robustness, and unforgeability.

Definition 2.1 (SRDS syntax). *A succinctly reconstructed distributed signatures scheme with message space $\mathcal{M}$ and signature space $\mathcal{X}$ for a set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, is defined by a quintuple of PPT algorithms* (Setup, KeyGen, Sign, Aggregate, Verify) *as follows:*

- $\mathsf{Setup}(1^\kappa, 1^n) \to \mathsf{pp}$*: On input the security parameter κ and the number of parties n, the setup algorithm outputs public parameters* pp.
- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{vk}, \mathsf{sk})$*: On input the public parameters* pp*, the key-generation algorithm outputs a verification key* vk *and a signing key* sk.
- $\mathsf{Sign}(\mathsf{pp}, i, \mathsf{sk}, m) \to \sigma$*: On input the public parameters* pp*, the signer's identity i, a signing key* sk*, and a message $m \in \mathcal{M}$, the signing algorithm outputs a signature $\sigma \in \mathcal{X} \cup \{\perp\}$.*
- $\mathsf{Aggregate}(\mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, m, \{\sigma_1, \ldots, \sigma_q\}) \to \sigma$*: On input the public parameters* pp*, the set of all verification keys $\{\mathsf{vk}_i\}_{i\in[n]}$, a message $m \in \mathcal{M}$, and a set of signatures $\{\sigma_i\}_{i\in[q]}$ for some $q = \mathsf{poly}(n)$, the aggregation algorithm outputs a signature $\sigma \in \mathcal{X} \cup \{\perp\}$.*
- $\mathsf{Verify}(\mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, m, \sigma)) \to b$*: On input the public parameters* pp*, the set of all verification keys $\{\mathsf{vk}_i\}_{i\in[n]}$, a message $m \in \mathcal{M}$, and a signature $\sigma \in \mathcal{X}$, the verification algorithm outputs a bit $b \in \{0, 1\}$, representing accept or reject.*

We assume without loss of generality that each signature encodes the index i of the corresponding verification key vk_i, and each aggregated signature encodes information about the maxima and minima of the indices associated with verification keys corresponding to the base signatures that are aggregated within them.[14] Given a base signature/aggregated signature, let $\max(\sigma)$ denote the function that extracts the maximum index associated with σ and $\min(\sigma)$ denote the function that extracts the minimum index associated with σ (in case of a base signature, both $\max(\sigma)$ and $\min(\sigma)$ will return the same value).

Remark (Notation n). Here, we use n to denote the number of parties in the SRDS scheme. Looking ahead, the effective number of parties in the SRDS used in our BA protocol in Section 3 will be larger than the actual participants of the protocol.

We proceed to define three properties of an SRDS scheme: *succinctness*, *robustness*, and *unforgeability*. We define these properties with respect to any $t < n/3$ corruptions. Although the definitions can be stated for $t < n/2$, we opted for the former for clarity and concreteness, as both our BA protocol (Section 3) and our SRDS constructions (Section 2.2) support $n/3$ corruptions.

Succinctness. We require that the size of each signature is $\tilde{O}(1)$. This holds both for signatures in the support of Sign and of Aggregate. In order for parties to jointly perform the signature aggregation process with low communication, we also require that the aggregate algorithm can be decomposed into two algorithms $\mathsf{Aggregate}_1$ and $\mathsf{Aggregate}_2$. Depending on the set of input signatures $\{\sigma_i\}_{i\in[q]}$ and the verification keys, the first algorithm $\mathsf{Aggregate}_1$ *deterministically* outputs a subset of the signatures S_{sig}. The second (possibly randomized) algorithm $\mathsf{Aggregate}_2$ then aggregates these signatures without relying on the verification keys. In particular, the input to the randomized step $\mathsf{Aggregate}_2$ is short.

Looking ahead at the BA protocol in Section 3, subsets of the parties will collectively run the aggregation algorithm. Although the inputs to the aggregation algorithm need not be kept private, it could be the case that the randomness used should remain secret. For this reason, the computation of $\mathsf{Aggregate}_2$ in the BA construction will be carried out using an MPC protocol; to keep the overall communication of every party $\tilde{O}(1)$, we require the circuit size representing $\mathsf{Aggregate}_2$ to be $\tilde{O}(1)$. The goal of $\mathsf{Aggregate}_1$ is to deterministically filter out invalid inputs (using the verification keys), such that $\mathsf{Aggregate}_2$ only depends on the verified signatures and *not* on the n verification keys (otherwise the circuit size will be too large).

Definition 2.2 (succinctness). *An n-party SRDS scheme is succinct if it satisfies the following:*

(1) ***Size of Signatures:*** *There exists $\alpha(n,\kappa) \in \mathsf{poly}(\log n, \kappa)$ such that $\mathcal{X} \subseteq \{0,1\}^{\alpha(n,\kappa)}$.*

(2) ***Decomposability:*** *The* Aggregate *algorithm can be decomposed into 2 algorithms* $\mathsf{Aggregate}_1$ *and* $\mathsf{Aggregate}_2$*, such that the following hold:*

- $\mathsf{Aggregate}_1(\mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, m, \{\sigma_1, \ldots, \sigma_q\}) \rightarrow S_{\mathsf{sig}}$*, where S_{sig} is of size* $\mathsf{poly}(\log n, \kappa)$ *and* $\mathsf{Aggregate}_1$ *is deterministic.*
- $\mathsf{Aggregate}_2(\mathsf{pp}, m, S_{\mathsf{sig}}) \rightarrow \sigma$*, i.e., aggregate the signatures in S_{sig} into a new signature σ.*

Robustness. Informally, a scheme is robust if no adversary can prevent sufficiently many honest parties from generating an accepting signature on a message. We define robustness as a game between a challenger and an adversary $\mathcal{A}$. The game is formally defined in Figure 1 and comprises of three phases. In the *setup and corruption* phase, the challenger generates the public parameters pp and a pair of signature keys for every party. Given pp and all verification keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$, the adversary can adaptively corrupt a subset of (up to) t parties and learn their secret keys. In the case of a bare PKI (but *not* of trusted PKI), the adversary can replace the verification key of any corrupted party by another key of its choice. Unless specified otherwise, we consider the bare PKI to be the default setup model.

In the *robustness challenge* phase, the adversary chooses a tree T describing the order in which the signatures of all the parties are to be aggregated. The nodes on level 0 correspond to set of all parties who generate signatures (i.e., all virtual parties in the BA protocol). We slightly abuse notation and refer to level-1 nodes as leaf nodes, as they correspond to the actual leaves in the communication tree of [48]. For our application in the BA protocol in Section 3, we require this tree to be an "$(n, \mathcal{I})$-party almost-everywhere-communication tree" (see Definition 2.3), where n is the number of parties and $\mathcal{I}$ is the set of corrupt parties.[15] Furthermore, we assume that level-0 nodes are indexed and ordered by the parties in such a way that when the tree topology is expressed flat as a planar graph (no crossovers), then the IDs of level-0 nodes are in increasing order. Looking ahead, we will show that this property of the tree is sufficient for our BA protocol in Section 3. The adversary also chooses messages $m \in \mathcal{M}$ and $\{m_i\}_{i\in\mathcal{N}}$, where $\mathcal{N}$ is the subset of honest parties that are assigned to leaf nodes that do not have a *good path* (i.e., where more than a third of the parties assigned to at least one of the nodes on the path are corrupt) to the root.

Given signatures of parties in $\mathcal{N}$ on the respective m_i's and of the remaining honest parties on m, the adversary computes signatures of all corrupt parties. The challenger and adversary then interactively aggregate all these signatures in the order specified by the tree T. In particular, partially aggregated signatures corresponding to intermediate nodes in the tree that consist of a majority of honest parties, are computed by the challenger, while partially aggregated signatures corresponding to the remaining nodes are chosen by the adversary.

Finally, in the *output* phase, the challenger runs the verification algorithm on the message m and the final aggregated signature obtained in the root of the tree, and $\mathcal{A}$ wins if the verification fails. We say that an SRDS scheme is robust if no adversary can win this game except with negligible probability.

We start by formally describing the properties of an $(n, \mathcal{I})$-party almost-everywhere-communication tree, which is a slight variant of the tree described in King et al. [48].

[14] Both our constructions presented in Section 2.2 achieve this property.

[15] This tree is a combinatorial object that was first defined by King et al. [48]. They also proposed an interactive protocol that allows the parties to collectively build such a tree on the fly. This tree and that protocol will be an integral part of our BA protocol in Section 3.

Definition 2.3 ($(n, \mathcal{I})$-party almost-everywhere-communication tree). *Let $\mathcal{I} \subseteq [n]$ be a subset of size t for $t < n/3$. A directed rooted tree $T = (V, E)$ is an $(n, \mathcal{I})$-party almost-everywhere-communication tree if the following properties are satisfied:*

1. *The height of T is $\ell^* \in O(\log n / \log\log n)$. Each node v from level $\ell > 1$ has $\log n$ children in level $\ell - 1$.*
2. *Each node on level $\ell > 1$ is assigned a set of $\log^3 n$ parties.*
3. *A node is* good *if less than a third of the parties assigned to it are in $\mathcal{I}$. Then, it holds that the root is good.*
4. *All but a $3/\log n$ fraction of the leaves have a* good path *(consisting of good nodes) to the root.*
5. *The nodes on level 0 correspond to the n parties.*
6. *Each party (on level 0) is assigned to exactly one leaf node (on level 1).*
7. *There are $n/\log^5 n$ leaf nodes and each leaf node is assigned a set of $\log^5 n$ parties.*

Experiment $\mathsf{Expt}^{\mathsf{robust}}_{\mathsf{mode},\Pi,\mathcal{A}}(\kappa, n, t)$

The experiment $\mathsf{Expt}^{\mathsf{robust}}$ is a game between a challenger and the adversary $\mathcal{A}$. The game is parametrized by an SRDS scheme Π and proceeds as follows:

A. **Setup and corruption.** In the first phase, the challenger generates the public parameters and the signature keys for the parties. Given the public information, $\mathcal{A}$ can adaptively corrupt parties, learn their secret information, and potentially change their public keys.
   (1) Compute $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa, 1^n)$.
   (2) For every $i \in [n]$, compute $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$.
   (3) Invoke $\mathcal{A}$ on $(1^\kappa, 1^n, \mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\})$ and set $\mathcal{I} = \emptyset$.
   (4) As long as $|\mathcal{I}| \le t$ and $\mathcal{A}$ requests to corrupt a party P_i:
      (a) Send sk_i to $\mathcal{A}$ and receive back vk'_i.
      (b) If mode = b-pki, set $\mathsf{vk}_i = \mathsf{vk}'_i$.
      (c) Set $\mathcal{I} = \mathcal{I} \cup \{i\}$.

B. **Robustness challenge.** In this phase, $\mathcal{A}$ tries to break the robustness of the scheme.
   (1) $\mathcal{A}$ chooses an $(n, \mathcal{I})$-party almost-everywhere-communication tree $T = (V, E)$ (as per Definition 2.3), in which level-0 nodes indexed and ordered by the parties in such a way that when the tree topology is expressed flat as a planar graph (no crossovers), then the IDs of level-0 nodes are in increasing order. Let $\mathcal{N}$ be the set of honest parties assigned to the leaf nodes that do not have a good path to the root.
   (2) $\mathcal{A}$ also chooses a message $m \in \mathcal{M}$ and a message $m_i \in \mathcal{M}$ for each $i \in \mathcal{N}$.
   (3) For every $i \in [n] \setminus (\mathcal{I} \cup \mathcal{N})$, let $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{pp}, i, \mathsf{sk}_i, m)$ and for every $i \in \mathcal{N}$, let $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{pp}, i, \mathsf{sk}_i, m_i)$.
   (4) Send $\{\sigma_i\}_{i \in [n] \setminus \mathcal{I}}$ to $\mathcal{A}$ and receive back $\{\sigma_i\}_{i \in \mathcal{I}}$.
   (5) For each $\ell = \{2, \ldots, \mathsf{height}(T)\}$ and every node v on level ℓ:
      - If v is a good node, compute
        $$\sigma_v \leftarrow \mathsf{Aggregate}(\mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, m, \{\sigma_u\}_{u \in \mathsf{child}(v)}),$$
        where $\mathsf{child}(v) \subseteq V$ refers to the set of children of the node $v \in V$, and send σ_v to $\mathcal{A}$.
      - Else, if v is a bad node, receive σ_v from $\mathcal{A}$.

C. **Output Phase.** Output $\mathsf{Verify}(\mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, m, \sigma_{\mathsf{root}})$, where root is the root node in T.

Figure 1: Robustness experiment for SRDS

Definition 2.4 (Robustness). *Let $t < n/3$. An SRDS scheme Π is t-robust with a bare PKI (resp., with a trusted PKI) if for* mode = b-pki *(resp.,* mode = tr-pki*) and for any (stateful) PPT adversary $\mathcal{A}$ it holds that:*

$$\Pr\left[\mathsf{Expt}^{\mathsf{robust}}_{\mathsf{mode},\Pi,\mathcal{A}}(\kappa, n, t) = 0\right] \le \mathsf{negl}(\kappa, n).$$

The experiment $\mathsf{Expt}^{\mathsf{robust}}_{\mathsf{mode},\Pi,\mathcal{A}}$ is defined in Figure 1.

We note that *robustness* is a strictly stronger notion than *completeness*. In a complete scheme correctness is guaranteed if all the parties are honest. In a robust scheme, even if a subset of parties are corrupted, as long as there are sufficiently many honest parties, correctness is still guaranteed. Hence, any signature scheme satisfying robustness, immediately satisfies completeness.

Unforgeability. Informally, a scheme is unforgeable if no adversary can use signatures of a large majority of the honest parties on a message m and of a few honest parties on messages of its choice to forge an aggregated SRDS signature on a message other than m.

In a similar way to robustness, we consider an unforgeability game between a challenger and an adversary. The *setup and corruption* phase is identical to that in the robustness game. In the *forgery challenge* phase, the adversary chooses a set $\mathcal{S} \subseteq [n] \setminus \mathcal{I}$ such that $|\mathcal{S} \cup \mathcal{I}| < n/3$, and messages m and $\{m_i\}_{i \in \mathcal{S}}$. Given signatures of all honest parties outside of $\mathcal{S}$ on the message m and a signature of each honest party P_i in $\mathcal{S}$ on the message m_i, the adversary outputs a signature σ. In the *output* phase, the challenger checks whether σ is a valid signature on a message different than m; if so, the adversary wins. An SRDS scheme is unforgeable if no adversary can win the game except for negligible probability.

Definition 2.5 (Unforgeability). *Let $t < n/3$. An SRDS scheme Π is t-unforgeable with a bare PKI (resp., with a trusted PKI) if for* mode = b-pki *(resp.,* mode = tr-pki*) and for every (stateful) PPT adversary $\mathcal{A}$ it holds that*

$$\Pr\left[\mathsf{Expt}^{\mathsf{forge}}_{\mathsf{mode},\Pi,\mathcal{A}}(\kappa, n, t) = 1\right] \le \mathsf{negl}(\kappa, n).$$

The experiment $\mathsf{Expt}^{\mathsf{forge}}_{\mathsf{mode},\Pi,\mathcal{A}}$ is defined in Figure 2.

Experiment $\mathsf{Expt}^{\mathsf{forge}}_{\mathsf{mode},\Pi,\mathcal{A}}(\kappa, n, t)$

The experiment $\mathsf{Expt}^{\mathsf{forge}}$ is a game between a challenger and the adversary $\mathcal{A}$. The game is parametrized by an SRDS scheme Π and consists of the following phases:

A. **Setup and Corruption.** As in the robustness experiment in Figure 1.

B. **Forgery Challenge.** In this phase, the adversary tries to forge a signature.
   (a) $\mathcal{A}$ chooses a subset $\mathcal{S} \subseteq [n] \setminus \mathcal{I}$ such that $|\mathcal{S} \cup \mathcal{I}| < n/3$. It also chooses messages m and $\{m_i\}_{i \in \mathcal{S}}$ from $\mathcal{M}$.
   (b) For every $i \in \mathcal{S}$, compute $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{pp}, i, \mathsf{sk}_i, m_i)$.
   (c) For every $i \notin (\mathcal{S} \cup \mathcal{I})$, compute $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{pp}, i, \mathsf{sk}_i, m)$.
   (d) Send $\{\sigma_i\}_{i \in [n] \setminus \mathcal{I}}$ to $\mathcal{A}$ and get back $\sigma' \in \mathcal{X}$ and $m' \in \mathcal{M}$.

C. **Output Phase.** Output 1 if and only if $\mathsf{Verify}(\mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, m', \sigma') = 1$ and $m' \neq m$.

Figure 2: Forgery experiment for SRDS

We note that as described, the security definition is only for one-time SRDS signatures. Although this is sufficient for our applications in Section 3, it is possible to extend the definition and provide the adversary an oracle access to signatures of honest parties on messages of its choice. However, in that case, the adversary must choose the set $\mathcal{S}$ before getting oracle access.

Security. We say that an SRDS scheme is secure in the respective PKI model, if it satisfies all the above properties.

Definition 2.6 (Secure SRDS). *Let $t < n/3$. An SRDS scheme Π is t-secure with a bare PKI (resp., with a trusted PKI) if it is succinct, t-unforgeable and t-robust with a bare PKI (resp., with a trusted PKI).*

2.2 SRDS Constructions

In this section, we present a high-level overview of our two constructions of SRDS that offer a tradeoff between cryptographic and setup assumptions. The first assumes one-way functions in the trusted-PKI model, and the second assumes collision-resistant hash functions (CRH) and succinct non-interactive arguments of knowledge (SNARKs) with linear extraction in the common random string (CRS) and bare-PKI model. Due to space constraints, we defer formal descriptions to the full version of our paper [12].

SRDS from One-Way Functions. Our first construction is influenced by the "sortition approach" of Algorand [22] and merely requires one-way functions (OWF); however, the public-key infrastructure (PKI) is assumed to be *honestly* generated (either by the parties themselves or by an external trusted third party), and corrupted parties cannot alter their keys. The construction is based on digital signatures augmented with an oblivious key-generation algorithm for sampling a verification key without knowing the corresponding signing key.[16] Lamport's signatures [49], which are based on OWF, can easily be adjusted to support this property. To establish the PKI, every party decides whether to generate its public verification key obliviously or together with a signing key by tossing a biased coin, such that with overwhelming probability all but polylog(n) keys are generated obliviously. Since those with the ability to sign are determined at random (as part of the trusted PKI), only parties who hold a signing key can sign messages. The oblivious key-generation algorithm ensures that an adversary who only sees a list of verification keys, cannot distinguish between the keys that have a corresponding signing key and ones that do not. As a result, even if the adversary chooses the set of corrupt parties after the keys are sampled, with a high probability, the fraction of honest parties will be preserved in the signing subset. SRDS signature-aggregation is done by concatenation, and verification of an SRDS signature requires counting how many valid signatures were signed on the message.

It would be desirable to reduce the trust assumption in establishing the PKI, e.g., by using verifiable pseudorandom functions (VRF) [54] as done in [22]. However, this approach [22] is defined within a blockchain model where a fresh random string (the hash of the recent block) is assumed to be consistently available to all parties later in the protocol and serves as the seed for the sortition; equivalently, that parties have access to a common random string (CRS) *independent* of corrupted parties' public keys. Without this extra model assumption, their VRF approach does not apply. We note that several recent consensus protocols [1, 8, 18, 19, 25, 26, 58, 59] also follow the sortition approach of [22]; however, similar to our first construction, their PKI is assumed to be honestly generated by a trusted third party.

Theorem 2.7 (SRDS from OWF and trusted PKI, informal). *Let $\beta < 1/3$ and assume that one-way functions exist. Then, there exists a βn-secure SRDS in the trusted-PKI model.*

SRDS from CRH and SNARKs. Our second construction is based on a weaker bare-PKI setup, in which each party locally computes its signature keys, and the adversary can corrupt parties and change their keys as a function of honest parties' public keys.

In a simplified case where all of the nodes in the almost-everywhere communication tree are honest, a naïve construction would be to have all parties sign the message using their private keys and send the signature to their respective leaf nodes. The leaf nodes would then count the number of verified signatures received and send the message and the counter to their parents. In a recursive way, each node would simply add the counters received from its child nodes and send it to its parent. As a result, the root node would get a final count of the total number of verified signatures. This approach completely breaks, however, if even one node is not honest. To enforce an honest behavior of the nodes, we need to make sure that the aggregation is done in a verifiable way, i.e., ensure that the bad nodes send a valid count of the number of signatures aggregated so far.

Towards this, our first idea is to require each node to attach a "succinct proof" of honest behavior to their messages. In particular, in addition to the message m and count c that a leaf node sends to its parent, it must also send a proof to convince the parent that it knows c distinct signatures on the message m. Similarly, nodes on the next level must prove that they received sufficiently many valid proofs from the leaf nodes and so on. To verify, it is sufficient to check at the root node, whether sufficiently many "base" signatures were aggregated. Such a solution, however, requires proof systems that support *recursive composition*. For this reason, we use *proof-carrying data* (PCD) systems [23].

A PCD system extends the notion of *succinct non-interactive arguments of knowledge* (SNARKs) to the distributed setting by allowing recursive composition in a succinct way. Informally, every party can generate a succinct proof on some statement, certifying that it satisfies a given local property with respect to its private input and previously received messages (statements and their proofs). Bitansky et al. [7] proved that PCD systems for logarithmic-depth DAGs exist assuming SNARKs with *linear extraction*, i.e., where the size of the extractor is linear in the size of the prover.[17] Extractability assumptions of this kind have been considered in, e.g., [14, 27, 40, 57]. Since PCD systems allow for propagation of information up a communication tree in a succinct and publicly verifiable way, they seem to exactly capture our requirements for SRDS.

[16] We note that standard signatures can be used if we strengthen the model assumptions, e.g., by assuming that a party can securely erase its signature key, or by considering a trusted party that only provides the verification keys to some parties. We opted not to rely on stronger model assumption since we can establish signatures with oblivious key generation from the minimal assumption of one-way functions.

[17] We note that although SNARKs with linear extraction are a stronger assumption than standard SNARKs (with polynomial extraction), standard SNARKs techniques do not separate the two notions.

This simple idea, however, is vulnerable to an adversary that generates a valid-looking aggregate signature by using multiple copies of the same signature. Indeed, since the partially aggregated signature must be succinct, the parties cannot afford to keep track of *which* base signatures were already incorporated, leaving them vulnerable to a repeat occurrence. We protect against such an attack by encoding additional information in the partially aggregated signatures using collision-resistant hash functions (CRH).

Theorem 2.8 (SRDS from CRH, SNARKs, and bare PKI, informal). *Let $\beta < n/3$ and assume that CRH and SNARKs with linear extraction exist. Then, there exists a βn-secure SRDS in the CRS and bare-PKI model.*

3 BALANCED COMMUNICATION-EFFICIENT BYZANTINE AGREEMENT

In this section, we consider Byzantine agreement protocols with $\tilde{O}(1)$ communication per party. We show how to use succinctly reconstructed distributed signatures (SRDS) to boost almost-everywhere agreement to full agreement in a balanced way via a single communication round. In particular, we show how to combine SRDS with the protocol of Boyle et al. [13] to obtain BA with balanced $\tilde{O}(1)$ communication. We prove the following theorem in the full version of our paper [12].

Theorem 3.1 (Theorem 1.1, restated). *Let $\beta < 1/3$ and assume existence of a βn-secure SRDS scheme in the bare-PKI model (resp., trusted PKI model). Then, there exists a βn-resilient BA protocol for generating the SRDS setup and the relevant PKI, such that:*

- *The round complexity and communication locality are* $\mathrm{polylog}(n)$*; every party communicates* $\mathrm{polylog}(n) \cdot \mathrm{poly}(\kappa)$ *bits.*
- *The adversary can adaptively corrupt the parties based on the public setup and the PKI before the onset of the protocol. For bare PKI, the adversary can additionally replace the corrupted parties' public keys.*

By instantiating Theorem 3.1 with our SRDS constructions from Section 2.2, we get the following corollaries.

Corollary 3.2. *Let $\beta < 1/3$. Assuming OWF, there exists a βn-resilient BA protocol in the trusted-PKI model with balanced $\tilde{O}(1)$ communication per party.*

Corollary 3.3. *Let $\beta < 1/3$. Assuming CRH and SNARKs with linear extraction, there exists a βn-resilient BA protocol in the bare-PKI and CRS model with balanced $\tilde{O}(1)$ communication per party.*

In Section 3.1, we define the sub-functionalities to be used in the BA protocol and in Section 3.2 we give our protocol.

3.1 Functionalities used in the Protocol

We start by describing the sub-functionalities used in our construction. Due to space constraints, we defer formal specification of these functionalities to the full version of the paper [12].

Almost-everywhere communication. The functionality $f_{\text{ae-comm}}$ is a reactive functionality that abstracts the properties obtained by the protocol from [48]. In the first invocation, the adversary specifies a special communication tree that allows all honest parties to communicate, except for a $o(1)$ fraction of isolated parties $\mathcal{D} \subset [n]$. In all subsequent calls the "supreme committee," i.e., the parties associated with the root of the tree, can send messages to all of the parties but $\mathcal{D}$. We use a slightly modified version of the $(n, \mathcal{I})$-party almost-everywhere-communication tree defined in Section 2. Specifically, in Definition 2.3, each party was assigned to a single leaf node of the tree. Here, each party in the BA protocol will be assigned to multiple leaf nodes (but will participate in the SRDS aggregation as multiple "virtual" parties, one for each appearance).

Definition 3.4 ($(n, \mathcal{I})$ almost-everywhere-communication tree with repeated parties). *Let $\mathcal{I} \subseteq [n]$ be a subset of size βn for a constant $\beta < 1/3$. A directed rooted tree $T = (V, E)$ is an $(n, \mathcal{I})$-almost-everywhere-communication tree with repeated parties if it satisfies the first four properties of an $(n, \mathcal{I})$-party almost-everywhere-communication tree (Definition 2.3) and additionally, the following properties are satisfied:*

(1) Each leaf node of the tree is assigned a set of $\log^5 n$ *parties.*
(2) Each party is assigned to $O(\log^4 n)$ *nodes at each level.*

As observed in [13], the fact that $1 - o(1)$ fraction of the leaves are on good paths to the root implies that for a $1 - o(1)$ fraction of the parties, a majority of the leaf nodes that they are assigned to are good. The protocol of King et al. [48] securely realizes $f_{\text{ae-comm}}$ in the authenticated-channels model tolerating a computationally unbounded, malicious adversary statically corrupting βn parties, for a constant $\beta < 1/3$. Every invocation requires $\mathrm{polylog}(n)$ rounds, and every party sends and processes $\mathrm{polylog}(n)$ bits. Throughout all invocations, every party sends to, and processes messages received from, $\mathrm{polylog}(n)$ other parties.

Byzantine agreement. We consider the standard Byzantine agreement functionality f_{ba} (to be used within small committees in the larger protocol). Every party sends its input to the trusted party who forwards the input value to the adversary. If more than $n - t$ inputs equal the same value $y \in \{0, 1\}$, then deliver y as the output for every party. Otherwise, let the adversary choose the value $y \in \{0, 1\}$ to be delivered.

The n-party BA protocol of Garay and Moses [36] realizes f_{ba} over authenticated channels tolerating a computationally unbounded, malicious adversary statically corrupting $t < n/3$ parties using $t + 1$ rounds and $\mathrm{poly}(n)$ communication complexity. An immediate corollary is that for $n' = \mathrm{polylog}(n)$, the n'-party BA functionality f_{ba} can be instantiated using $\mathrm{polylog}(n)$ rounds and $\mathrm{polylog}(n)$ communication complexity.

Coin tossing. The coin-tossing functionality f_{ct} samples a uniformly distributed $s \in \{0, 1\}^\kappa$ and delivers s to all the parties. The protocol of Chor et al. [24] realizes f_{ct} over a broadcast channel assuming an honest majority (by having each party verifiably secret share (VSS) a random value, and later reconstruct all values and XOR them). By instantiating the broadcast channel using the protocol of [36], $n' = \mathrm{polylog}(n)$ parties can agree on a random κ-bit string in $\mathrm{polylog}(n)$ rounds and $\mathrm{polylog}(n) \cdot \mathrm{poly}(\kappa)$ communication.

Signature aggregation. The signature-aggregation functionality $f_{\text{aggr-sig}}$ is an n'-party functionality, where every party P_i provides a message m_i and a set of signatures. The functionality first determines the set of signatures received from a majority of the parties and aggregates only those signatures to obtain a new signature σ, which is delivered as the output for every party.

Protocol π_{ba}

- **Common Input:** An SRDS scheme and a pseudo-random function (PRF) family $\mathcal{F} = \{F_s\}_{s\in\{0,1\}^\kappa}$ mapping elements of $[n]$ to subsets of $[n]$ of size $\text{polylog}(n)$.
- **Private Input:** Every P_i, for $i \in [n]$, has input $x_i \in \{0,1\}$.
- **Setup:** Let $z = O(\log^4 n)$, $z^* = O(\log^5 n)$ and let $\text{pp} \leftarrow \text{Setup}(1^\kappa, 1^{n\cdot z})$. Every party P_i locally computes $(\text{vk}_{i,j}, \text{sk}_{i,j}) \leftarrow \text{KeyGen}(\text{pp})$ for every $j \in [z]$. The public output consists of pp and the set of public keys $\text{vk} = \{\text{vk}_{i,j}\}_{i\in[n],j\in[z]}$. We assume that there exists a mapping $\text{idmap} : [n] \times [z] \to [n \cdot z]$ that maps the each (i, j) above to a virtual ID $i^* \in [n \cdot z]$, such that virtual IDs of the parties assigned and corresponding to the k^{th} leaf node belong in the range $[(k-1)\cdot z^* + 1, k \cdot z^*]$ (This ensures that when the tree topology is expressed flat as a planar graph (no crossovers), then the virtual IDs of the leaf nodes are in increasing order.).
- **Hybrid Model:** The protocol is defined in the $(f_{\text{ae-comm}}, f_{\text{ba}}, f_{\text{ct}}, f_{\text{aggr-sig}})$-hybrid model.
- **The Protocol:**

(1) Every party invokes $f_{\text{ae-comm}}$ and receives back its local view in the communication tree $T = (V, E)$. Let C denote supreme committee, i.e., the parties assigned to the root node.
(2) Every party P_i in the supreme committee (i.e., with $i \in C$) proceeds as follows. Invoke f_{ba} on its input value x_i to receive back $y \in \{0,1\}$ and invoke f_{ct} to receive back $s \in \{0,1\}^\kappa$.
(3) The parties in the supreme committee C send (y, s) to $f_{\text{ae-comm}}$. For every $i \in [n]$ denote the output of party P_i as (y_i, s_i).
(4) Every party P_i signs the received message (y_i, s_i) for each virtual identity $j \in [z]$ as $\sigma_{i,j} \leftarrow \text{Sign}(\text{pp}, \text{idmap}(i, j), \text{sk}_{i,j}, (y_i, s_i))$. Let $L_i = \{v_{i_1}, \ldots, v_{i_z}\} \subseteq V$ be the subset of leaves assigned to P_i. For each $j \in [z]$, P_i sends $\sigma_{i,j}$ to all the parties assigned to the leaf node v_{i_j}.
(5) Denote by $\text{party}(v)$ the set of parties assigned to a node $v \in V$. Similarly, denote by $\text{child}(v)$ and $\text{parent}(v)$ the set of children nodes and parent node of $v \in V$, resp. Let $\text{range}(v)$ denote the range of virtual IDs of the parties assigned to the leaf nodes that have a path to node $v \in V$. For each level $\ell = 1, \ldots, \ell^*$ and for each node v on level ℓ, the protocol proceeds as follows:
   (a) For each $i \in \text{party}(v)$, let $S_{\text{sig}}^{i,\ell,1}$ be the set of signatures received by P_i in the previous round (for $\ell = 1$, i.e., for leaf nodes, from each P_j with $v \in L_j$; for $\ell > 1$, from every party P_j assigned to a child node of v).
   (b) Every P_i with $i \in \text{party}(v)$ broadcasts the set $S_{\text{sig}}^{i,\ell,1}$ to all the parties in $\text{party}(v)$. Let $S_{\text{sig}}^{i,\ell,2}$ be the union of all sets received from the parties in $\text{party}(v)$.
   (c) Every P_i with $i \in \text{party}(v)$ computes $\text{Aggregate}_1(\text{pp}, \{\text{vk}_{1,1}, \ldots, \text{vk}_{n,z}\}, (y_i, s_i), S_{\text{sig}}^{i,\ell,2}) \to S_{\text{sig}}^{i,\ell,3}$. If $\ell = 1$, for each sig in $S_{\text{sig}}^{i,\ell,3}$ it checks if $\min(\text{sig}) = \max(\text{sig})$ and if $\min(\text{sig}) \in \text{range}(v)$ and if $\ell > 1$, it checks if $\exists v' \in \text{child}(v)$ such that the range $[\min(\text{sig}), \max(\text{sig})]$ falls within the range $\text{range}(v')$. If this check fails for any sig, it updates $S_{\text{sig}}^{i,\ell,3} = S_{\text{sig}}^{i,\ell,3} \setminus \{\text{sig}\}$. It invokes $f_{\text{aggr-sig}}$ on input $((y_i, s_i), S_{\text{sig}}^{i,\ell,3})$ to obtain the aggregated signature σ_v.
   (d) If $\ell < \ell^*$, for each $i \in \text{party}(v)$, party P_i sends σ_v to all parties in $\text{parent}(v)$.
(6) Let σ_{root} be the signature obtained by the supreme committee. The parties in the supreme committee send $(y, s, \sigma_{\text{root}})$ to $f_{\text{ae-comm}}$. Let the output of party P_i for $i \in [n]$ be (y_i', s_i', σ_i')
(7) Each party P_i (for $i \in [n]$) computes $C_i = F_{s_i'}(i)$, and sends (y_i', s_i', σ_i') to every party in C_i, where F is a pseudo-random function.
(8) A party P_j that receives a valid message (y, s, σ) from a party P_i, satisfying $j \in F_s(i)$ and $\text{Verify}(\text{pp}, \{\text{vk}_{1,1}, \ldots, \text{vk}_{n,z}\}, (y, s), \sigma) = 1$, outputs y and halts.

Figure 3: Byzantine agreement with balanced polylog **communication**

Assuming the existence of OWF, the protocol of Damgård and Ishai [28] can be used to realize the n'-party functionality $f_{\text{aggr-sig}}$, for $n' = \text{polylog}(n)$, over secure channels, tolerating a malicious adversary corrupting a minority of the parties. In addition, if the size of set S_{sig} is $\tilde{O}(1)$ the protocol requires $\text{polylog}(n) \cdot \text{poly}(\kappa)$ communication. In our construction, this functionality is used by the parties assigned to a node (in the almost-everywhere communication-tree obtained from $f_{\text{ae-comm}}$) for aggregating signatures received from parties assigned to their children. From Definition 3.4, we know that each node only has $\log(n)$ child nodes and each node is assigned $\text{polylog}(n)$ parties. Therefore, $f_{\text{aggr-sig}}$ is only used for aggregating at most $\text{polylog}(n)$ signatures. Note that in [28] a broadcast channel is also required and the resulting protocol is constant round. For $n' = \text{polylog}(n)$ the broadcast can be realized by a deterministic protocol, e.g., from [36], and the resulting protocol has $\text{polylog}(n)$ rounds and $\text{polylog}(n) \cdot \text{poly}(\kappa)$ communication.

3.2 Byzantine Agreement Protocol

Having defined all the sub-functionalities, we are now ready to present our BA protocol in Figure 3.

The parties communicate in a way that mimics almost-everywhere agreement. As described in Definition 3.4, each party is assigned to $z = O(\log^4 n)$ leaf nodes and $z^* = O(\log^5 n)$ parties are assigned to each leaf node in the communication tree. Since each party will send a signature to every leaf node it is assigned to, it is essential to ensure that the same fraction of signatures is generated by corrupted parties as their fraction in the party-set. For this reason, we allocate z "virtual identities" to every party. The SRDS is used for $n \cdot z$ virtual identities and each party samples separate SRDS keys for each of its virtual identities.

Once *certified* almost-everywhere agreement on (y, s) is reached, full agreement is obtained as in [13]. Every party P_i sends its SRDS-signed pair (y, s), to a (pseudo)random subset of $\text{polylog}(n)$ parties defined by a pseudorandom function (PRF) F on the seed s and its identity i. Receiving parties verify the SRDS on (y, s) and that it was supposed to receive a message from the sender (using s). The complete proof is deferred to the full version [12].

ACKNOWLEDGMENTS

E. Boyle's research is supported in part by ISF grant 1861/16 and AFOSR Award FA9550-17-1-0069 and ERC project HSS (852952). R. Cohen's research is supported in part by NSF grant 1646671. A. Goel's work was done in part while visiting the FACT Center at IDC Herzliya, Israel. Her research is supported in part by an NSF CNS grant 1814919, NSF CAREER award 1942789 and Johns Hopkins University Catalyst award.

REFERENCES

[1] Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. 2019. Communication Complexity of Byzantine Agreement, Revisited. In *Proceedings of the 38th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. 317–326.

[2] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. 2019. Synchronous Byzantine Agreement with Expected O(1) Rounds, Expected $O(n^2)$ Communication, and Optimal Resilience. In *Financial Cryptography and Data Security*. 320–334.

[3] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. 2012. Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE. In *31st Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. 483–501.

[4] Mihir Bellare and Gregory Neven. 2006. Multi-signatures in the plain public-Key model and a general forking lemma. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*. 390–399.

[5] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. 1988. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*. 1–10.

[6] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. 2017. The Hunting of the SNARK. *Journal of Cryptology* 30, 4 (2017), 989–1066.

[7] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. 2013. Recursive composition and bootstrapping for SNARKs and proof-carrying data. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*. 111–120.

[8] Erica Blum, Jonathan Katz, Chen-Da Liu-Zhang, and Julian Loss. 2020. Asynchronous Byzantine Agreement with Subquadratic Communication. In *Proceedings of the 18th Theory of Cryptography Conference (TCC), part I*. 353–380.

[9] Alexandra Boldyreva. 2003. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In *Proceedings of the 6th International Conference on the Theory and Practice of Public-Key Cryptography (PKC)*. 31–46.

[10] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. 2003. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *22nd International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. 416–432.

[11] Elette Boyle, Ran Cohen, Deepesh Data, and Pavel Hubáček. 2018. Must the Communication Graph of MPC Protocols be an Expander?. In *38th Annual International Cryptology Conference (CRYPTO), part III*. 243–272.

[12] Elette Boyle, Ran Cohen, and Aarushi Goel. 2020. Breaking the $O(\sqrt{n})$-Bits Barrier: Byzantine Agreement with Polylog Bits Per-Party. Cryptology ePrint Archive, Report 2020/130. https://eprint.iacr.org/2020/130.

[13] Elette Boyle, Shafi Goldwasser, and Stefano Tessaro. 2013. Communication Locality in Secure Multi-party Computation - How to Run Sublinear Algorithms in a Distributed Setting. In *Proceedings of the 10th Theory of Cryptography Conference (TCC)*. 356–376.

[14] Elette Boyle, Abhishek Jain, Manoj Prabhakaran, and Ching-Hua Yu. 2018. The Bottleneck Complexity of Secure Multiparty Computation. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP)*. 24:1–24:16.

[15] Nicolas Braud-Santoni, Rachid Guerraoui, and Florian Huc. 2013. Fast Byzantine agreement. In *Proceedings of the 32th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. 57–64.

[16] Ran Canetti. 2004. Universally Composable Signature, Certification, and Authentication. In *17th IEEE Computer Security Foundations Workshop, (CSFW)*. 219.

[17] Ran Canetti, Daniel Shahaf, and Margarita Vald. 2016. Universally Composable Authentication and Key-Exchange with Global PKI. In *Proceedings of the 19th International Conference on the Theory and Practice of Public-Key Cryptography (PKC), part II*. 265–296.

[18] T.-H. Hubert Chan, Rafael Pass, and Elaine Shi. 2019. Consensus Through Herding. In *38th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), part I*. 720–749.

[19] T.-H. Hubert Chan, Rafael Pass, and Elaine Shi. 2020. Sublinear-Round Byzantine Agreement Under Corrupt Majority. In *Proceedings of the 23rd International Conference on the Theory and Practice of Public-Key Cryptography (PKC), part II*. 246–265.

[20] Nishanth Chandran, Wutichai Chongchitmate, Juan A. Garay, Shafi Goldwasser, Rafail Ostrovsky, and Vassilis Zikas. 2015. The Hidden Graph Model: Communication Locality and Optimal Resiliency with Adaptive Faults. In *Proceedings of the 6th Annual Innovations in Theoretical Computer Science (ITCS) conference*. 153–162.

[21] David Chaum, Claude Crépeau, and Ivan Damgård. 1988. Multiparty Unconditionally Secure Protocols (Extended Abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*. 11–19.

[22] Jing Chen and Silvio Micali. 2019. Algorand: A secure and efficient distributed ledger. *Theoretical Computer Science* 777 (2019), 155–183.

[23] Alessandro Chiesa and Eran Tromer. 2010. Proof-Carrying Data and Hearsay Arguments from Signature Cards. In *Innovations in Computer Science - ICS*. 310–331.

[24] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. 1985. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract). In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing (STOC)*. 383–395.

[25] Ran Cohen, Iftach Haitner, Nikolaos Makriyannis, Matan Orland, and Alex Samorodnitsky. 2019. On the Round Complexity of Randomized Byzantine Agreement. In *Proceedings of the 33rd International Symposium on Distributed Computing (DISC)*. 12:1–12:17.

[26] Shir Cohen, Idit Keidar, and Alexander Spiegelman. 2020. Not a COINcidence: Sub-Quadratic Asynchronous Byzantine Agreement WHP. In *Proceedings of the 34th International Symposium on Distributed Computing (DISC)*. 25:1–25:17.

[27] Ivan Damgård, Sebastian Faust, and Carmit Hazay. 2012. Secure Two-Party Computation with Low Communication. In *Proceedings of the 9th Theory of Cryptography Conference (TCC)*. 54–74.

[28] Ivan Damgård and Yuval Ishai. 2005. Constant-Round Multiparty Computation Using a Black-Box Pseudorandom Generator. In *24th Annual International Cryptology Conference (CRYPTO)*. 378–394.

[29] Ivan Damgård and Yuval Ishai. 2006. Scalable Secure Multiparty Computation. In *25th Annual International Cryptology Conference (CRYPTO)*. 501–520.

[30] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam D. Smith. 2008. Scalable Multiparty Computation with Nearly Optimal Work and Resilience. In *27th Annual International Cryptology Conference (CRYPTO)*. 241–261.

[31] Yvo Desmedt and Yair Frankel. 1989. Threshold Cryptosystems. In *8th Annual International Cryptology Conference (CRYPTO)*. 307–315.

[32] Danny Dolev. 1982. The Byzantine Generals Strike Again. *J. Algorithms* 3, 1 (1982), 14–30.

[33] Danny Dolev and Rüdiger Reischuk. 1985. Bounds on Information Exchange for Byzantine Agreement. *J. ACM* 32, 1 (1985), 191–204.

[34] Cynthia Dwork, David Peleg, Nicholas Pippenger, and Eli Upfal. 1988. Fault Tolerance in Networks of Bounded Degree. *SIAM J. Comput.* 17, 5 (1988), 975–988.

[35] Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. 1986. Easy Impossibility Proofs for Distributed Consensus Problems. *Distributed Computing* 1, 1 (1986), 26–39.

[36] Juan A. Garay and Yoram Moses. 1993. Fully polynomial Byzantine agreement in t+1 rounds. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC)*. 31–41.

[37] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. 2001. Robust Threshold DSS Signatures. *Inf. Comput.* 164, 1 (2001), 54–84.

[38] Craig Gentry and Daniel Wichs. 2011. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*. 99–108.

[39] Oded Goldreich, Silvio Micali, and Avi Wigderson. 1987. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*. 218–229.

[40] Divya Gupta and Amit Sahai. 2014. On Constant-Round Concurrent Zero-Knowledge from a Knowledge Assumption. In *INDOCRYPT*. 71–88.

[41] Dan Holtby, Bruce M. Kapron, and Valerie King. 2008. Lower bound for scalable Byzantine Agreement. *Distributed Computing* 21, 4 (2008), 239–248.

[42] K. Itakura and K. Nakamura. 1983. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development* 71 (1983), 1–8.

[43] Jonathan Katz and Chiu-Yuen Koo. 2006. On Expected Constant-Round Protocols for Byzantine Agreement. In *25th Annual International Cryptology Conference (CRYPTO)*. 445–462.

[44] Dafna Kidron and Yehuda Lindell. 2011. Impossibility Results for Universal Composability in Public-Key Models and with Fixed Inputs. *Journal of Cryptology* 24, 3 (2011), 517–544.

[45] Valerie King, Steven Lonargan, Jared Saia, and Amitabh Trehan. 2011. Load Balanced Scalable Byzantine Agreement through Quorum Building, with Full Information. In *Proceedings of the 12th International Conference on Distributed Computing and Networking (ICDCN)*. 203–214.

[46] Valerie King and Jared Saia. 2009. From Almost Everywhere to Everywhere: Byzantine Agreement with $\tilde{O}(n^{3/2})$ Bits. In *Proceedings of the 23th International Symposium on Distributed Computing (DISC)*. 464–478.

[47] Valerie King and Jared Saia. 2011. Breaking the $O(n^2)$ bit barrier: scalable Byzantine agreement with an adaptive adversary. *J. ACM* 58, 4 (2011), 18:1–18:24.

[48] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. 2006. Scalable leader election. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 990–999.

[49] Leslie Lamport. 1979. *Constructing Digital Signatures from a One Way Function* (sri international ed.). Technical Report CSL-98. SRI International.
[50] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. 1982. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems* 4, 3 (1982), 382–401.
[51] Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin. 2006. On the composition of authenticated Byzantine Agreement. *J. ACM* 53, 6 (2006), 881–917.
[52] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. 2013. Sequential Aggregate Signatures, Multisignatures, and Verifiably Encrypted Signatures Without Random Oracles. *Journal of Cryptology* 26, 2 (2013), 340–373.
[53] Silvio Micali. 1994. CS Proofs (Extended Abstracts). In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*. 436–453.
[54] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. 1999. Verifiable Random Functions. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*. 120–130.
[55] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. 1980. Reaching Agreement in the Presence of Faults. *J. ACM* 27, 2 (1980), 228–234.
[56] Tal Rabin and Michael Ben-Or. 1989. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract). In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*. 73–85.
[57] Paul Valiant. 2008. Incrementally Verifiable Computation or Proofs of Knowledge Imply Time/Space Efficiency. In *Proceedings of the 5th Theory of Cryptography Conference (TCC)*. 1–18.
[58] Jun Wan, Hanshen Xiao, Srinivas Devadas, and Elaine Shi. 2020. Round-Efficient Byzantine Broadcast Under Strongly Adaptive and Majority Corruptions. In *Proceedings of the 18th Theory of Cryptography Conference (TCC), part I*. 412–456.
[59] Jun Wan, Hanshen Xiao, Elaine Shi, and Srinivas Devadas. 2020. Expected Constant Round Byzantine Broadcast Under Dishonest Majority. In *Proceedings of the 18th Theory of Cryptography Conference (TCC), part I*. 381–411.
[60] Andrew Chi-Chih Yao. 1982. Protocols for Secure Computations (Extended Abstract). In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS)*. 160–164.